



NEMO ocean engine

Version 4.0.1 - October, 2019

- © Romain Bourdallé-Badie
- Mike Bell
- Jérôme Chanut
- © Emanuela Clementi
- © Andrew Coward
- © Massimiliano Drudi
- Christian Éthé
- © Doroteaciro Iovino
- Dan Lea
- © Claire Lévy
- © Gurvan Madec
- Nicolas Martin
- © Sébastien Masson
- Pierre Mathiot
- © Silvia Mocavero
- Simon Müller
- George Nurser
- © Guillaume Samson
- Dave Storkey

Abstract

NEMO (“Nucleus for European Modelling of the Ocean”) is a framework of ocean-related engines. It is intended to be a flexible tool for studying the ocean dynamics and thermodynamics (“blue ocean”), as well as its interactions with the components of the Earth climate system over a wide range of space and time scales. Within *NEMO*, the ocean engine is interfaced with a sea-ice model (*SI³* or *CICE*), passive tracers and biogeochemical models (*TOP*) and, via the *OASIS* coupler, with several atmospheric general circulation models. It also supports two-way grid embedding by means of the *AGRIF* software.

The primitive equation model is adapted to regional and global ocean circulation problems down to kilometeric scale. Prognostic variables are the three-dimensional velocity field, a non-linear sea surface height, the *Conservative* Temperature and the *Absolute* 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, so as various HPC functionalities to improve performances.



Community

Ocean

Model

Disclaimer






Like all components of the modelling framework, the NEMO core engine 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 development 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 authors assume no responsibility for problems, errors, or incorrect usage of *NEMO*.

Other resources

Additional information can be found on:

-  the [website](#) of the project detailing several associated applications and an exhaustive users bibliography
-  the [development platform](#) of the model with the code repository for the shared reference and some main resources (wiki, ticket system, forums, . . .)
-  the [repository of the demonstration cases](#) for research or training
-  the [online archive](#) delivering the publications issued by the consortium (manuals, reports, datasets, . . .)
-  two mailing lists: the [newsletter](#) for top-down communications from the project (announcements, calls, job opportunities, . . .) and the [forge updates](#) (commits, tickets and forums)

Citation

Reference for papers and other publications is as follows:

“**NEMO ocean engine**”, *Scientific Notes of Climate Modelling Center*, **27** — ISSN 1288-1619, Institut Pierre-Simon Laplace (IPSL), [doi:10.5281/zenodo.1464816](https://doi.org/10.5281/zenodo.1464816)

List of Figures

1.1. Ocean boundary conditions	3
1.2. Geographical and curvilinear coordinate systems	5
1.3. Curvilinear z -coordinate systems ($\{non-\}$ linear free-surface cases and re-scaled z^*)	9
2.1. Leapfrog time stepping sequence with split-explicit free surface	17
2.2. Forcing integration methods for modified leapfrog (top and bottom)	18
3.1. Arrangement of variables in the unit cell of space domain	21
3.2. Comparison of grid-point position, vertical grid-size and scale factors	22
3.3. Horizontal integer indexing	23
3.4. Vertical integer indexing	24
3.5. Ocean bottom regarding coordinate systems (z , s and hybrid $s - z$)	26
4.1. Ways to evaluate the tracer value and the amount of tracer exchanged	31
4.2. Penetration profile of the downward solar irradiance calculated by four models	39
4.3. Geothermal heat flux	40
4.4. Advective/diffusive bottom boundary layer	41
4.5. Discretisation of the horizontal difference and average of tracers in the z -partial step coordinate	46
5.1. Triads used in the energy and enstrophy conserving scheme (EEN)	51
5.2. Split-explicit time stepping scheme for the external and internal modes	58
5.3. Combinations controlling the limiting of the horizontal pressure gradient in wetting and drying regimes	64
6.1. Ice shelf location and fresh water flux definition	82
6.2. Reconstruction of the diurnal cycle variation of short wave flux	88
6.3. Reconstruction of the diurnal cycle variation of short wave flux on an ORCA2 grid	89
7.1. Lateral boundary at T -level	92
7.2. Lateral boundary conditions	94
7.3. Setting of east-west cyclic and symmetric across the Equator boundary conditions	95
7.4. North fold boundary in ORCA 2° , $1/4^\circ$ and $1/12^\circ$	95
7.5. Positioning of a sub-domain when massively parallel processing is used	96
7.6. Atlantic domain defined for the CLIPPER projet	97
7.7. Geometry of unstructured open boundary	102
7.8. Header for a <i>coordinates.bdy.nc</i> file	103
8.1. Averaging procedure for isopycnal slope computation	108
8.2. Vertical profile of the slope used for lateral mixing in the mixed layer	109
9.1. Mixing length computation	117
9.2. The structure of the entraining boundary layer. (a) Mean buoyancy profile. (b) Profile of the buoyancy flux.	121
9.3. Subgrid kinetic energy integration in GLS and TKE schemes	123
9.4. Unstable density profile treated by the non penetrative convective adjustment algorithm	125
9.5. Diapycnal diffusivities for temperature and salt in regions of salt fingering and diffusive convection	127
9.6. Partitioning coefficient used to partition vertical velocities into parts	133

9.7. OVERFLOW: time-series of temperature vertical cross-sections	134
9.8. OVERFLOW: sample temperature vertical cross-sections from mid- and end-run	135
9.9. OVERFLOW: maximum partitioning coefficient during a series of test runs	135
9.10. OVERFLOW: maximum partitioning coefficient for the case overlaid	135
10.1. Decomposition of the World Ocean to compute transports as well as the meridional stream-function	157
11.1. Observational weights with a rectangular footprint	168
11.2. Observational weights with a radial footprint	169
11.3. Observations with the geographical distribution	170
11.4. Observations with the round-robin distribution	171
11.5. Main window of dataplot	176
11.6. Profile plot from dataplot	177
14.1. Two methods to defined the Gibraltar strait	186
14.2. Mask fields for the <i>closea.F90</i> module	186
15.1. ORCA mesh conception	193
15.2. Horizontal scale factors and ratio of anisotropy for ORCA 0.5° mesh	194
15.3. Snapshot of relative vorticity at the surface of the model domain in GYRE R9, R27 and R54	196
D.1. Triads arrangement and tracer gradients to give lateral and vertical tracer fluxes	229
D.2. Triad notation for quarter cells	229
D.3. Boundary triads	233
D.4. Definition of mixed-layer depth and calculation of linearly tapered triads	239
E.1. DOMAINcfg: default vertical mesh for ORCA2	242
E.2. DOMAINcfg: examples of the stretching function applied to a seamount	247
E.3. DOMAINcfg: comparison of <i>s</i> - and <i>z</i> -coordinate	247

List of Namelists

2.1.	&namrun	19
3.1.	&namdom	25
3.2.	&namtsd	28
4.1.	&namtra_adv	30
4.2.	&namtra_ldf	34
4.3.	&namtra_qsr	38
4.4.	&nambbc	40
4.5.	&namtbl	40
4.6.	&namtra_dmp	42
4.7.	&nameos	44
5.1.	&namdyn_adv	49
5.2.	&namdyn_vor	49
5.3.	&namdyn_hpg	54
5.4.	&namdyn_spg	56
5.5.	&namdyn_ldf	59
5.6.	&namwad	61
6.1.	&namsbc	67
6.2.	&namsbc_sas	72
6.3.	&namsbc_flx	74
6.4.	&namsbc_blk	74
6.5.	&namsbc_cpl	76
6.6.	&namsbc_apr	77
6.7.	&nam_tide	78
6.8.	&namsbc_rnf	78
6.9.	&namsbc_isf	80
6.10.	&namsbc_iscpl	83
6.11.	&namberg	84
6.12.	&namsbc_wave	85
6.13.	&namsbc_ssr	90
7.1.	&namlbc	92
7.2.	&nammpp	95
7.3.	&nambdy	98
7.4.	&nambdy_dta	99
7.5.	&nambdy_tide	104
8.1.	&namtra_eiv	112
8.2.	&namtra_mle	112
9.1.	&namzdf	114
9.2.	&namzdf_ric	115

9.3.	&namzdf_tke	116
9.4.	&namzdf_gls	119
9.5.	&namzdf_osm	120
9.6.	&namdrg	127
9.7.	&namdrg_top	127
9.8.	&namdrg_bot	128
9.9.	&namzdf_iwm	131
10.1.	&namnc4	149
10.2.	&namtrd	151
10.3.	&namflo	151
10.4.	&nam_diaharm	153
10.5.	&nam_diadct	153
10.6.	&namptr	157
10.7.	&nam_dia25h	158
10.8.	&nam_diatmb	158
11.1.	&namobs	161
12.1.	&nam_asminc	180
13.1.	&namsto	183
14.1.	&namctl	189
15.1.	&namcfg	192
E.1.	&namdom_domcfg	241
E.2.	&namzgr_sco_domcfg	246

List of Tables

3.1. Location of grid-points	21
4.1. Standard value of S-EOS coefficients	45
6.1. Ocean variables provided to the surface module)	69
6.2. Naming nomenclature for climatological or interannual input file	70
8.1. Description of expected input files if mixing coefficients are read from NetCDF files	110
9.1. Set of predefined GLS parameters or equivalently predefined turbulence models available	120
9.2. Advective CFL criteria for the leapfrog with Robert Asselin filter time-stepping	132
10.1. XIOS: context tags	146
10.2. XIOS: field tags ("field_*")	146
10.3. XIOS: file tags ("file_*")	146
10.4. XIOS: axis tags ("axis_*")	147
10.5. XIOS: domain tags ("domain_*")	147
10.6. XIOS: grid tags ("grid_*")	147
10.7. XIOS: reference attributes ("*_ref")	147
10.8. XIOS: domain attributes ("zoom_*")	147
10.9. XIOS: file attributes	148
10.10 XIOS: field attributes	148
10.11 XIOS: miscellaneous attributes	149
10.12 Filesize comparison between NetCDF3 and NetCDF4 with chunking and compression	150
15.1. Domain size of ORCA family configurations	195
E.1. Default vertical mesh in z -coordinate for 30 layers ORCA2 configuration	244

The **Nucleus for European Modelling of the Ocean (NEMO)** is a framework of ocean related engines, namely the aforementioned for the ocean dynamics and thermodynamics, SI^{3*} for the sea-ice dynamics and thermodynamics, TOP^\dagger for the biogeochemistry (both transport and sources minus sinks ($PISCES^\ddagger$)). The ocean component has been developed from the legacy of the OPA^\S model, described in Madec et al. (1998). 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 sea-ice and/or the atmosphere.

This manual 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. For the use of framework, a guide which gathers the README files spread out in the source code can be build and exported in a web or printable format (see `./doc/rst`). Also a online copy is available on the [forge platform](#).

Manual outline

Chapters

The manual mirrors the organization of the model and it is organised in as follows: after the presentation of the continuous equations (primitive equations with temperature and salinity, and an equation of seawater) in the next chapter, the following chapters refer to specific terms of the equations each associated with a group of modules.

Model Basics presents the equations and their assumptions, the vertical coordinates used, and the subgrid scale physics.

The equations are written in a curvilinear coordinate system, with a choice of vertical coordinates (z , s , z^* , s^* , \tilde{z} , \tilde{s} , and a mix of them). Momentum equations are formulated in vector invariant or flux form. Dimensional units in the meter, kilogram, second (MKS) international system are used throughout. The following chapters deal with the discrete equations.

Time Domain presents the model time stepping environment. it 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.

Space Domain (DOM) presents the model **DOM**ain. It 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).

Ocean Tracers (TRA) and Ocean Dynamics (DYN) describe the discretisation of the prognostic equations for the active **TR**acers (potential temperature and salinity) and the momentum (**DY**Namic). 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).

*Sea-Ice modelling Integrated Initiative

†Tracer in the Ocean Paradigm

‡Pelagic Interactions Scheme for Carbon and Ecosystem Studies

§Océan PARallélisé (French)

Surface Boundary Condition (SBC, SAS, ISF, ICB) 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 (*SI³*) and a biogeochemistry model (*PISCES*). Interactive coupling to Atmospheric models is possible via the *OASIS* coupler. Two-way nesting is also available through an interface to the *AGRIF* package, *i.e.* **Adaptative Grid Refinement in Fortran** (Debreu et al., 2008). The interface code for coupling to an alternative sea ice model (*CICE*) has now been upgraded so that it works for both global and regional domains.

Lateral Boundary Condition (LBC) presents the **Lateral BounDarY** Conditions. 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.

Lateral Ocean Physics (LDF) and Vertical Ocean Physics (ZDF) describe the physical parameterisations (**Lateral DiFfusion** and vertical **Z DiFfusion**) The model includes an implicit treatment of vertical viscosity and diffusivity. The lateral Laplacian and biharmonic viscosity and diffusion can be rotated following a geopotential or neutral direction. There is an optional eddy induced velocity (Gent and McWilliams, 1990) with a space and time variable coefficient Tréguier et al. (1997). The model has vertical harmonic viscosity and diffusion with a space and time variable coefficient, with options to compute the coefficients with Blanke and Delécluse (1993), Pacanowski and Philander (1981), or Umlauf and Burchard (2003) mixing schemes.

Output and Diagnostics (IOM, DIA, TRD, FLO) describes model **In-Outputs** Management and specific online **DI**Agnostics. The diagnostics includes the output of all the tendencies of the momentum and tracers equations, the output of tracers **TR**enDs averaged over the time evolving mixed layer, the output of the tendencies of the barotropic vorticity equation, the computation of on-line **FLO**ats trajectories...

Observation and Model Comparison (OBS) describes a tool which reads in **OBS**ervation 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 fantastic tool for model and data comparison.

Apply Assimilation Increments (ASM) describes how increments produced by data **AsSi**Milation may be applied to the model equations.

Stochastic Parametrization of EOS (STO)

Miscellaneous Topics (including solvers)

Configurations provides finally a brief introduction to the pre-defined model configurations (water column model *C1D*, ORCA and GYRE families of configurations).

Appendices

Curvilinear s -Coordinate Equations

Diffusive Operators

Discrete Invariants of the Equations

Iso-Neutral Diffusion and Eddy Advection using Triads

A brief guide to the DOMAINcfg tool

Coding Rules

1. Model Basics	1
1.1. Primitive equations	2
1.1.1. Vector invariant formulation	2
1.1.2. Boundary conditions	3
1.2. Horizontal pressure gradient	4
1.2.1. Pressure formulation	4
1.2.2. Free surface formulation	4
1.3. Curvilinear z -coordinate system	4
1.3.1. Tensorial formalism	4
1.3.2. Continuous model equations	5
1.4. Curvilinear generalised vertical coordinate system	7
1.4.1. S -coordinate formulation	8
1.4.2. Curvilinear z^* -coordinate system	9
1.4.3. Curvilinear terrain-following s -coordinate	10
1.4.4. Curvilinear \tilde{z} -coordinate	11
1.5. Subgrid scale physics	11
1.5.1. Vertical subgrid scale physics	11
1.5.2. Formulation of the lateral diffusive and viscous operators	12
2. Time Domain	14
2.1. Time stepping environment	15
2.2. Non-diffusive part — Leapfrog scheme	15
2.3. Diffusive part — Forward or backward scheme	15
2.4. Surface pressure gradient	16
2.5. Modified LeapFrog – Robert Asselin filter scheme (LF-RA)	16
2.6. Start/Restart strategy	18
3. Space Domain (DOM)	20
3.1. Fundamentals of the discretisation	21
3.1.1. Arrangement of variables	21
3.1.2. Discrete operators	22
3.1.3. Numerical indexing	23
3.2. Spatial domain configuration	24
3.2.1. Domain size	24
3.2.2. Horizontal grid mesh (<i>domhgr.F90</i>)	25
3.2.3. Vertical grid (<i>domzgr.F90</i>)	25
3.2.4. Closed seas	27
3.2.5. Output grid files	27
3.3. Initial state (<i>istate.F90</i> and <i>dtatsd.F90</i>)	28
4. Ocean Tracers (TRA)	29
4.1. Tracer advection (<i>traadv.F90</i>)	30
4.1.1. CEN: Centred scheme (<i>ln_traadv_cen</i>)	32
4.1.2. FCT: Flux Corrected Transport scheme (<i>ln_traadv_fct</i>)	32

4.1.3.	MUSCL: Monotone Upstream Scheme for Conservative Laws (<code>ln_traadv_mus</code>)	33
4.1.4.	UBS a.k.a. UP3: Upstream-Biased Scheme (<code>ln_traadv_ubs</code>)	33
4.1.5.	QCK: QuiCKest scheme (<code>ln_traadv_qck</code>)	34
4.2.	Tracer lateral diffusion (<code>traldf.F90</code>)	34
4.2.1.	Type of operator (<code>ln_traldf_{OFF, lap, blp}</code>)	34
4.2.2.	Action direction (<code>ln_traldf_{lev, hor, iso, triad}</code>)	35
4.2.3.	Iso-level (bi-)laplacian operator (<code>ln_traldf_iso</code>)	35
4.2.4.	Standard and triad (bi-)laplacian operator	36
4.3.	Tracer vertical diffusion (<code>trazdf.F90</code>)	36
4.4.	External forcing	37
4.4.1.	Surface boundary condition (<code>trasbc.F90</code>)	37
4.4.2.	Solar radiation penetration (<code>traqsr.F90</code>)	38
4.4.3.	Bottom boundary condition (<code>trabbc.F90</code>) - <code>ln_trabbc</code>	39
4.5.	Bottom boundary layer (<code>trabbl.F90</code> - <code>ln_trabbl</code>)	39
4.5.1.	Diffusive bottom boundary layer (<code>nn_bbl_ldf=1</code>)	41
4.5.2.	Advective bottom boundary layer (<code>nn_bbl_adv=1, 2</code>)	41
4.6.	Tracer damping (<code>tradmp.F90</code>)	42
4.7.	Tracer time evolution (<code>tranxt.F90</code>)	43
4.8.	Equation of state (<code>eosbn2.F90</code>)	43
4.8.1.	Equation of seawater (<code>ln_{teos10, eos80, seos}</code>)	43
4.8.2.	Brunt-Väisälä frequency	45
4.8.3.	Freezing point of seawater	45
4.9.	Horizontal derivative in <i>zps</i> -coordinate (<code>zpsjde.F90</code>)	45
5.	Ocean Dynamics (DYN)	47
5.1.	Sea surface height and diagnostic variables (η, ζ, χ, w)	48
5.1.1.	Horizontal divergence and relative vorticity (<code>divcur.F90</code>)	48
5.1.2.	Horizontal divergence and relative vorticity (<code>sshwzv.F90</code>)	48
5.2.	Coriolis and advection: vector invariant form	49
5.2.1.	Vorticity term (<code>dynvor.F90</code>)	49
5.2.2.	Kinetic energy gradient term (<code>dynkeg.F90</code>)	52
5.2.3.	Vertical advection term (<code>dynzad.F90</code>)	52
5.3.	Coriolis and advection: flux form	52
5.3.1.	Coriolis plus curvature metric terms (<code>dynvor.F90</code>)	52
5.3.2.	Flux form advection term (<code>dynadv.F90</code>)	53
5.4.	Hydrostatic pressure gradient (<code>dynhpg.F90</code>)	54
5.4.1.	Full step <i>Z</i> -coordinate (<code>ln_dynhpg_zco</code>)	54
5.4.2.	Partial step <i>Z</i> -coordinate (<code>ln_dynhpg_zps</code>)	54
5.4.3.	<i>S</i> - and <i>Z-S</i> -coordinates	54
5.4.4.	Ice shelf cavity	55
5.4.5.	Time-scheme (<code>ln_dynhpg_imp</code>)	55
5.5.	Surface pressure gradient (<code>dynspg.F90</code>)	56
5.5.1.	Explicit free surface (<code>ln_dynspg_exp</code>)	56
5.5.2.	Split-explicit free surface (<code>ln_dynspg_ts</code>)	56
5.5.3.	Filtered free surface (<code>dynspg_filt</code> ?)	57
5.6.	Lateral diffusion term and operators (<code>dynldf.F90</code>)	57
5.6.1.	Iso-level laplacian (<code>ln_dynldf_lap</code>)	59
5.6.2.	Rotated laplacian (<code>ln_dynldf_iso</code>)	59
5.6.3.	Iso-level bilaplacian (<code>ln_dynldf_bilap</code>)	60
5.7.	Vertical diffusion term (<code>dynzdf.F90</code>)	60
5.8.	External forcings	61
5.9.	Wetting and drying	61
5.9.1.	Directional limiter (<code>wet_dry.F90</code>)	62
5.9.2.	Iterative limiter (<code>wet_dry.F90</code>)	62
5.9.3.	The WAD test cases (<code>usrdef_zgr.F90</code>)	65
5.10.	Time evolution term (<code>dynnxt.F90</code>)	65
6.	Surface Boundary Condition (SBC, SAS, ISF, ICB)	66
6.1.	Surface boundary condition for the ocean	68
6.2.	Input data generic interface	69
6.2.1.	Input data specification (<code>fldread.F90</code>)	69

6.2.2.	Interpolation on-the-fly	71
6.2.3.	Standalone surface boundary condition scheme (SAS)	72
6.3.	Flux formulation (<i>sbcflx.F90</i>)	73
6.4.	Bulk formulation (<i>sbcblk.F90</i>)	73
6.4.1.	Ocean-Atmosphere Bulk formulae (<i>sbcblk_algo_coare.F90, sbcblk_algo_coare3p5.F90, sbcblk_algo_ecmwf.F90, sbcblk_algo_ncar.F90</i>)	75
6.4.2.	Ice-Atmosphere Bulk formulae	75
6.5.	Coupled formulation (<i>sbcopl.F90</i>)	76
6.6.	Atmospheric pressure (<i>sbcapr.F90</i>)	77
6.7.	Surface tides (<i>sbctide.F90</i>)	77
6.8.	River runoffs (<i>sbcrrf.F90</i>)	78
6.9.	Ice shelf melting (<i>sbcisf.F90</i>)	79
6.10.	Ice sheet coupling	82
6.11.	Handling of icebergs (ICB)	83
6.12.	Interactions with waves (<i>sbcwave.F90, ln_wave</i>)	85
6.12.1.	Neutral drag coefficient from wave model (<i>ln_cdgw</i>)	86
6.12.2.	3D Stokes Drift (<i>ln_sdw & nn_sdrift</i>)	86
6.12.3.	Stokes-Coriolis term (<i>ln_stcor</i>)	87
6.12.4.	Wave modified stress (<i>ln_tauwoc & ln_tauw</i>)	87
6.13.	Miscellaneous options	87
6.13.1.	Diurnal cycle (<i>sbcncy.F90</i>)	87
6.13.2.	Rotation of vector pairs onto the model grid directions	87
6.13.3.	Surface restoring to observed SST and/or SSS (<i>sbcssr.F90</i>)	88
6.13.4.	Handling of ice-covered area (<i>sbcice_...</i>)	88
6.13.5.	Interface to CICE (<i>sbcice_cice.F90</i>)	89
6.13.6.	Freshwater budget control (<i>sbcfwb.F90</i>)	90
7.	Lateral Boundary Condition (LBC)	91
7.1.	Boundary condition at the coast (<i>rn_shlat</i>)	92
7.2.	Model domain boundary condition (<i>jperio</i>)	93
7.2.1.	Closed, cyclic (=0, 1, 2, 7)	93
7.2.2.	North-fold (=3, 6)	93
7.3.	Exchange with neighbouring processors (<i>lbclnk.F90, lib_mpp.F90</i>)	95
7.4.	Unstructured open boundary conditions (BDY)	97
7.4.1.	Namelist	98
7.4.2.	Flow relaxation scheme	100
7.4.3.	Flather radiation scheme	101
7.4.4.	Orlanski radiation scheme	101
7.4.5.	Relaxation at the boundary	101
7.4.6.	Boundary geometry	102
7.4.7.	Input boundary data files	102
7.4.8.	Volume correction	102
7.4.9.	Tidal harmonic forcing	103
8.	Lateral Ocean Physics (LDF)	105
8.1.	Lateral mixing operators	106
8.1.1.	No lateral mixing (<i>ln_traldf_OFF & ln_dynldf_OFF</i>)	106
8.1.2.	Laplacian mixing (<i>ln_traldf_lap & ln_dynldf_lap</i>)	106
8.1.3.	Bilaplacian mixing (<i>ln_traldf_blp & ln_dynldf_blp</i>)	106
8.2.	Direction of lateral mixing (<i>ldfslp.F90</i>)	106
8.2.1.	Slopes for tracer geopotential mixing in the <i>s</i> -coordinate	106
8.2.2.	Slopes for tracer iso-neutral mixing	107
8.2.3.	Slopes for momentum iso-neutral mixing	108
8.3.	Lateral mixing coefficient (<i>nn_aht_ijk_t & nn_ahm_ijk_t</i>)	109
8.3.1.	Mixing coefficients read from file (=20, -30)	110
8.3.2.	Constant mixing coefficients (=0)	110
8.3.3.	Vertically varying mixing coefficients (=10)	110
8.3.4.	Mesh size dependent mixing coefficients (=20)	110
8.3.5.	Mesh size and depth dependent mixing coefficients (=30)	110
8.3.6.	Velocity dependent mixing coefficients (=31)	111
8.3.7.	Deformation rate dependent viscosities (<i>nn_ahm_ijk_t=32</i>)	111

8.3.8.	About space and time varying mixing coefficients	111
8.4.	Eddy induced velocity (<code>ln_ldfeiv</code>)	111
8.5.	Mixed layer eddies (<code>ln_mle</code>)	112
9.	Vertical Ocean Physics (ZDF)	113
9.1.	Vertical mixing	114
9.1.1.	Constant (<code>ln_zdfcst</code>)	114
9.1.2.	Richardson number dependent (<code>ln_zdfric</code>)	114
9.1.3.	TKE turbulent closure scheme (<code>ln_zdf_tke</code>)	115
9.1.4.	GLS: Generic Length Scale (<code>ln_zdf_gls</code>)	119
9.1.5.	OSM: OSMOSIS boundary layer scheme (<code>ln_zdf_osc = .true.</code>)	120
9.1.6.	Discrete energy conservation for TKE and GLS schemes	123
9.2.	Convection	124
9.2.1.	Non-penetrative convective adjustment (<code>ln_tranpc</code>)	125
9.2.2.	Enhanced vertical diffusion (<code>ln_zdfevd</code>)	125
9.2.3.	Handling convection with turbulent closure schemes (<code>ln_zdf_{tke, gls, osc}</code>)	126
9.3.	Double diffusion mixing (<code>ln_zdfddm</code>)	126
9.4.	Bottom and top friction (<code>zdfdr.F90</code>)	127
9.4.1.	Linear top/bottom friction (<code>ln_lin</code>)	128
9.4.2.	Non-linear top/bottom friction (<code>ln_non_lin</code>)	129
9.4.3.	Log-layer top/bottom friction (<code>ln_loglayer</code>)	129
9.4.4.	Explicit top/bottom friction (<code>ln_drgimp=.false.</code>)	129
9.4.5.	Implicit top/bottom friction (<code>ln_drgimp=.true.</code>)	130
9.4.6.	Bottom friction with split-explicit free surface	130
9.5.	Internal wave-driven mixing (<code>ln_zdfiwm</code>)	131
9.6.	Surface wave-induced mixing (<code>ln_zdfswm</code>)	132
9.7.	Adaptive-implicit vertical advection (<code>ln_zad_Aimp</code>)	132
9.7.1.	Adaptive-implicit vertical advection in the OVERFLOW test-case	133
10.	Output and Diagnostics (IOM, DIA, TRD, FLO)	136
10.1.	Model output	137
10.2.	Standard model output (IOM)	137
10.2.1.	XIOS: Reading and writing restart file	138
10.2.2.	XIOS: XML Inputs-Outputs Server	138
10.2.3.	Practical issues	139
10.2.4.	XML fundamentals	140
10.2.5.	Detailed functionalities	142
10.2.6.	XML reference tables	144
10.2.7.	CF metadata standard compliance	145
10.3.	NetCDF4 support (<code>key_netcdf4</code>)	149
10.4.	Tracer/Dynamics trends (<code>&namtrd</code>)	150
10.5.	FLO: On-Line Floats trajectories (<code>key_floats</code>)	151
10.6.	Harmonic analysis of tidal constituents (<code>key_diaharm</code>)	153
10.7.	Transports across sections (<code>key_diadct</code>)	153
10.8.	Diagnosing the steric effect in sea surface height	155
10.9.	Other diagnostics	157
10.9.1.	Depth of various quantities (<code>diahth.F90</code>)	157
10.9.2.	CMIP specific diagnostics (<code>diar5.F90</code> , <code>diapr.F90</code>)	157
10.9.3.	25 hour mean output for tidal models	157
10.9.4.	Top middle and bed hourly output	158
10.9.5.	Courant numbers	158
11.	Observation and Model Comparison (OBS)	159
11.1.	Running the observation operator code example	160
11.2.	Technical details (feedback type observation file headers)	161
11.2.1.	Profile feedback file	161
11.2.2.	Sea level anomaly feedback file	163
11.2.3.	Sea surface temperature feedback file	165
11.3.	Theoretical details	166
11.3.1.	Horizontal interpolation and averaging methods	166
11.3.2.	Grid search	168

11.3.3. Parallel aspects of horizontal interpolation	169
11.3.4. Vertical interpolation operator	169
11.4. Standalone observation operator	172
11.4.1. Concept	172
11.4.2. Using the standalone observation operator	172
11.4.3. Configuring the standalone observation operator	172
11.5. Observation utilities	173
11.5.1. Obstacles	173
11.5.2. Building the obstacles	174
11.5.3. Dataplot	175
12. Apply Assimilation Increments (ASM)	178
12.1. Direct initialization	179
12.2. Incremental analysis updates	179
12.3. Divergence damping initialisation	179
12.4. Implementation details	180
13. Stochastic Parametrization of EOS (STO)	181
13.1. Stochastic processes	182
13.2. Implementation details	183
14. Miscellaneous Topics	184
14.1. Representation of unresolved straits	185
14.1.1. Hand made geometry changes	185
14.2. Closed seas (<i>closea.F90</i>)	185
14.3. Sub-domain functionality	187
14.3.1. Simple subsetting of input files via NetCDF attributes	187
14.4. Accuracy and reproducibility (<i>lib_fortran.F90</i>)	188
14.4.1. Issues with intrinsic SIGN function (key_nosignedzero)	188
14.4.2. MPP reproducibility	188
14.4.3. MPP scalability	188
14.5. Model optimisation, control print and benchmark	189
14.5.1. Vector optimisation	189
14.5.2. Control print	189
15. Configurations	191
15.1. Introduction	192
15.2. CID: 1D Water column model (key_c1d)	192
15.3. ORCA family: global ocean with tripolar grid	192
15.3.1. ORCA tripolar grid	193
15.3.2. ORCA pre-defined resolution	193
15.4. GYRE family: double gyre basin	195
15.5. AMM: atlantic margin configuration	196
A. Curvilinear s-Coordinate Equations	197
A.1. Chain rule for s -coordinates	198
A.2. Continuity equation in s -coordinates	198
A.3. Momentum equation in s -coordinate	199
A.4. Tracer equation	202
B. Diffusive Operators	204
B.1. Horizontal/Vertical 2^{nd} order tracer diffusive operators	205
B.2. Iso/Diapycnal 2^{nd} order tracer diffusive operators	206
B.3. Lateral/Vertical momentum diffusive operators	208
C. Discrete Invariants of the Equations	209
C.1. Introduction / Notations	210
C.2. Continuous conservation	211
C.3. Discrete total energy conservation: vector invariant form	213
C.3.1. Total energy conservation	213
C.3.2. Vorticity term (coriolis + vorticity part of the advection)	213
C.3.3. Pressure gradient term	216

C.4.	Discrete total energy conservation: flux form	217
C.4.1.	Total energy conservation	217
C.4.2.	Coriolis and advection terms: flux form	217
C.5.	Discrete enstrophy conservation	218
C.6.	Conservation properties on tracers	220
C.6.1.	Advection term	220
C.7.	Conservation properties on lateral momentum physics	220
C.7.1.	Conservation of potential vorticity	221
C.7.2.	Dissipation of horizontal kinetic energy	221
C.7.3.	Dissipation of enstrophy	222
C.7.4.	Conservation of horizontal divergence	222
C.7.5.	Dissipation of horizontal divergence variance	222
C.8.	Conservation properties on vertical momentum physics	222
C.9.	Conservation properties on tracer physics	224
C.9.1.	Conservation of tracers	224
C.9.2.	Dissipation of tracer variance	225
D.	Iso-Neutral Diffusion and Eddy Advection using Triads	226
D.1.	Choice of <code>namtra</code> and <code>ldf</code> namelist parameters	227
D.2.	Triad formulation of iso-neutral diffusion	227
D.2.1.	Iso-neutral diffusion operator	227
D.2.2.	Standard discretization	228
D.2.3.	Expression of the skew-flux in terms of triad slopes	228
D.2.4.	Full triad fluxes	229
D.2.5.	Ensuring the scheme does not increase tracer variance	230
D.2.6.	Triad volumes in Griffes's scheme and in <i>NEMO</i>	231
D.2.7.	Summary of the scheme	231
D.2.8.	Treatment of the triads at the boundaries	233
D.2.9.	Limiting of the slopes within the interior	233
D.2.10.	Tapering within the surface mixed layer	234
D.3.	Eddy induced advection formulated as a skew flux	235
D.3.1.	Continuous skew flux formulation	235
D.3.2.	Discrete skew flux formulation	236
D.3.3.	Treatment of the triads at the boundaries	237
D.3.4.	Limiting of the slopes within the interior	238
D.3.5.	Tapering within the surface mixed layer	238
D.3.6.	Streamfunction diagnostics	238
E.	A brief guide to the DOMAINcfg tool	240
E.1.	Choice of horizontal grid	241
E.2.	Vertical grid	242
E.2.1.	Vertical reference coordinate	242
E.2.2.	Model bathymetry	245
E.2.3.	Choice of vertical grid	245
F.	Coding Rules	249
F.1.	Introduction	250
F.2.	Overview and general conventions	250
F.3.	Architecture	251
F.4.	Style rules	251
F.4.1.	Argument list format	251
F.4.2.	Array syntax	251
F.4.3.	Case	251
F.4.4.	Comments	252
F.4.5.	Continuation lines	252
F.4.6.	Declaration of arguments and local variables	252
F.4.7.	F90 Standard	252
F.4.8.	Free-Form Source	252
F.4.9.	Indentation	253
F.4.10.	Loops	253
F.4.11.	Naming Conventions: files	253

F.4.12. Naming Conventions: modules	253
F.4.13. Naming Conventions: variables	253
F.4.14. Operators	253
F.4.15. Pre processor	253
F.5. Content rules	254
F.5.1. Configurations	254
F.5.2. Constants	254
F.5.3. Declaration for variables and constants	254
F.5.4. Headers	255
F.5.5. Interface blocks	255
F.5.6. I/O Error Conditions	255
F.5.7. PRINT - ASCII output files	255
F.5.8. Precision	256
F.5.9. Structures	256
F.6. Packages coding rules	256
F.6.1. Bounds checking	256
F.6.2. Communication	256
F.6.3. Error conditions	256
F.6.4. Memory management	257
F.6.5. Optimisation	258
F.6.6. Package attribute: PRIVATE, PUBLIC, USE, ONLY	258
F.6.7. Parallelism using MPI	258
F.7. Features to be avoided	258

Bibliography **260**

Indexes **267**

Namelist blocks	267
CPP keys	268
FORTTRAN modules	269
Namelist parameters	270
FORTTRAN subroutines	273

Table of contents

1.1. Primitive equations	2
1.1.1. Vector invariant formulation	2
1.1.2. Boundary conditions	3
1.2. Horizontal pressure gradient	4
1.2.1. Pressure formulation	4
1.2.2. Free surface formulation	4
1.3. Curvilinear z -coordinate system	4
1.3.1. Tensorial formalism	4
1.3.2. Continuous model equations	5
1.4. Curvilinear generalised vertical coordinate system	7
1.4.1. S -coordinate formulation	8
1.4.2. Curvilinear z^* -coordinate system	9
1.4.3. Curvilinear terrain-following s -coordinate	10
1.4.4. Curvilinear \tilde{z} -coordinate	11
1.5. Subgrid scale physics	11
1.5.1. Vertical subgrid scale physics	11
1.5.2. Formulation of the lateral diffusive and viscous operators	12

Changes record

Release	Author(s)	Modifications
4.0	<i>Mike Bell</i>	<i>Review</i>
3.6	<i>Tim Graham and Gurvan Madec</i>	<i>Updates</i>
≤ 3.4	<i>Gurvan Madec and Sébastien Masson</i>	<i>First version</i>

1.1. Primitive equations

1.1.1. 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:

Spherical Earth approximation The geopotential surfaces are assumed to be oblate spheroids that follow the Earth's bulge; these spheroids are approximated by spheres with gravity locally vertical (parallel to the Earth's radius) and independent of latitude (White et al., 2005, section 2).

Thin-shell approximation The ocean depth is neglected compared to the earth's radius

Turbulent closure hypothesis The turbulent fluxes (which represent the effect of small scale processes on the large-scale) are expressed in terms of large-scale features

Boussinesq hypothesis Density variations are neglected except in their contribution to the buoyancy force

$$\rho = \rho(T, S, p) \quad (1.1)$$

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)

$$\frac{\partial p}{\partial z} = -\rho g \quad (1.2)$$

Incompressibility hypothesis The three dimensional divergence of the velocity vector \mathbf{U} is assumed to be zero.

$$\nabla \cdot \mathbf{U} = 0 \quad (1.3)$$

Neglect of additional Coriolis terms The Coriolis terms that vary with the cosine of latitude are neglected. These terms may be non-negligible where the Brunt-Väisälä frequency N is small, either in the deep ocean or in the sub-mesoscale motions of the mixed layer, or near the equator (White et al., 2005, section 1). They can be consistently included as part of the ocean dynamics (White et al., 2005, section 3(d)) and are retained in the MIT ocean model.

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 equations:

– the momentum balance

$$\frac{\partial \mathbf{U}_h}{\partial t} = - \left[(\nabla \times \mathbf{U}) \times \mathbf{U} + \frac{1}{2} \nabla (\mathbf{U}^2) \right]_h - f \mathbf{k} \times \mathbf{U}_h - \frac{1}{\rho_o} \nabla_h p + D^{\mathbf{U}} + F^{\mathbf{U}} \quad (1.4a)$$

– the heat and salt conservation equations

$$\frac{\partial T}{\partial t} = -\nabla \cdot (T \mathbf{U}) + D^T + F^T \quad (1.4b)$$

$$\frac{\partial S}{\partial t} = -\nabla \cdot (S \mathbf{U}) + D^S + F^S \quad (1.4c)$$

where ∇ is the generalised derivative vector operator in (i, j, k) directions, t is the time, z is the vertical coordinate, ρ is the *in situ* density given by the equation of state (equation 1.1), ρ_o is a reference density, p the pressure, $f = 2 \cdot \mathbf{k}$ is the Coriolis acceleration (where $\mathbf{\Omega}$ is the Earth's angular velocity vector), and g is the gravitational acceleration. $D^{\mathbf{U}}$, D^T and D^S are the parameterisations of small-scale physics for momentum, temperature and salinity, and $F^{\mathbf{U}}$, F^T and F^S surface forcing terms. Their nature and formulation are discussed in section 1.5 and subsection 1.1.2.

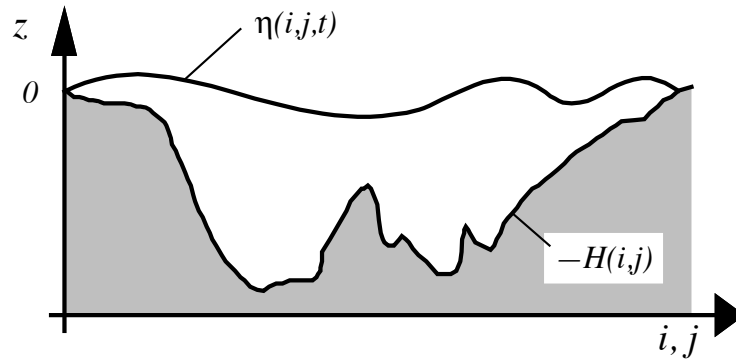


Figure 1.1.: The ocean is bounded by two surfaces, $z = -H(i, j)$ and $z = \eta(i, j, t)$, where H is the depth of the sea floor and η the height of the sea surface. Both H and η are referenced to $z = 0$.

1.1.2. 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 (discretisation can introduce additional artificial “side-wall” boundaries). Both H and η are referenced to a surface of constant geopotential (*i.e.* a mean sea surface height) on which $z = 0$ (figure 1.1). 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 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 Heat and salt fluxes through the sea floor are small, except in special areas of little extent. They are usually neglected in the model *. The boundary condition is thus set to no flux of heat and salt across solid boundaries. For momentum, the situation is different. There is no flow across solid boundaries, *i.e.* the velocity normal to the ocean bottom and coastlines is zero (in other words, the bottom velocity is parallel to solid boundaries). This kinematic boundary condition can be expressed as:

$$w = -U_h \cdot \nabla_h(H) \quad (1.5)$$

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 D^U in equation 1.4a. It is discussed in equation 1.17.

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

$$w = \frac{\partial \eta}{\partial t} + U_h|_{z=\eta} \cdot \nabla_h(\eta) + P - E$$

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 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 ($\sim 4 - 6 \text{ psu}$) compared to those of the ocean ($\sim 34 \text{ psu}$). The cycle of freezing/melting is associated with fresh water and salt fluxes that cannot be neglected.

*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 subsection 4.4.3).

1.2. Horizontal pressure gradient

1.2.1. 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 (equation 1.2), assuming that pressure in decibars can be approximated by depth in meters in (equation 1.1). The hydrostatic pressure is then given by:

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

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 has to be very short when they are 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 this document (see the next sub-section).

1.2.2. 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 (equation 1.5):

$$\frac{\partial \eta}{\partial t} = -D + P - E \quad \text{where} \quad D = \nabla \cdot [(H + \eta) \bar{U}_h] \quad (1.6)$$

and using (equation 1.2) 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 (*i.e.* nearly independent of depth) 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.

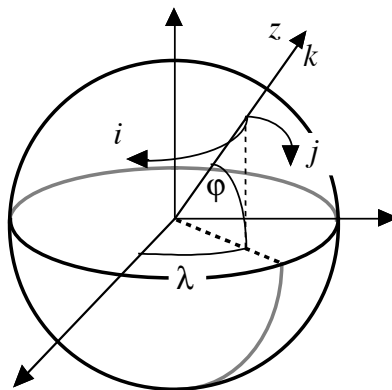
- If one is interested in EGWs, in particular the tides and their interaction with the baroclinic structure of the ocean (internal waves) possibly in shallow seas, then a non linear free surface is the most appropriate. This means that no approximation is made in equation 1.6 and that the variation of 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 subsection 5.5.1 or subsection 5.5.2).
- If one is not interested in EGWs 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 equation 1.6, which still allows to take into account freshwater fluxes applied at the ocean surface (Roulet and Madec, 2000). Nevertheless, with the linearization, an exact conservation of heat and salt contents is lost.

The filtering of EGWs in models with a free surface is usually a matter of discretisation of the temporal derivatives, using a split-explicit method (Killworth et al., 1991; Zhang and Endoh, 1992) or the implicit scheme (Dukowicz and Smith, 1994) or the addition of a filtering force in the momentum equation (Roulet and Madec, 2000). With the present release, *NEMO* offers the choice between an explicit free surface (see subsection 5.5.1) or a split-explicit scheme strongly inspired the one proposed by Shchepetkin and McWilliams (2005) (see subsection 5.5.2).

1.3. Curvilinear z-coordinate system

1.3.1. 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


 Figure 1.2.: the geographical coordinate system (λ, φ, z) and the curvilinear coordinate system (i, j, k) .

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 (Madec and Imbard, 1996; Murray, 1996). 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 Eiseman and Stone (1980) in their survey of the conservation laws of fluid dynamics.

Let (i, j, k) be a set of orthogonal curvilinear coordinates on the sphere associated with the positively oriented orthogonal set of unit vectors (i, j, k) linked to the earth such that k is the local upward vector and (i, j) are two vectors orthogonal to k , *i.e.* along geopotential surfaces (figure 1.2). 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 (figure 1.2). The local deformation of the curvilinear coordinate system is given by e_1, e_2 and e_3 , the three scale factors:

$$e_1 = (a + z) \left[\left(\frac{\partial \lambda}{\partial i} \cos \varphi \right)^2 + \left(\frac{\partial \varphi}{\partial i} \right)^2 \right]^{1/2} \quad e_2 = (a + z) \left[\left(\frac{\partial \lambda}{\partial j} \cos \varphi \right)^2 + \left(\frac{\partial \varphi}{\partial j} \right)^2 \right]^{1/2} \quad e_3 = \left(\frac{\partial z}{\partial k} \right) \quad (1.7)$$

Since the ocean depth is far smaller than the earth's radius, $a + z$, can be replaced by a in (equation 1.7) (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 (equation 1.4a to equation 1.1) can then be written in the tensorial form, invariant in any orthogonal horizontal curvilinear coordinate system transformation:

$$\nabla q = \frac{1}{e_1} \frac{\partial q}{\partial i} \mathbf{i} + \frac{1}{e_2} \frac{\partial q}{\partial j} \mathbf{j} + \frac{1}{e_3} \frac{\partial q}{\partial k} \mathbf{k} \quad (1.8a)$$

$$\nabla \cdot \mathbf{A} = \frac{1}{e_1 e_2} \left[\frac{\partial(e_2 a_1)}{\partial i} + \frac{\partial(e_1 a_2)}{\partial j} \right] + \frac{1}{e_3} \left[\frac{\partial a_3}{\partial k} \right] \quad (1.8b)$$

$$\nabla \times \mathbf{A} = \left[\frac{1}{e_2} \frac{\partial a_3}{\partial j} - \frac{1}{e_3} \frac{\partial a_2}{\partial k} \right] \mathbf{i} + \left[\frac{1}{e_3} \frac{\partial a_1}{\partial k} - \frac{1}{e_1} \frac{\partial a_3}{\partial i} \right] \mathbf{j} + \frac{1}{e_1 e_2} \left[\frac{\partial(e_2 a_2)}{\partial i} - \frac{\partial(e_1 a_1)}{\partial j} \right] \mathbf{k} \quad (1.8c)$$

$$\Delta q = \nabla \cdot (\nabla q) \quad (1.8d)$$

$$\Delta \mathbf{A} = \nabla (\nabla \cdot \mathbf{A}) - \nabla \times (\nabla \times \mathbf{A}) \quad (1.8e)$$

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

1.3.2. 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 equation 1.8a) to equation 1.8e. Let us set $\mathbf{U} = (u, v, w) = U_h + w \mathbf{k}$, the velocity in the (i, j, k) coordinates system, and define the relative vorticity ζ and the divergence of the horizontal

velocity field χ , by:

$$\zeta = \frac{1}{e_1 e_2} \left[\frac{\partial(e_2 v)}{\partial i} - \frac{\partial(e_1 u)}{\partial j} \right] \quad (1.9)$$

$$\chi = \frac{1}{e_1 e_2} \left[\frac{\partial(e_2 u)}{\partial i} + \frac{\partial(e_1 v)}{\partial j} \right] \quad (1.10)$$

Using again 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 , *NLT* the nonlinear term of [equation 1.4a](#) can be transformed as follows:

$$\begin{aligned} NLT &= \left[(\nabla \times \mathbf{U}) \times \mathbf{U} + \frac{1}{2} \nabla (\mathbf{U}^2) \right]_h \\ &= \left(\begin{array}{c} \left[\frac{1}{e_3} \frac{\partial u}{\partial k} - \frac{1}{e_1} \frac{\partial w}{\partial i} \right] w - \zeta v \\ \zeta u - \left[\frac{1}{e_2} \frac{\partial w}{\partial j} - \frac{1}{e_3} \frac{\partial v}{\partial k} \right] w \end{array} \right) + \frac{1}{2} \left(\begin{array}{c} \frac{1}{e_1} \frac{\partial(u^2+v^2+w^2)}{\partial i} \\ \frac{1}{e_2} \frac{\partial(u^2+v^2+w^2)}{\partial j} \end{array} \right) \\ &= \left(\begin{array}{c} -\zeta v \\ \zeta u \end{array} \right) + \frac{1}{2} \left(\begin{array}{c} \frac{1}{e_1} \frac{\partial(u^2+v^2)}{\partial i} \\ \frac{1}{e_2} \frac{\partial(u^2+v^2)}{\partial j} \end{array} \right) + \frac{1}{e_3} \left(\begin{array}{c} w \frac{\partial u}{\partial k} \\ w \frac{\partial v}{\partial k} \end{array} \right) - \left(\begin{array}{c} \frac{w}{e_1} \frac{\partial w}{\partial i} - \frac{1}{2e_1} \frac{\partial w^2}{\partial i} \\ \frac{w}{e_2} \frac{\partial w}{\partial j} - \frac{1}{2e_2} \frac{\partial w^2}{\partial j} \end{array} \right) \end{aligned}$$

The last term of the right hand side is obviously zero, and thus the **NonLinear Term** (*NLT*) of [equation 1.4a](#) is written in the (i, j, k) coordinate system:

$$NLT = \zeta \mathbf{k} \times \mathbf{U}_h + \frac{1}{2} \nabla_h (\mathbf{U}_h^2) + \frac{1}{e_3} w \frac{\partial \mathbf{U}_h}{\partial k} \quad (1.11)$$

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 [equation 1.11](#) (the i -component) is transformed as follows:

$$\begin{aligned} NLT_i &= -\zeta v + \frac{1}{2 e_1} \frac{\partial(u^2 + v^2)}{\partial i} + \frac{1}{e_3} w \frac{\partial u}{\partial k} \\ &= \frac{1}{e_1 e_2} \left(-v \frac{\partial(e_2 v)}{\partial i} + v \frac{\partial(e_1 u)}{\partial j} \right) + \frac{1}{e_1 e_2} \left(e_2 u \frac{\partial u}{\partial i} + e_2 v \frac{\partial v}{\partial i} \right) + \frac{1}{e_3} \left(w \frac{\partial u}{\partial k} \right) \\ &= \frac{1}{e_1 e_2} \left[- \left(v^2 \frac{\partial e_2}{\partial i} + e_2 v \frac{\partial v}{\partial i} \right) + \left(\frac{\partial(e_1 u v)}{\partial j} - e_1 u \frac{\partial v}{\partial j} \right) + \left(\frac{\partial(e_2 u u)}{\partial i} - u \frac{\partial(e_2 u)}{\partial i} \right) + e_2 v \frac{\partial v}{\partial i} \right] \\ &\quad + \frac{1}{e_3} \left(\frac{\partial(w u)}{\partial k} - u \frac{\partial w}{\partial k} \right) \\ &= \frac{1}{e_1 e_2} \left(\frac{\partial(e_2 u u)}{\partial i} + \frac{\partial(e_1 u v)}{\partial j} \right) + \frac{1}{e_3} \frac{\partial(w u)}{\partial k} + \frac{1}{e_1 e_2} \left[-u \left(\frac{\partial(e_1 v)}{\partial j} - v \frac{\partial e_1}{\partial j} \right) - u \frac{\partial(e_2 u)}{\partial i} \right] - \frac{1}{e_3} \frac{\partial w}{\partial k} u \\ &\quad + \frac{1}{e_1 e_2} \left(-v^2 \frac{\partial e_2}{\partial i} \right) \\ &= \nabla \cdot (\mathbf{U} u) - (\nabla \cdot \mathbf{U}) u + \frac{1}{e_1 e_2} \left(-v^2 \frac{\partial e_2}{\partial i} + u v \frac{\partial e_1}{\partial j} \right) \end{aligned}$$

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

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

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

$$NLT = \nabla \cdot \left(\begin{array}{c} \mathbf{U} u \\ \mathbf{U} v \end{array} \right) + \frac{1}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \mathbf{k} \times \mathbf{U}_h \quad (1.12)$$

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:

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

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

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

$$\begin{aligned}\frac{\partial u}{\partial t} &= +(\zeta + f)v - \frac{1}{2e_1} \frac{\partial}{\partial i}(u^2 + v^2) - \frac{1}{e_3} w \frac{\partial u}{\partial k} - \frac{1}{e_1} \frac{\partial}{\partial i} \left(\frac{p_s + p_h}{\rho_o} \right) + D_u^U + F_u^U \\ \frac{\partial v}{\partial t} &= -(\zeta + f)u - \frac{1}{2e_2} \frac{\partial}{\partial j}(u^2 + v^2) - \frac{1}{e_3} w \frac{\partial v}{\partial k} - \frac{1}{e_2} \frac{\partial}{\partial j} \left(\frac{p_s + p_h}{\rho_o} \right) + D_v^U + F_v^U\end{aligned}\quad (1.13)$$

Flux form of the momentum equations

$$\begin{aligned}\frac{\partial u}{\partial t} &= + \left[f + \frac{1}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right] v - \frac{1}{e_1 e_2} \left(\frac{\partial(e_2 u u)}{\partial i} + \frac{\partial(e_1 v u)}{\partial j} \right) - \frac{1}{e_3} \frac{\partial(w u)}{\partial k} \\ &\quad - \frac{1}{e_1} \frac{\partial}{\partial i} \left(\frac{p_s + p_h}{\rho_o} \right) + D_u^U + F_u^U \\ \frac{\partial v}{\partial t} &= - \left[f + \frac{1}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right] u - \frac{1}{e_1 e_2} \left(\frac{\partial(e_2 u v)}{\partial i} + \frac{\partial(e_1 v v)}{\partial j} \right) - \frac{1}{e_3} \frac{\partial(w v)}{\partial k} \\ &\quad - \frac{1}{e_2} \frac{\partial}{\partial j} \left(\frac{p_s + p_h}{\rho_o} \right) + D_v^U + F_v^U\end{aligned}$$

where ζ , the relative vorticity, is given by [equation 1.9](#) and p_s , the surface pressure, is given by:

$$p_s = \rho g \eta$$

and η is the solution of [equation 1.6](#).

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

$$\frac{\partial w}{\partial k} = -\chi e_3 \quad \frac{\partial p_h}{\partial k} = -\rho g e_3$$

where the divergence of the horizontal velocity, χ is given by [equation 1.10](#).

Tracer equations

$$\begin{aligned}\frac{\partial T}{\partial t} &= -\frac{1}{e_1 e_2} \left[\frac{\partial(e_2 T u)}{\partial i} + \frac{\partial(e_1 T v)}{\partial j} \right] - \frac{1}{e_3} \frac{\partial(T w)}{\partial k} + D^T + F^T \\ \frac{\partial S}{\partial t} &= -\frac{1}{e_1 e_2} \left[\frac{\partial(e_2 S u)}{\partial i} + \frac{\partial(e_1 S v)}{\partial j} \right] - \frac{1}{e_3} \frac{\partial(S w)}{\partial k} + D^S + F^S \\ \rho &= \rho(T, S, z(k))\end{aligned}$$

The expression of D^U , D^S and D^T depends on the subgrid scale parameterisation used. It will be defined in [equation 1.17](#). The nature and formulation of F^U , F^T and F^S , the surface forcing terms, are discussed in [chapter 6](#).

1.4. 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 constraints 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) ([Chassignet et al., 2003](#)) or OPA (mixture of z -coordinate in vicinity the surface and steep topography areas and σ -coordinate elsewhere) ([Madec et al., 1996](#)) among others.

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

$$s = s(i, j, k, t) \quad (1.14)$$

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. [equation 1.14](#) 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 [equation 1.14](#). This so-called *generalised vertical coordinate* ([Kasahara, 1974](#)) is in fact an Arbitrary Lagrangian–Eulerian (ALE) coordinate. Indeed, one has a great deal of freedom in the choice of expression for s . The choice 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 sometimes referenced as an adaptive coordinate ([Hofmeister et al., 2010](#)), 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 regriding and remapping steps, the latter step implicitly embedding the vertical advection ([Hirt et al., 1974](#); [Chassignet et al., 2003](#); [White et al., 2009](#)). Here we follow the ([Kasahara, 1974](#)) strategy: a regriding 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 of 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} .

1.4.1. S-coordinate formulation

Starting from the set of equations established in [section 1.3](#) 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 σ -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:

$$\sigma_1 = \frac{1}{e_1} \left. \frac{\partial z}{\partial i} \right|_s \quad \text{and} \quad \sigma_2 = \frac{1}{e_2} \left. \frac{\partial z}{\partial j} \right|_s \quad (1.15)$$

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

$$\omega = w - \left. \frac{\partial z}{\partial t} \right|_s - \sigma_1 u - \sigma_2 v$$

The equations solved by the ocean model [equation 1.4](#) in s -coordinate can be written as follows (see [section A.3](#)):

Vector invariant form of the momentum equation

$$\begin{aligned} \frac{\partial u}{\partial t} &= +(\zeta + f)v - \frac{1}{2e_1} \frac{\partial}{\partial i} (u^2 + v^2) - \frac{1}{e_3} \omega \frac{\partial u}{\partial k} - \frac{1}{e_1} \frac{\partial}{\partial i} \left(\frac{p_s + p_h}{\rho_o} \right) - g \frac{\rho}{\rho_o} \sigma_1 + D_u^U + F_u^U \\ \frac{\partial v}{\partial t} &= -(\zeta + f)u - \frac{1}{2e_2} \frac{\partial}{\partial j} (u^2 + v^2) - \frac{1}{e_3} \omega \frac{\partial v}{\partial k} - \frac{1}{e_2} \frac{\partial}{\partial j} \left(\frac{p_s + p_h}{\rho_o} \right) - g \frac{\rho}{\rho_o} \sigma_2 + D_v^U + F_v^U \end{aligned}$$

Flux form of the momentum equation

$$\begin{aligned} \frac{1}{e_3} \frac{\partial (e_3 u)}{\partial t} &= + \left[f + \frac{1}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right] v - \frac{1}{e_1 e_2 e_3} \left(\frac{\partial (e_2 e_3 u u)}{\partial i} + \frac{\partial (e_1 e_3 v u)}{\partial j} \right) - \frac{1}{e_3} \frac{\partial (\omega u)}{\partial k} \\ &\quad - \frac{1}{e_1} \frac{\partial}{\partial i} \left(\frac{p_s + p_h}{\rho_o} \right) - g \frac{\rho}{\rho_o} \sigma_1 + D_u^U + F_u^U \end{aligned}$$

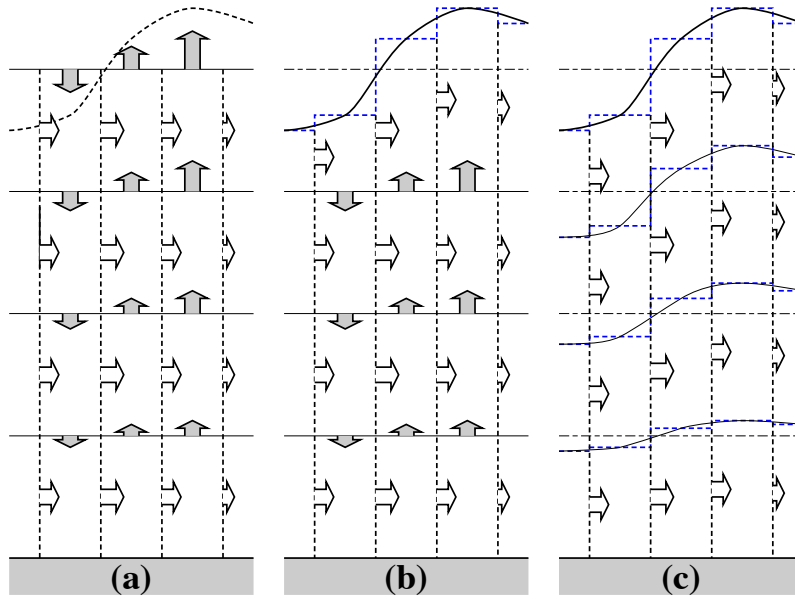


Figure 1.3.: (a) z -coordinate in linear free-surface case; (b) z -coordinate in non-linear free surface case; (c) re-scaled height coordinate (become popular as the z^* -coordinate (Adcroft and Campin, 2004)).

$$\begin{aligned} \frac{1}{e_3} \frac{\partial(e_3 v)}{\partial t} = & - \left[f + \frac{1}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right] u - \frac{1}{e_1 e_2 e_3} \left(\frac{\partial(e_2 e_3 u v)}{\partial i} + \frac{\partial(e_1 e_3 v v)}{\partial j} \right) - \frac{1}{e_3} \frac{\partial(\omega v)}{\partial k} \\ & - \frac{1}{e_2} \frac{\partial}{\partial j} \left(\frac{p_s + p_h}{\rho_o} \right) - g \frac{\rho}{\rho_o} \sigma_2 + D_v^U + F_v^U \end{aligned}$$

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](#)):

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

Tracer equations

$$\begin{aligned} \frac{1}{e_3} \frac{\partial(e_3 T)}{\partial t} = & - \frac{1}{e_1 e_2 e_3} \left(\frac{\partial(e_2 e_3 u T)}{\partial i} + \frac{\partial(e_1 e_3 v T)}{\partial j} \right) - \frac{1}{e_3} \frac{\partial(T \omega)}{\partial k} + D^T + F^S \\ \frac{1}{e_3} \frac{\partial(e_3 S)}{\partial t} = & - \frac{1}{e_1 e_2 e_3} \left(\frac{\partial(e_2 e_3 u S)}{\partial i} + \frac{\partial(e_1 e_3 v S)}{\partial j} \right) - \frac{1}{e_3} \frac{\partial(S \omega)}{\partial k} + D^S + F^S \end{aligned}$$

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

1.4.2. Curvilinear z^* -coordinate system

In this case, the free surface equation is nonlinear, and the variations of volume are fully taken into account. These coordinates systems is presented in a report (Levier et al., 2007) available on the *NEMO* web site.

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 (Adcroft and Campin, 2004). 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 [item 1.3](#). Note that with a flat bottom, such as in [item 1.3](#), 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 (δz^*) are expressed as:

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

Simple re-organisation of the above expressions gives

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

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 continuity equation:

$$\frac{\partial r}{\partial t} = \nabla_{z^*} \cdot (r U_h) + \frac{\partial r w^*}{\partial z^*} = 0$$

This z^* coordinate is closely related to the η coordinate used in many atmospheric models (see Black (1994) for a review of η 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^* are quasi-horizontal. Indeed, the z^* 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^* 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 subsection 1.4.3. Additionally, since $z^* = z$ 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^* models using the same techniques as in z -models (see Chapters 13-16 of Griffies (2004)) for a discussion of neutral physics in z -models, as well as section 8.2 in this document for treatment in *NEMO*).

The range over which z^* varies is time independent $-H \leq z^* \leq 0$. 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^* has a time independent range, all grid cells have static increments ds , and the sum of the vertical increments yields the time independent ocean depth. The z^* coordinate is therefore invisible to undulations of the free surface, since it moves along with the free surface. This property means that no spurious vertical transport is induced across surfaces of constant z^* by the motion of external gravity waves. Such spurious transport can be a problem in z -models, especially those with tidal forcing. Quite generally, the time independent range for the z^* coordinate is a very convenient property that allows for a nearly arbitrary vertical resolution even in the presence of large amplitude fluctuations of the surface height, again so long as $\eta > -H$.

1.4.3. Curvilinear terrain-following s-coordinate

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 (section 1.3), 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 Pacanowski and Gnanadesikan (1998). 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 (no more lateral boundary condition to specify). Nevertheless, a s -coordinate also has its drawbacks. Perfectly adapted to a homogeneous ocean, it has strong limitations as soon as stratification is introduced. The main two problems come from the truncation error in the horizontal pressure gradient and a possibly increased diapycnal diffusion. The horizontal pressure force in s -coordinate consists of two terms (see appendix A),

$$\nabla p|_z = \nabla p|_s - \frac{1}{e_3} \frac{\partial p}{\partial s} \nabla z|_s \quad (1.16)$$

The second term in [equation 1.16](#) depends on the tilt of the coordinate surface and leads to a truncation error that is not present in a z -model. In the special case of a σ -coordinate (i.e. a depth-normalised coordinate system $\sigma = z/H$), [Haney \(1991\)](#) and [Beckmann and Haidvogel \(1993\)](#) have given estimates of the magnitude of this truncation error. It depends on topographic slope, stratification, horizontal and vertical resolution, 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 ([Gerdes, 1993a,b](#); [Madec et al., 1996](#)). However, the definition of the model domain vertical coordinate becomes then a non-trivial thing for a realistic bottom topography: an 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 [subsection 3.2.3](#)).

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) ([Madec et al., 1996](#)). An alternate solution consists of rotating the lateral diffusive tensor to geopotential or to isoneutral surfaces (see [subsection 1.5.2](#)). Unfortunately, the slope of isoneutral surfaces relative to the s -surfaces can very large, strongly exceeding the stability limit of such an operator when it is discretized (see [chapter 8](#)).

The s -coordinates introduced here ([Lott et al., 1990](#); [Madec et al., 1996](#)) 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.

1.4.4. Curvilinear \tilde{z} -coordinate

The \tilde{z} -coordinate has been developed by [Leclair and Madec \(2011\)](#). It is available in *NEMO* since the version 3.4 and is more robust in version 4.0 than previously. Nevertheless, it is currently not robust enough to be used in all possible configurations. Its use is therefore not recommended.

1.5. Subgrid scale physics

The hydrostatic 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}^U , D^S and D^T in [equation 1.4a](#), [equation 1.4b](#) and [equation 1.4c](#) are divided into a lateral part \mathbf{D}^{lU} , D^{lS} and D^{lT} and a vertical part \mathbf{D}^{vU} , D^{vS} and D^{vT} . The formulation of these terms and their underlying physics are briefly discussed in the next two subsections.

1.5.1. 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:

$$D^{vU} = \frac{\partial}{\partial z} \left(A^{vm} \frac{\partial U_h}{\partial z} \right), D^{vT} = \frac{\partial}{\partial z} \left(A^{vT} \frac{\partial T}{\partial z} \right) \text{ and } D^{vS} = \frac{\partial}{\partial z} \left(A^{vT} \frac{\partial S}{\partial z} \right) \quad (1.17)$$

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 [chapter 6](#) and [chapter 9](#) and [section 4.5](#)). All the vertical physics is embedded in the specification of the eddy coefficients. They can be assumed to be either constant, or function of the local fluid properties (*e.g.* Richardson number, Brunt-Väisälä frequency, distance from the boundary ...), or computed from a turbulent closure model. The choices available in *NEMO* are discussed in [chapter 9](#).

1.5.2. Formulation of the lateral diffusive and viscous operators

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 [McDougall \(1987\)](#)) 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). [Gent and McWilliams \(1990\)](#) 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 the more scale selective 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 to ensure the stability of the model while not interfering with the resolved 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 an advective scheme which is diffusive enough to maintain the model stability. It must be emphasised that then, all the sub-grid scale physics is included in the formulation of the advection scheme.

All these parameterisations of subgrid scale physics have advantages and drawbacks. They are not all available in *NEMO*. For active tracers (temperature and salinity) the main ones are: Laplacian and bilaplacian operators acting along geopotential or iso-neutral surfaces, [Gent and McWilliams \(1990\)](#) parameterisation, and various slightly diffusive advection schemes. For momentum, the main ones are: Laplacian and bilaplacian operators acting along geopotential surfaces, and UBS advection schemes when flux form is chosen for the momentum advection.

Lateral laplacian tracer diffusive operator

The lateral Laplacian tracer diffusive operator is defined by (see [appendix B](#)):

$$D^{lT} = \nabla \cdot (A^{lT} \Re \nabla T) \quad \text{with} \quad \Re = \begin{pmatrix} 1 & 0 & -r_1 \\ 0 & 1 & -r_2 \\ -r_1 & -r_2 & r_1^2 + r_2^2 \end{pmatrix} \quad (1.18)$$

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 [equation 1.18](#) 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 [equation 1.18](#) ([Cox, 1987](#)). 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 *iso-level* diffusion, r_1 and r_2 are zero. \Re reduces to the identity in the horizontal direction, no rotation is applied.

For *geopotential* diffusion, r_1 and r_2 are the slopes between the geopotential and computational surfaces: they are equal to σ_1 and σ_2 , respectively (see [equation 1.15](#)).

For *isoneutral* diffusion r_1 and r_2 are the slopes between the isoneutral and computational surfaces. Therefore, they are different quantities, but have similar expressions in z - and s -coordinates. In z -coordinates:

$$r_1 = \frac{e_3}{e_1} \left(\frac{\partial \rho}{\partial i} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1} \quad r_2 = \frac{e_3}{e_2} \left(\frac{\partial \rho}{\partial j} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1} \quad (1.19)$$

while in s -coordinates $\frac{\partial}{\partial k}$ is replaced by $\frac{\partial}{\partial s}$.

Eddy induced velocity

When the *eddy induced velocity* parametrisation (eiv) ([Gent and McWilliams, 1990](#)) is used, an additional tracer advection is introduced in combination with the isoneutral diffusion of tracers:

$$D^{lT} = \nabla \cdot (A^{lT} \Re \nabla T) + \nabla \cdot (U^* T)$$

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

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

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. Their values are thus independent of the vertical coordinate, but their expression depends on the coordinate:

$$\tilde{r}_n = \begin{cases} r_n & \text{in } z\text{-coordinate} \\ r_n + \sigma_n & \text{in } z^*\text{- and } s\text{-coordinates} \end{cases} \quad \text{where } n = 1, 2 \quad (1.20)$$

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. [chapter 8](#)).

Lateral bilaplacian tracer diffusive operator

The lateral bilaplacian tracer diffusive operator is defined by:

$$D^{lT} = -\Delta (\Delta T) \quad \text{where} \quad \Delta \bullet = \nabla \cdot (\sqrt{B^{lT}} \Re \nabla \bullet)$$

It is the Laplacian operator given by [equation 1.18](#) applied twice with the harmonic eddy diffusion coefficient set to the square root of the biharmonic one.

Lateral Laplacian momentum diffusive operator

The Laplacian momentum diffusive operator along z - or s -surfaces is found by applying [equation 1.8e](#) to the horizontal velocity vector (see [appendix B](#)):

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

Such a formulation ensures a complete separation between the vorticity and horizontal divergence fields (see [appendix C](#)). Unfortunately, it is only available in *iso-level* direction. When a rotation is required (*i.e.* geopotential diffusion in s -coordinates or isoneutral diffusion in both z - and s -coordinates), the u and v -fields are considered as independent scalar fields, so that the diffusive operator is given by:

$$D_u^{lU} = \nabla \cdot (A^{lm} \Re \nabla u) \quad D_v^{lU} = \nabla \cdot (A^{lm} \Re \nabla v)$$

where \Re is given by [equation 1.18](#). 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 β -plane, not on the sphere. It is also a very good approximation in vicinity of the Equator in a geographical coordinate system ([Lengaigne et al., 2003](#)).

Lateral bilaplacian momentum diffusive operator

As for tracers, the bilaplacian order momentum diffusive operator is a re-entering Laplacian operator with the harmonic eddy diffusion coefficient set to the square root of the biharmonic one. Nevertheless it is currently not available in the iso-neutral case.

Table of contents

2.1. Time stepping environment	15
2.2. Non-diffusive part — Leapfrog scheme	15
2.3. Diffusive part — Forward or backward scheme	15
2.4. Surface pressure gradient	16
2.5. Modified LeapFrog – Robert Asselin filter scheme (LF-RA)	16
2.6. Start/Restart strategy	18

Changes record

Release	Author(s)	Modifications
4.0	Jérôme Chanut Tim Graham	Review Update
3.6	Christian Étché	Update
≤ 3.4	Gurvan Madec	First version

Having defined the continuous equations in [chapter 1](#), we need now to choose a time discretization, a key feature of an ocean model as it exerts a strong influence on the structure of the computer code (*i.e.* on its flowchart). 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.

2.1. Time stepping environment

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

$$x^{t+\Delta t} = x^{t-\Delta t} + 2 \Delta t \text{ RHS}_x^{t-\Delta t, t, t+\Delta t} \quad (2.1)$$

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 stepping depending on the physics with which it is associated.

The choice of the time stepping 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 [equation 2.1](#)) prior to time-stepping the equation. The time stepping itself is performed once at each time step where implicit vertical diffusion is computed, *i.e.* in the `trazdf.F90` and `dynzdf.F90` modules.

2.2. Non-diffusive part — Leapfrog scheme

The time stepping used for processes other than diffusion is the well-known **LeapFrog** (LF) scheme ([Mesinger and Arakawa, 1976](#)). This scheme is widely used for advection processes in low-viscosity fluids. It is a time centred scheme, *i.e.* the RHS in [equation 2.1](#) 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 [Robert \(1966\)](#) and more comprehensively studied by [Asselin \(1972\)](#), is a kind of laplacian diffusion in time that mixes odd and even time steps:

$$x_F^t = x^t + \gamma [x_F^{t-\Delta t} - 2x^t + x^{t+\Delta t}] \quad (2.2)$$

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.e-3` (see [section 2.5](#)), causing only a weak dissipation of high frequency motions ([Farge Coulombier, 1987](#)). 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 2^nd 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.

2.3. Diffusive part — Forward or backward scheme

The leapfrog differencing scheme is unsuitable for the representation of diffusion and damping processes. For a tendency D_x , representing a diffusion term or a restoring term to a tracer climatology (when present, see [section 4.6](#)), a forward time differencing scheme is used :

$$x^{t+\Delta t} = x^{t-\Delta t} + 2 \Delta t D_x^{t-\Delta t}$$

This is diffusive in time and conditionally stable. The conditions for stability of second and fourth order horizontal diffusion schemes are (Griffies, 2004):

$$A^h < \begin{cases} \frac{e^2}{8 \Delta t} & \text{laplacian diffusion} \\ \frac{e^4}{64 \Delta t} & \text{bilaplacian diffusion} \end{cases} \quad (2.3)$$

where e is the smallest grid size in the two horizontal directions and A^h is the mixing coefficient. The linear constraint equation 2.3 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. To overcome the stability constraint, a backward (or implicit) time differencing scheme is used. This scheme is unconditionally stable but diffusive and can be written as follows:

$$x^{t+\Delta t} = x^{t-\Delta t} + 2 \Delta t \text{ RHS}_x^{t+\Delta t} \quad (2.4)$$

This scheme is rather time consuming since it requires a matrix inversion. For example, the finite difference approximation of the temperature equation is:

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

where RHS is the right hand side of the equation except for the vertical diffusion term. We rewrite equation 2.4 as:

$$-c(k+1) T^{t+1}(k+1) + d(k) T^{t+1}(k) - c(k) T^{t+1}(k-1) \equiv b(k) \quad (2.5)$$

where

$$c(k) = A_w^{vT}(k) / e_{3w}(k), \quad d(k) = e_{3t}(k) / (2\Delta t) + c_k + c_{k+1} \quad \text{and} \quad b(k) = e_{3t}(k) (T^{t-1}(k) / (2\Delta t) + \text{RHS})$$

equation 2.5 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 Richtmyer and Morton (1967)).

2.4. Surface pressure gradient

The leapfrog environment supports a centred in time computation of the surface pressure, *i.e.* evaluated at *now* time step. This refers to as the explicit free surface case in the code (`ln_dynspg_exp=.true.`). This choice however imposes a strong constraint on the time step which should be small enough to resolve the propagation of external gravity waves. As a matter of fact, one rather use in a realistic setup, a split-explicit free surface (`ln_dynspg_ts=.true.`) in which barotropic and baroclinic dynamical equations are solved separately with ad-hoc time steps. The use of the time-splitting (in combination with non-linear free surface) imposes some constraints on the design of the overall flowchart, in particular to ensure exact tracer conservation (see figure 2.1).

Compared to the former use of the filtered free surface in *NEMO* v3.6 (Roulet and Madec (2000)), the use of a split-explicit free surface is advantageous on massively parallel computers. Indeed, no global computations are anymore required by the elliptic solver which saves a substantial amount of communication time. Fast barotropic motions (such as tides) are also simulated with a better accuracy.

2.5. Modified LeapFrog – Robert Asselin filter scheme (LF-RA)

Significant changes have been introduced by Leclair and Madec (2009) 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 equation 2.2 so that Q is redistributed over several time step. In the modified LF-RA environment, these two formulations have been replaced by:

$$x^{t+\Delta t} = x^{t-\Delta t} + \Delta t \left(Q^{t-\Delta t/2} + Q^{t+\Delta t/2} \right) \quad (2.6)$$

$$x_F^t = x^t + \gamma \left(x_F^{t-\Delta t} - 2x^t + x^{t+\Delta t} \right) - \gamma \Delta t \left(Q^{t+\Delta t/2} - Q^{t-\Delta t/2} \right) \quad (2.7)$$

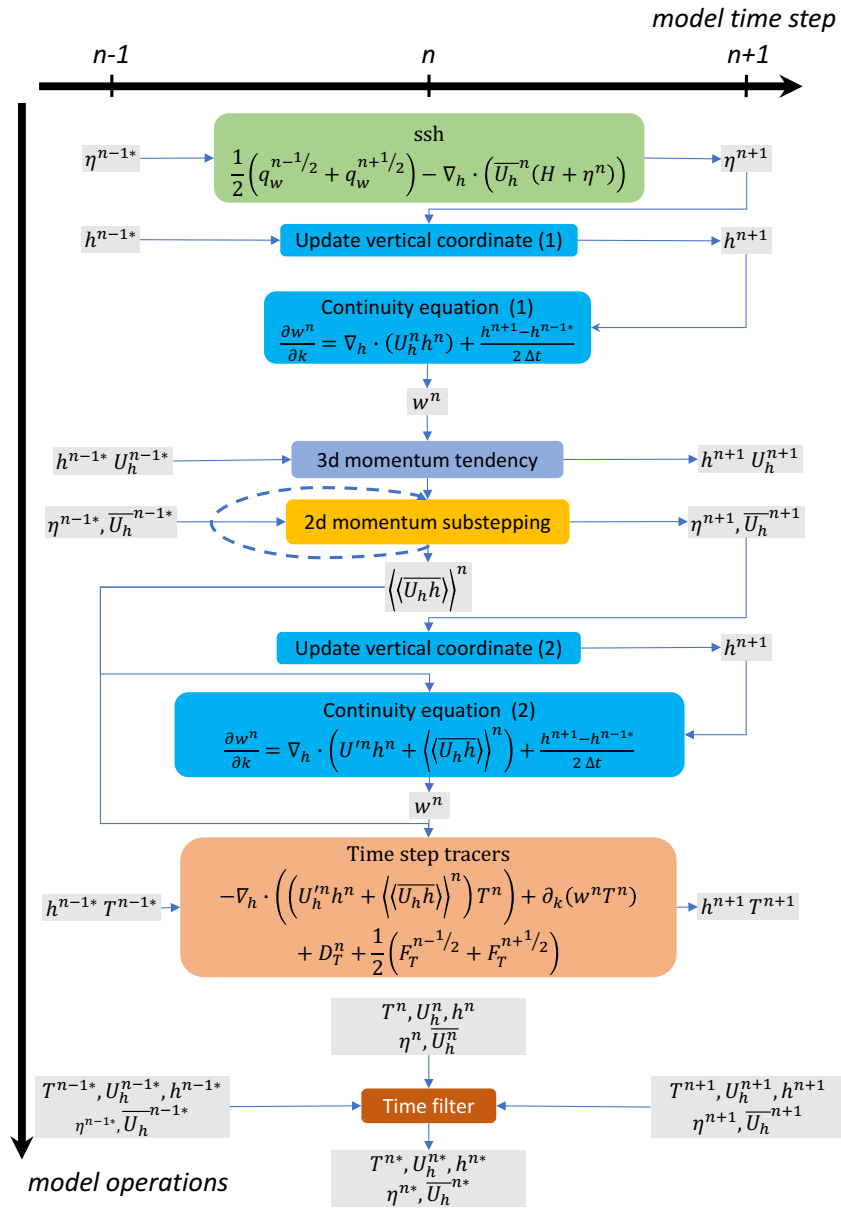


Figure 2.1.: Sketch of the leapfrog time stepping sequence in *NEMO* with split-explicit free surface. The latter combined with non-linear free surface requires the dynamical tendency being updated prior tracers tendency to ensure conservation. Note the use of time integrated fluxes issued from the barotropic loop in subsequent calculations of tracer advection and in the continuity equation. Details about the time-splitting scheme can be found in subsection 5.5.2.

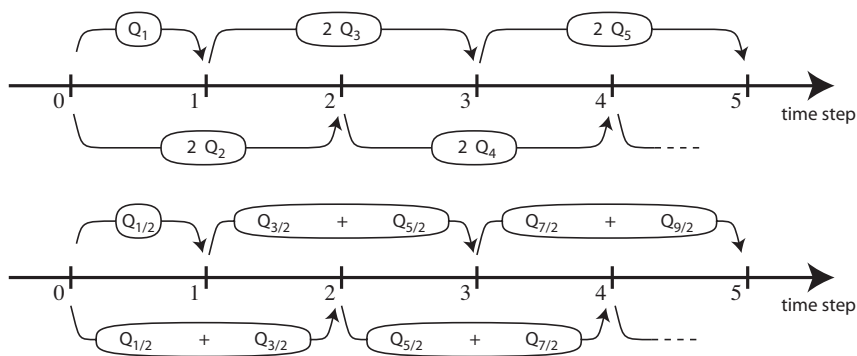


Figure 2.2.: Illustration of forcing integration methods. (top) "Traditional" formulation: the forcing is defined at the same time as the variable to which it is applied (integer value of the time step index) and it is applied over a $2\Delta t$ period. (bottom) modified formulation: the forcing is defined 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.

The change in the forcing formulation given by equation 2.6 (see figure 2.2) has a significant effect: the forcing term no longer excites the divergence of odd and even time steps (Leclair and Madec, 2009). This property improves the LF-RA scheme in two aspects. 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 explicitly removed by adding a new term in the filter (last term in equation 2.7 compared to equation 2.2). Since the filtering of the forcing was the source of non-conservation in the classical LF-RA scheme, the modified formulation becomes conservative (Leclair and Madec, 2009). Second, the LF-RA becomes a truly quasi-second order scheme. Indeed, equation 2.6 used in combination with a careful treatment of static instability (subsection 9.2.2) and of the TKE physics (subsection 9.1.6) (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, equation 2.7, allows for 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.

2.6. Start/Restart strategy

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

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

This is done simply by keeping the leapfrog environment (*i.e.* the equation 2.1 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 a leapfrog time step ($2\Delta t$).

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 the time step Δt , is also saved in the restart file. When restarting, if the time step has been changed, or one of the prognostic variables at *before* time step is missing, an Euler time stepping scheme is imposed. A forward initial step can still be enforced by the user by setting the namelist variable `nn_euler=0`. Other options to control the time integration of the model are defined through the `&namrun` (namelist 2.1) namelist variables.

```

!-----
&namrun      !  parameters of the run
!-----
nn_no       = 0      !  Assimilation cycle index
cn_exp      = "ORCA2" !  experience name
nn_it000    = 1      !  first time step
nn_itend    = 5840   !  last time step (std 5840)
nn_date0    = 010101 !  date at nit_0000 (format yyyymmdd) used if ln_rstart=F or (ln_rstart=T and
↳ nn_rstctl=0 or 1)
nn_time0    = 0      !  initial time of day in hhmm
nn_leapy    = 0      !  Leap year calendar (1) or not (0)
ln_rstart   = .false. !  start from rest (F) or from a restart file (T)
nn_euler    = 1      !  = 0 : start with forward time step if ln_rstart=T
nn_rstctl   = 0      !  restart control ==> activated only if ln_rstart=T
!           !           = 0 nn_date0 read in namelist ; nn_it000 : read in namelist
!           !           = 1 nn_date0 read in namelist ; nn_it000 : check consistency between
↳ namelist and restart
!           !           = 2 nn_date0 read in restart ; nn_it000 : check consistency between
↳ namelist and restart
cn_ocerst_in = "restart" !  suffix of ocean restart name (input)
cn_ocerst_indir = "."    !  directory from which to read input ocean restarts
cn_ocerst_out = "restart" !  suffix of ocean restart name (output)
cn_ocerst_outdir = "."   !  directory in which to write output ocean restarts
ln_iscpl    = .false.   !  cavity evolution forcing or coupling to ice sheet model
nn_istate   = 0         !  output the initial state (1) or not (0)
ln_rst_list = .false.   !  output restarts at list of times using nn_stocklist (T) or at set frequency with
↳ nn_stock (F)
nn_stock    = 0         !  used only if ln_rst_list = F: output restart frequency (modulo referenced to 1)
!           !           = 0 force to write restart files only at the end of the run
!           !           = -1 do not do any restart
nn_stocklist = 0,0,0,0,0,0,0,0,0,0 ! List of timesteps when a restart file is to be written
nn_write    = 0         !  used only if key_iomput is not defined: output frequency (modulo referenced to
↳ nn_it000)
!           !           = 0 force to write output files only at the end of the run
!           !           = -1 do not do any output file
ln_mskland  = .false.   !  mask land points in NetCDF outputs
ln_cfmeta   = .false.   !  output additional data to netCDF files required for compliance with the CF metadata
↳ standard
ln_clobber  = .true.    !  clobber (overwrite) an existing file
nn_chunksz  = 0         !  chunksize (bytes) for NetCDF file (works only with iom_nf90 routines)
ln_xios_read = .FALSE.  !  use XIOS to read restart file (only for a single file restart)
nn_wxios   = 0         !  use XIOS to write restart file 0 - no, 1 - single file output, 2 - multiple file output
/

```

namelist 2.1.: &namrun

Table of contents

3.1. Fundamentals of the discretisation	21
3.1.1. Arrangement of variables	21
3.1.2. Discrete operators	22
3.1.3. Numerical indexing	23
3.2. Spatial domain configuration	24
3.2.1. Domain size	24
3.2.2. Horizontal grid mesh (<i>domhgr.F90</i>)	25
3.2.3. Vertical grid (<i>domzgr.F90</i>)	25
3.2.4. Closed seas	27
3.2.5. Output grid files	27
3.3. Initial state (<i>istate.F90</i> and <i>dtatsd.F90</i>)	28

Changes record

Release	Author(s)	Modifications
4.0	<i>Simon Müller & Andrew Coward</i>	<i>Compatibility changes: many options moved to external domain configuration tools (see appendix E).</i>
3.6	<i>Simona Flavoni and Tim Graham Rachid Benshila, Christian Éthé, Pierre Mathiot and Gurban Madec</i>	<i>Updates</i>
≤ 3.4	<i>Gurban Madec and Sébastien Masson</i>	<i>First version</i>

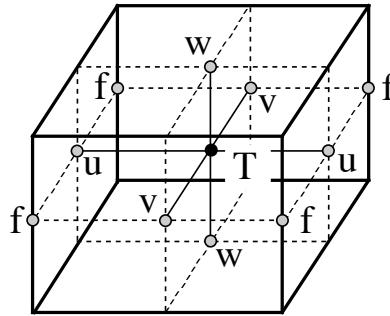


Figure 3.1.: Arrangement of variables in the unit cell of space domain. t indicates scalar points where temperature, salinity, density, pressure and horizontal divergence are defined. (u, v, w) indicates vector points, and f indicates vorticity points where both relative and planetary vorticities are defined.

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	$i + 1/2$	$j + 1/2$	$k + 1/2$

Table 3.1.: Location of grid-points as a function of integer or integer and a half value of the column, line or level. This indexing is only used for the writing of the semi-discrete equations. In the code, the indexing uses integer values only and is positive downwards in the vertical with $k = 1$ at the surface. (see subsection 3.1.3)

Having defined the continuous equations in [chapter 1](#) and chosen a time discretisation [chapter 2](#), we need to choose a grid for spatial discretisation and related numerical algorithms. In the present chapter, we provide a general description of the staggered grid used in *NEMO*, and other relevant information about the DOM (DOMain) source code modules.

3.1. Fundamentals of the discretisation

3.1.1. Arrangement of variables

The numerical techniques used to solve the Primitive Equations in this model are based on the traditional, centred second-order finite difference approximation. Special attention has been given to the homogeneity of the solution in the three spatial 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 ([figure 3.1](#)). This is the generalisation to three dimensions of the well-known “C” grid in Arakawa’s classification ([Mesinger and Arakawa, 1976](#)). The relative and planetary vorticity, ζ and f , are defined in the centre of each vertical edge and the barotropic stream function ψ is defined at horizontal points overlying the ζ and f -points.

The ocean mesh (*i.e.* the position of all the scalar and vector points) is defined by the transformation that gives (λ, φ, z) as a function of (i, j, k) . The grid-points are located at integer or integer and a half value of (i, j, k) as indicated on [table 3.1](#). 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 [equation 1.7](#). As a result, the mesh on which partial derivatives $\frac{\partial}{\partial \lambda}$, $\frac{\partial}{\partial \varphi}$ and $\frac{\partial}{\partial z}$ are evaluated is a uniform mesh with a grid size of unity. Discrete partial derivatives are formulated by the traditional, centred second order finite difference approximation while the scale factors are chosen equal to their local analytical value. An important point here is that the partial derivative of the scale factors must be evaluated by centred finite difference approximation, not from their analytical expression. This preserves the symmetry of the discrete set of equations and therefore satisfies many of the continuous properties (see [appendix C](#)). A similar, related remark can be made about the domain size: when needed, an area, volume, or the total ocean depth must be evaluated as the product or sum of the relevant scale factors (see [equation 3.1](#) in the next section).

Note that the definition of the scale factors (*i.e.* as the analytical first derivative of the transformation that results in (λ, φ, z) as a function of (i, j, k)) is specific to the *NEMO* model ([Marti et al., 1992](#)). As an example, a scale factor in the i direction 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:

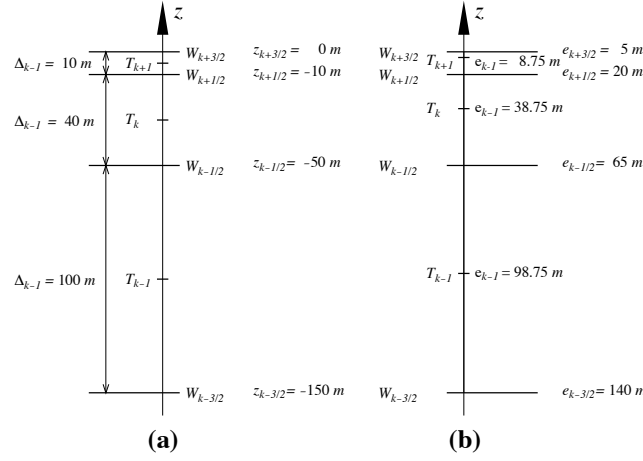


Figure 3.2.: Comparison of (a) traditional definitions of grid-point position and grid-size in the vertical, and (b) analytically derived grid-point position and scale factors. For both grids here, the same w -point depth has been chosen but in (a) the t -points are set half way between w -points while in (b) they are defined from an analytical function: $z(k) = 5(k - 1/2)^3 - 45(k - 1/2)^2 + 140(k - 1/2) - 150$. Note the resulting difference between the value of the grid-size Δ_k and those of the scale factor e_k .

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) (Tréguier et al., 1996). An example of the effect of such a choice is shown in figure 3.2.

3.1.2. Discrete operators

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

$$\begin{aligned}\delta_i[q] &= q(i + 1/2) - q(i - 1/2) \\ \bar{q}^i &= \{q(i + 1/2) + q(i - 1/2)\}/2\end{aligned}$$

Similar operators are defined with respect to $i + 1/2$, j , $j + 1/2$, k , and $k + 1/2$. Following equation 1.8a and equation 1.8d, the gradient of a variable q defined at a t -point has its three components defined at u -, v - and w -points while its Laplacian is defined at the t -point. These operators have the following discrete forms in the curvilinear s -coordinates system:

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

Following equation 1.8c and equation 1.8b, a vector $\mathbf{A} = (a_1, a_2, a_3)$ defined at vector points (u, v, w) has its three curl components defined at vw -, uw , and f -points, and its divergence defined at t -points:

$$\begin{aligned}\nabla \times \mathbf{A} &\equiv \frac{1}{e_{2v} e_{3vw}} \left[\delta_{j+1/2}(e_{3w} a_3) - \delta_{k+1/2}(e_{2v} a_2) \right] \mathbf{i} \\ &\quad + \frac{1}{e_{2u} e_{3uw}} \left[\delta_{k+1/2}(e_{1u} a_1) - \delta_{i+1/2}(e_{3w} a_3) \right] \mathbf{j} \\ &\quad + \frac{1}{e_{1f} e_{2f}} \left[\delta_{i+1/2}(e_{2v} a_2) - \delta_{j+1/2}(e_{1u} a_1) \right] \mathbf{k} \\ \nabla \cdot \mathbf{A} &\equiv \frac{1}{e_{1t} e_{2t} e_{3t}} \left[\delta_i(e_{2u} e_{3u} a_1) + \delta_j(e_{1v} e_{3v} a_2) \right] + \frac{1}{e_{3t}} \delta_k(a_3)\end{aligned}$$

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

$$\bar{q} = \frac{1}{H} \int_{k^b}^{k^o} q e_{3q} dk \equiv \frac{1}{H_q} \sum_k q e_{3q} \quad (3.1)$$

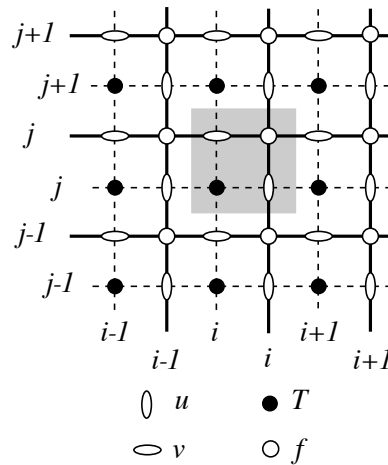


Figure 3.3.: Horizontal integer indexing used in the FORTRAN code. The dashed area indicates the cell in which variables contained in arrays have the same i - and j -indices

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 \sum_k 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:

$$\nabla \times \nabla q = 0 \quad (3.2)$$

$$\nabla \cdot (\nabla \times A) = 0 \quad (3.3)$$

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 a value of zero inside continental areas. It can be shown that the differencing operators (δ_i, δ_j and δ_k) are skew-symmetric linear operators, and further that the averaging operators ($\bar{\cdot}^i, \bar{\cdot}^j$ and $\bar{\cdot}^k$) are symmetric linear operators, *i.e.*

$$\sum_i a_i \delta_i [b] \equiv - \sum_i \delta_{i+1/2} [a] b_{i+1/2} \quad (3.4)$$

$$\sum_i a_i \bar{b}^i \equiv \sum_i \bar{a}^{i+1/2} b_{i+1/2} \quad (3.5)$$

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

3.1.3. Numerical indexing

The array representation used in the FORTRAN code requires an integer indexing. However, the analytical definition of the mesh (see [subsection 3.1.1](#)) is associated with the use of integer values for t -points only while all the other points involve integer and a half values. Therefore, a specific integer indexing has been defined for points other than t -points (*i.e.* velocity and vorticity grid-points). Furthermore, the direction of the vertical indexing has been reversed and the surface level set at $k = 1$.

Horizontal indexing

The indexing in the horizontal plane has been chosen as shown in [figure 3.3](#). For an increasing i index (j index), the t -point and the eastward u -point (northward v -point) have the same index (see the dashed area in [figure 3.3](#)). A t -point and its nearest north-east f -point have the same i - and j -indices.

Vertical indexing

In the vertical, the chosen indexing requires special attention since the direction of the k -axis in the FORTRAN code is the reverse of that used in the semi-discrete equations and given in [subsection 3.1.1](#). The sea surface corresponds to the w -level $k = 1$, which is the same index as the t -level just below ([figure 3.4](#)). The last w -level ($k = jpk$) either corresponds

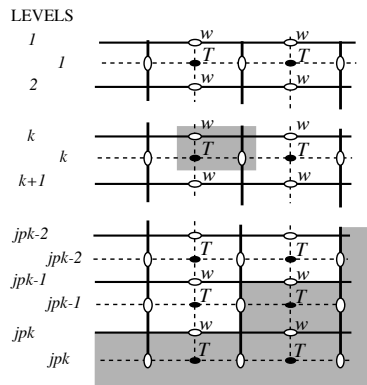


Figure 3.4.: Vertical integer indexing used in the FORTRAN code. Note that the k -axis is oriented downward. The dashed area indicates the cell in which variables contained in arrays have a common k -index.

to or is below the ocean floor while the last t -level is always outside the ocean domain (figure 3.4). Note that a w -point and the directly underlying t -point have a common k index (*i.e.* t -points and their nearest w -point neighbour in negative index direction), in contrast to the indexing on the horizontal plane where the t -point has the same index as the nearest velocity points in the positive direction of the respective horizontal axis index (compare the dashed area in figure 3.3 and figure 3.4). Since the scale factors are chosen to be strictly positive, a *minus sign* is included in the FORTRAN implementations of *all the vertical derivatives* of the discrete equations given in this manual in order to accommodate the opposing vertical index directions in implementation and documentation.

3.2. Spatial domain configuration

Two typical methods are available to specify the spatial domain configuration; they can be selected using parameter `ln_read_cfg` parameter in namelist `&namcfg` (namelist 15.1).

If `ln_read_cfg` is set to `.true.`, the domain-specific parameters and fields are read from a NetCDF input file, whose name (without its `.nc` suffix) can be specified as the value of the `cn_domcfg` parameter in namelist `&namcfg` (namelist 15.1).

If `ln_read_cfg` is set to `.false.`, the domain-specific parameters and fields can be provided (*e.g.* analytically computed) by subroutines `usrdef_hgr.F90` and `usrdef_zgr.F90`. These subroutines can be supplied in the `MY_SRC` directory of the configuration, and default versions that configure the spatial domain for the GYRE reference configuration are present in the `./src/OCE/USR` directory.

In version 4.0 there are no longer any options for reading complex bathymetries and performing a vertical discretisation at run-time. Whilst it is occasionally convenient to have a common bathymetry file and, for example, to run similar models with and without partial bottom boxes and/or sigma-coordinates, supporting such choices leads to overly complex code. Worse still is the difficulty of ensuring the model configurations intended to be identical are indeed so when the model domain itself can be altered by runtime selections. The code previously used to perform vertical discretisation has been incorporated into an external tool (`./tools/DOMAINcfg`) which is briefly described in appendix E.

The next subsections summarise the parameter and fields related to the configuration of the whole model domain. These represent the minimum information that must be provided either via the `cn_domcfg` file or set by code inserted into user-supplied versions of the `usrdef_*` subroutines. The requirements are presented in three sections: the domain size (subsection 3.2.1), the horizontal mesh (subsection 3.2.2), and the vertical grid (subsection 3.2.3).

3.2.1. Domain size

The total size of the computational domain is set by the parameters `jpiglo`, `jpjglo` and `jkpglo` for the i , j and k directions, respectively. Note, that the variables `jpj` and `jkj` refer to the size of each processor subdomain when the code is run in parallel using domain decomposition (`key_mpp_mpi` defined, see section 7.3).

The name of the configuration is set through parameter `cn_cfg`, and the nominal resolution through parameter `nn_cfg` (unless in the input file both of variables `ORCA` and `ORCA_index` are present, in which case `cn_cfg` and `nn_cfg` are set from these values accordingly).

The global lateral boundary condition type is selected from 8 options using parameter `jprio`. See section 7.2 for details on the available options and the corresponding values for `jprio`.


```

!-----
&namdom      !   time and space domain
!-----
ln_linssh    = .false.  ! =T linear free surface ==> model level are fixed in time
rn_isfhmin   = 1.00    ! treshold [m] to discriminate grounding ice from floating ice
!
rn_rdt       = 5400.    ! time step for the dynamics and tracer
rn_atfp      = 0.1     ! asselin time filter parameter
!
ln_crs       = .false.  ! Logical switch for coarsening module      (T => fill namcrs)
!
ln_meshmask  = .false.  ! =T create a mesh file
/

```

namelist 3.1.: &namdom

3.2.2. Horizontal grid mesh (*domhgr.F90*)

Required fields

The explicit specification of a range of mesh-related fields are required for the definition of a configuration. These include:

```

int    jpiglo, jpnglo, jpkglo    /* global domain sizes                */
int    jperio                    /* lateral global domain b.c.         */
double glamt, glamu, glamv, glamf /* geographic longitude (t,u,v and f points respectively) */
double gphit, gphiu, gp hiv, gp hif /* geographic latitude                */
double elt, elu, elv, elf       /* horizontal scale factors            */
double e2t, e2u, e2v, e2f       /* horizontal scale factors            */

```

The values of the geographic longitude and latitude arrays at indices i, j correspond to the analytical expressions of the longitude λ and latitude φ as a function of (i, j) , evaluated at the values as specified in [table 3.1](#) for the respective grid-point position. The calculation of the values of the horizontal scale factor arrays in general additionally involves partial derivatives of λ and φ with respect to i and j , evaluated for the same arguments as λ and φ .

Optional fields

```

/* Optional: */
int    ORCA, ORCA_index /* configuration name, configuration resolution */
double ele2u, ele2v     /* U and V surfaces (if grid size reduction in some straits) */
double ff_f, ff_t      /* Coriolis parameter (if not on the sphere) */

```

NEMO can support the local reduction of key strait widths by altering individual values of $e2u$ or $e1v$ at the appropriate locations. This is particularly useful for locations such as Gibraltar or Indonesian Throughflow pinch-points (see [section 14.1](#) for illustrated examples). The key is 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 cells. Doing otherwise can lead to numerical instability issues. In normal operation the surface areas are computed from $e1u * e2u$ and $e1v * e2v$ but in cases where a gridsize reduction is required, the unaltered surface areas at u and v grid points ($e1e2u$ and $e1e2v$, respectively) must be read or pre-computed in *usrdef_hgr.F90*. If these arrays are present in the *cn_domcfg* file they are read and the internal computation is suppressed. Versions of *usrdef_hgr.F90* which set their own values of $e1e2u$ and $e1e2v$ should set the surface-area computation flag: *ie1e2u_v* to a non-zero value to suppress their re-computation.

Similar logic applies to the other optional fields: *ff_f* and *ff_t* which can be used to provide the Coriolis parameter at F- and T-points respectively if the mesh is not on a sphere. If present these fields will be read and used and the normal calculation ($2 * \Omega * \sin(\varphi)$) suppressed. Versions of *usrdef_hgr.F90* which set their own values of *ff_f* and *ff_t* should set the Coriolis computation flag: *iff* to a non-zero value to suppress their re-computation.

Note that longitudes, latitudes, and scale factors at w points are exactly equal to those of t points, thus no specific arrays are defined at w points.

3.2.3. Vertical grid (*domzgr.F90*)

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 (*jkp*);
3. the analytical transformation $z(i, j, k)$ and the vertical scale factors (derivatives of the transformation); and

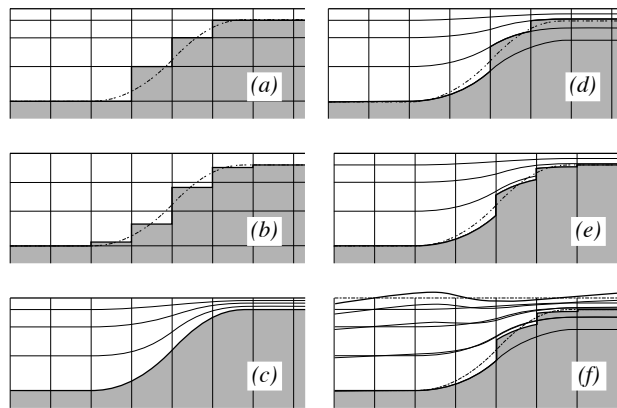


Figure 3.5.: The ocean bottom as seen by the model: (a) z -coordinate with full step, (b) z -coordinate with partial step, (c) s -coordinate: terrain following representation, (d) hybrid $s - z$ coordinate, (e) hybrid $s - z$ coordinate with partial step, and (f) same as (e) but in the non-linear free surface (`ln_linssh=.false.`). Note that the non-linear free surface can be used with any of the 5 coordinates (a) to (e).

4. the masking system, *i.e.* the number of wet model levels at each (i, j) location of the horizontal grid.

The choice of a vertical coordinate is made when setting up the configuration; it is not intended to be an option which can be changed in the middle of an experiment. The one exception to this statement being the choice of linear or non-linear free surface. In v4.0 the linear free surface option is implemented as a special case of the non-linear free surface. This is computationally wasteful since it uses the structures for time-varying 3D metrics for fields that (in the linear free surface case) are fixed. However, the linear free-surface is rarely used and implementing it this way means a single configuration file can support both options.

By default a non-linear free surface is used (`ln_linssh` set to `=.false.` in `&namdom (namelist 3.1)`): the coordinate follow the time-variation of the free surface so that the transformation is time dependent: $z(i, j, k, t)$ (*e.g.* item 3.5f). When a linear free surface is assumed (`ln_linssh` set to `=.true.` in `&namdom (namelist 3.1)`), the vertical coordinates are fixed in time, but the seawater can move up and down across the z_0 surface (in other words, the top of the ocean is not a rigid lid).

Note that settings: `ln_zco`, `ln_zps`, `ln_sco` and `ln_isfcav` mentioned in the following sections appear to be namelist options but they are no longer truly namelist options for *NEMO*. Their value is written to and read from the domain configuration file and they should be treated as fixed parameters for a particular configuration. They are namelist options for the `DOMAINcfg` tool that can be used to build the configuration file and serve both to provide a record of the choices made whilst building the configuration and to trigger appropriate code blocks within *NEMO*. These values should not be altered in the `cn_domcfg` file.

The decision on these choices must be made when the `cn_domcfg` file is constructed. Three main choices are offered (item 3.5a-c):

- z -coordinate with full step bathymetry (`ln_zco=.true.`),
- z -coordinate with partial step (zps) bathymetry (`ln_zps=.true.`),
- Generalized, s -coordinate (`ln_sco=.true.`).

Additionally, hybrid combinations of the three main coordinates are available: $s - z$ or $s - zps$ coordinate (item 3.5d and item 3.5e).

A further choice related to vertical coordinate concerns the presence (or not) of ocean cavities beneath ice shelves within the model domain. A setting of `ln_isfcav` as `.true.` indicates that the domain contains ocean cavities, otherwise the top, wet layer of the ocean will always be at the ocean surface. This option is currently only available for z - or zps -coordinates. In the latter case, partial steps are also applied at the ocean/ice shelf interface.

Within the model, the arrays describing the grid point depths and vertical scale factors are three set of three dimensional arrays (i, j, k) defined at *before*, *now* and *after* time step. The time at which they are defined is indicated by a suffix: `_b`, `_n`, or `_a`, respectively. They are updated at each model time step. The initial fixed reference coordinate system is held in variable names with a `_0` suffix. When the linear free surface option is used (`ln_linssh=.true.`), *before*, *now* and *after* arrays are initially set to their reference counterpart and remain fixed.

Required fields

The explicit specification of a range of fields related to the vertical grid are required for the definition of a configuration. These include:

```

int    ln_zco, ln_zps, ln_sco          /* flags for z-coord, z-coord with partial steps and s-coord */
int    ln_isfcav                      /* flag for ice shelf cavities */
double e3t_ld, e3w_ld                 /* reference vertical scale factors at T and W points */
double e3t_0, e3u_0, e3v_0, e3f_0, e3w_0 /* vertical scale factors 3D coordinate at T,U,V,F and W points */
double e3uw_0, e3vw_0                /* vertical scale factors 3D coordinate at UW and VW points */
int    bottom_level, top_level        /* last wet T-points, 1st wet T-points (for ice shelf cavities) */
float  bathy_metry                    /* For reference:
float  bathy_metry                    /* bathymetry used in setting top and bottom levels */

```

This set of vertical metrics is sufficient to describe the initial depth and thickness of every gridcell in the model regardless of the choice of vertical coordinate. With constant z-levels, e3 metrics will be uniform across each horizontal level. In the partial step case each e3 at the `bottom_level` (and, possibly, `top_level` if ice cavities are present) may vary from its horizontal neighbours. And, in s-coordinates, variations can occur throughout the water column. With the non-linear free-surface, all the coordinates behave more like the s-coordinate in that variations occur throughout the water column with displacements related to the sea surface height. These variations are typically much smaller than those arising from bottom fitted coordinates. The values for vertical metrics supplied in the domain configuration file can be considered as those arising from a flat sea surface with zero elevation.

The `bottom_level` and `top_level` 2D arrays define the `bottom_level` and top wet levels in each grid column. Without ice cavities, `top_level` is essentially a land mask (0 on land; 1 everywhere else). With ice cavities, `top_level` determines the first wet point below the overlying ice shelf.

Level bathymetry and mask

From `top_level` and `bottom_level` fields, the mask fields are defined as follows:

$$tmask(i, j, k) = \begin{cases} 0 & \text{if } k < top_level(i, j) \\ 1 & \text{if } bottom_level(i, j) \leq k \leq top_level(i, j) \\ 0 & \text{if } k > bottom_level(i, j) \end{cases}$$

$$umask(i, j, k) = tmask(i, j, k) * tmask(i + 1, j, k)$$

$$vmask(i, j, k) = tmask(i, j, k) * tmask(i, j + 1, k)$$

$$fmask(i, j, k) = tmask(i, j, k) * tmask(i + 1, j, k) * tmask(i, j, k) * tmask(i + 1, j, k)$$

$$wmask(i, j, k) = tmask(i, j, k) * tmask(i, j, k - 1)$$

with $wmask(i, j, 1) = tmask(i, j, 1)$

Note that, without ice shelves cavities, masks at t - and w -points are identical with the numerical indexing used (subsection 3.1.3). Nevertheless, $wmask$ are required with ocean cavities to deal with the top boundary (ice shelf/ocean interface) exactly in the same way as for the bottom boundary.

3.2.4. Closed seas

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 section 14.2) and to optionally decide on the fate of any freshwater imbalance over the area. The options are explained in section 14.2 but it should be noted here that a successful use of these options requires appropriate mask fields to be present in the domain configuration file. Among the possibilities are:

```

int closea_mask /* non-zero values in closed sea areas for optional masking */
int closea_mask_rnf /* non-zero values in closed sea areas with runoff locations (precip only) */
int closea_mask_emp /* non-zero values in closed sea areas with runoff locations (total emp) */

```

3.2.5. Output grid files

Most of the arrays relating to a particular ocean model configuration discussed in this chapter (grid-point position, scale factors) can be saved in a file if namelist parameter `ln_write_cfg` (namelist `&namcfg` (namelist 15.1)) is set to `.true.`; the output filename is set through parameter `cn_domcfg_out`. This is only really useful if the fields are computed in subroutines `usrdef_hgr.F90` or `usrdef_zgr.F90` and checking or confirmation is required.

Alternatively, all the arrays relating to a particular ocean model configuration (grid-point position, scale factors, depths and masks) can be saved in a file called `mesh_mask` if namelist parameter `ln_meshmask` (namelist `&namdom` (namelist 3.1)) is set to `.true.`. This file contains additional fields that can be useful for post-processing applications.

```

!-----
&namtsd      !   Temperature & Salinity Data  (init/dmp)           (default: OFF)
!-----
!
!   ! =T  read T-S fields for:
ln_tsd_init = .false.      ! ocean initialisation
ln_tsd_dmp  = .false.      ! T-S restoring  (see namtra_dmp)

cn_dir      = './'        ! root directory for the T-S data location

↪ !-----!-----!-----!-----!-----!-----!-----!-----!-----!-----!
!   ! file name           ! frequency (hours) ! variable ! time interp.! clim ! 'yearly' / !
↪ weights filename ! rotation ! land/sea mask !
!   !                   ! (if <0 months) ! name     ! (logical) ! (T/F) ! 'monthly' !
↪ ! pairing ! filename !
sn_tem = 'data_lm_potential_temperature_nomask', -1.    , 'votemper', .true.  , .true.  , 'yearly' ,
↪ ''      , ''      , ''      , ''      , ''      , ''      , ''      , ''      , ''      ,
sn_sal = 'data_lm_salinity_nomask'                , -1.    , 'vosaline', .true.  , .true.  , 'yearly' ,
↪ ''      , ''      , ''      , ''      , ''      , ''      , ''      , ''      , ''      ,
/

```

namelist 3.2.: &namtsd

3.3. Initial state (*istate.F90* and *dtatsd.F90*)

Basic initial state options are defined in &namtsd (namelist 3.2). By default, the ocean starts from rest (the velocity field is set to zero) and the initialization of temperature and salinity fields is controlled through the `ln_tsd_init` namelist parameter.

ln_tsd_init=.true. Use T and S input files that can be given on the model grid itself or on their native input data grids. In the latter case, the data will be interpolated on-the-fly both in the horizontal and the vertical to the model grid (see subsection 6.2.2). The information relating to the input files are specified in the `sn_tem` and `sn_sal` structures. The computation is done in the *dtatsd.F90* module.

ln_tsd_init=.false. Initial values for T and S are set via a user supplied `usr_def_istate` routine contained in *userdef_istate.F90*. The default version sets horizontally uniform T and profiles as used in the GYRE configuration (see section 15.4).

Table of contents

4.1. Tracer advection (<i>traadv.F90</i>)	30
4.1.1. CEN: Centred scheme (<i>ln_traadv_cen</i>)	32
4.1.2. FCT: Flux Corrected Transport scheme (<i>ln_traadv_fct</i>)	32
4.1.3. MUSCL: Monotone Upstream Scheme for Conservative Laws (<i>ln_traadv_mus</i>)	33
4.1.4. UBS a.k.a. UP3: Upstream-Biased Scheme (<i>ln_traadv_ubs</i>)	33
4.1.5. QCK: QuiCKest scheme (<i>ln_traadv_qck</i>)	34
4.2. Tracer lateral diffusion (<i>traldf.F90</i>)	34
4.2.1. Type of operator (<i>ln_traldf_{OFF,lap,blp}</i>)	34
4.2.2. Action direction (<i>ln_traldf_{lev,hor,iso,triad}</i>)	35
4.2.3. Iso-level (bi-)laplacian operator (<i>ln_traldf_iso</i>)	35
4.2.4. Standard and triad (bi-)laplacian operator	36
4.3. Tracer vertical diffusion (<i>trazdf.F90</i>)	36
4.4. External forcing	37
4.4.1. Surface boundary condition (<i>trasbc.F90</i>)	37
4.4.2. Solar radiation penetration (<i>traqsr.F90</i>)	38
4.4.3. Bottom boundary condition (<i>trabbc.F90</i>) - <i>ln_trabbc</i>	39
4.5. Bottom boundary layer (<i>trabbl.F90</i> - <i>ln_trabbl</i>)	39
4.5.1. Diffusive bottom boundary layer (<i>nn_bbl_ldf=1</i>)	41
4.5.2. Advective bottom boundary layer (<i>nn_bbl_adv=1, 2</i>)	41
4.6. Tracer damping (<i>tradmp.F90</i>)	42
4.7. Tracer time evolution (<i>tranxt.F90</i>)	43
4.8. Equation of state (<i>eosbn2.F90</i>)	43
4.8.1. Equation of seawater (<i>ln_{teos10,eos80,seos}</i>)	43
4.8.2. Brunt-Väisälä frequency	45
4.8.3. Freezing point of seawater	45
4.9. Horizontal derivative in <i>zps</i> -coordinate (<i>zpsjde.F90</i>)	45

Changes record

Release	Author(s)	Modifications
4.0	Christian Éthé	Review
3.6	Gurvan Madec	Update
≤ 3.4	Gurvan Madec and Sébastien Masson	First version

```

!-----
&namtra_adv ! advection scheme for tracer (default: NO selection)
!-----
ln_traadv_OFF = .false. ! No tracer advection
ln_traadv_cen = .false. ! 2nd order centered scheme
  nn_cen_h = 4 ! =2/4, horizontal 2nd order CEN / 4th order CEN
  nn_cen_v = 4 ! =2/4, vertical 2nd order CEN / 4th order COMPACT
ln_traadv_fct = .false. ! FCT scheme
  nn_fct_h = 2 ! =2/4, horizontal 2nd / 4th order
  nn_fct_v = 2 ! =2/4, vertical 2nd / COMPACT 4th order
ln_traadv_mus = .false. ! MUSCL scheme
  ln_mus_ups = .false. ! use upstream scheme near river mouths
ln_traadv_ubs = .false. ! UBS scheme
  nn_ubs_v = 2 ! =2, vertical 2nd order FCT / COMPACT 4th order
ln_traadv_qck = .false. ! QUICKEST scheme
/

```

namelist 4.1.: &namtra_adv

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

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

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

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

In the present chapter we also describe the diagnostic equations used to compute the sea-water properties (density, Brunt-Väisälä 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. 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 equivalent code can be found in the *traTTT* or *traTTT_xxx* module, in the `./src/OCE/TRA` directory.

The user has the option of extracting each tendency term on the RHS of the tracer equation for output (`ln_tra_trd` or `ln_tra_mxl=.true.`), as described in [chapter 10](#).

4.1. Tracer advection (traadv.F90)

When considered (*i.e.* when `ln_traadv_OFF` is not set to `.true.`), the advection tendency of a tracer is expressed in flux form, *i.e.* as the divergence of the advective fluxes. Its discrete expression is given by:

$$\text{ADV}_\tau = -\frac{1}{b_t} \left(\delta_i [e_{2u} e_{3u} u \tau_u] + \delta_j [e_{1v} e_{3v} v \tau_v] \right) - \frac{1}{e_{3t}} \delta_k [w \tau_w] \quad (4.1)$$

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 [equation 4.1](#) implicitly requires the use of the continuity equation. Indeed, it is obtained by using the following equality: $\nabla \cdot (UT) = U \cdot \nabla T$ which results from the use of the continuity equation, $\partial_t e_3 + e_3 \nabla \cdot U = 0$ (which reduces to $\nabla \cdot U = 0$ in linear free surface, *i.e.* `ln_linssh=.true.`). 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 setting $\tau = 1$ in ([equation 4.1](#)) we recover the discrete form of the continuity equation which is used to calculate the vertical velocity.

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

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:

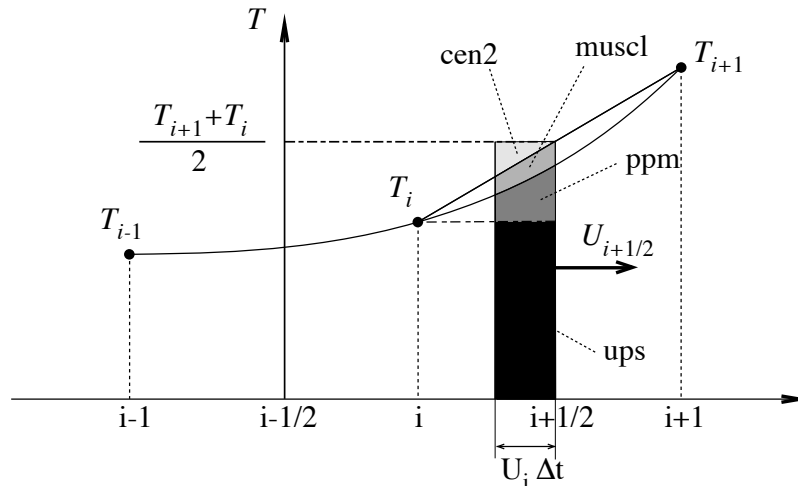


Figure 4.1.: Schematic representation of some ways used to evaluate the tracer value at u -point and the amount of tracer exchanged between two neighbouring grid points. Upstream biased scheme (ups): the upstream value is used and the black area is exchanged. Piecewise parabolic method (ppm): a parabolic 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.

linear free surface (`ln_linssh=.true.`) 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 (`ln_linssh=.false.`) 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 non-linear free surface case, but *not* in the linear free surface case. Nevertheless, in the latter case, it is achieved to a good approximation since the non-conservative term is the product of the time derivative of the tracer and the free surface height, two quantities that are not correlated (Roulet and Madec, 2000; Griffies et al., 2001; Campin et al., 2004).

The velocity field that appears in (equation 4.1) is the centred (*now*) effective ocean velocity, *i.e.* the eulerian velocity (see chapter 5) plus the eddy induced velocity (*eiv*) and/or the mixed layer eddy induced velocity (*eiv*) when those parameterisations are used (see chapter 8).

Several tracer advection scheme are proposed, namely a 2^{nd} or 4^{th} order **CEN**tered schemes (CEN), a 2^{nd} or 4^{th} order **Flux Corrected Transport** scheme (FCT), a **Monotone Upstream Scheme for Conservative Laws** scheme (MUSCL), a 3^{rd} **Upstream Biased Scheme** (UBS, also often called UP3), and a **Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Terms** scheme (QUICKEST). The choice is made in the `&namtra_adv` (namelist 4.1) namelist, by setting to `.true.` 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. By default (*i.e.* in the reference namelist, `namelist_ref`), all the logicals are set to `.false..` If the user does not select an advection scheme in the configuration namelist (`namelist_cfg`), the tracers will *not* be advected!

Details of the advection schemes are given below. The choosing 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. In particular, we note that

1. CEN and FCT schemes require an explicit diffusion operator while the other schemes are diffusive enough so that they do not necessarily need additional diffusion;
2. CEN and UBS are not *positive* schemes*, 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.

*negative values can appear in an initially strictly positive tracer field which is advected

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.

4.1.1. CEN: Centred scheme (`ln_traadv_cen`)

The CENtered advection scheme (CEN) is used when `ln_traadv_cen=.true.`. Its order (2^{nd} or 4^{th}) can be chosen independently on horizontal (iso-level) and vertical direction by setting `nn_cen_h` and `nn_cen_v` to 2 or 4. CEN implementation can be found in the `traadv_cen.F90` module.

In the 2^{nd} order centred formulation (CEN2), the tracer at velocity points is evaluated as the mean of the two neighbouring T -point values. For example, in the i -direction :

$$\tau_u^{cen2} = \overline{T}^{i+1/2} \quad (4.2)$$

CEN2 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 (equation 4.2) is the *now* tracer value.

Note that using the CEN2, the overall tracer advection is of second order accuracy since both (equation 4.1) and (equation 4.2) have this order of accuracy.

In the 4^{th} order formulation (CEN4), tracer values are evaluated at u- and v-points as a 4^{th} order interpolation, and thus depend on the four neighbouring T -points. For example, in the i -direction:

$$\tau_u^{cen4} = T - \frac{1}{6} \delta_i \left[\delta_{i+1/2}[T] \right]^{i+1/2} \quad (4.3)$$

In the vertical direction (`nn_cen_v=4`), a 4^{th} COMPACT interpolation has been preferred (Demange, 2014). In the COMPACT scheme, both the field and its derivative are interpolated, which leads, after a matrix inversion, spectral characteristics similar to schemes of higher order (Lele, 1992).

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 equation 4.1 is kept at 2^{nd} order. The expression *4th 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 the continuity equation, and the momentum advection and pressure 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. As in CEN2 case, the time-stepping is performed using a leapfrog scheme in conjunction with an Asselin time-filter, so T in (equation 4.3) 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.

4.1.2. FCT: Flux Corrected Transport scheme (`ln_traadv_fct`)

The Flux Corrected Transport schemes (FCT) is used when `ln_traadv_fct=.true.`. Its order (2^{nd} or 4^{th}) can be chosen independently on horizontal (iso-level) and vertical direction by setting `nn_fct_h` and `nn_fct_v` to 2 or 4. FCT implementation can be found in the `traadv_fct.F90` module.

In FCT formulation, the tracer at velocity points is evaluated using a combination of an upstream and a centred scheme. For example, in the i -direction :

$$\tau_u^{ups} = \begin{cases} T_{i+1} & \text{if } u_{i+1/2} < 0 \\ T_i & \text{if } u_{i+1/2} \geq 0 \end{cases} \quad (4.4)$$

$$\tau_u^{fct} = \tau_u^{ups} + c_u (\tau_u^{cen} - \tau_u^{ups})$$

where c_u is a flux limiter function taking values between 0 and 1. The FCT order is the one of the centred scheme used (*i.e.* it depends on the setting of `nn_fct_h` and `nn_fct_v`). There exist many ways to define c_u , each corresponding to a different FCT scheme. The one chosen in *NEMO* is described in Zalesak (1979). c_u only departs from 1 when the advective term produces a local extremum in the tracer field. The resulting scheme is quite expensive but *positive*. It can be used on both active and passive tracers. A comparison of FCT-2 with MUSCL and a MPDATA scheme can be found in Lévy et al. (2001).

For stability reasons (see [chapter 2](#)), τ_u^{cen} is evaluated in ([equation 4.4](#)) 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.

4.1.3. MUSCL: Monotone Upstream Scheme for Conservative Laws (`ln_traadv_mus`)

The Monotone Upstream Scheme for Conservative Laws (MUSCL) is used when `ln_traadv_mus=.true.`. MUSCL implementation can be found in the `traadv_mus.F90` module.

MUSCL has been first implemented in *NEMO* by [Lévy et al. \(2001\)](#). In its formulation, the tracer at velocity points is evaluated assuming a linear tracer variation between two T -points ([figure 4.1](#)). For example, in the i -direction :

$$\tau_u^{mus} = \begin{cases} \tau_i + \frac{1}{2} \left(1 - \frac{u_{i+1/2} \Delta t}{e_{1u}} \right) \widetilde{\partial}_i \tau & \text{if } u_{i+1/2} \geq 0 \\ \tau_{i+1/2} + \frac{1}{2} \left(1 + \frac{u_{i+1/2} \Delta t}{e_{1u}} \right) \widetilde{\partial}_{i+1/2} \tau & \text{if } u_{i+1/2} < 0 \end{cases}$$

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, an upstream flux is used. This choice ensure the *positive* character of the scheme. In addition, fluxes round a grid-point where a runoff is applied can optionally be computed using upstream fluxes (`ln_mus_ups=.true.`).

4.1.4. UBS a.k.a. UP3: Upstream-Biased Scheme (`ln_traadv_ubs`)

The Upstream-Biased Scheme (UBS) is used when `ln_traadv_ubs=.true.`. UBS implementation can be found in the `traadv_mus.F90` module.

The UBS scheme, often called UP3, is also known as the Cell Averaged QUICK scheme (Quadratic Upstream Interpolation for Convective Kinematics). It is an upstream-biased third order scheme based on an upstream-biased parabolic interpolation. For example, in the i -direction:

$$\tau_u^{ubs} = \overline{T}^{i+1/2} - \frac{1}{6} \begin{cases} \tau''_i & \text{if } u_{i+1/2} \geq 0 \\ \tau''_{i+1} & \text{if } u_{i+1/2} < 0 \end{cases} \quad \text{where } \tau''_i = \delta_i [\delta_{i+1/2}[\tau]] \quad (4.5)$$

This results in a dissipatively dominant (i.e. hyper-diffusive) truncation error ([Shchepetkin and McWilliams, 2005](#)). The overall performance of the advection scheme is similar to that reported in [Farrow and Stevens \(1995\)](#). It is a relatively good compromise between accuracy and smoothness. Nevertheless the scheme is not *positive*, meaning that false extrema are permitted, but the amplitude of such are significantly reduced over the centred second or fourth order method. Therefore 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 ([Shchepetkin and McWilliams, 2005](#); [Demange, 2014](#)). Therefore the vertical flux is evaluated using either a 2^{nd} order FCT scheme or a 4^{th} order COMPACT scheme (`nn_ubs_v=2` or `4`).

For stability reasons (see [chapter 2](#)), the first term in [equation 4.5](#) (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 [Webb et al. \(1998\)](#) in the context of the QUICK advection scheme. UBS and QUICK schemes only differ by one coefficient. Replacing 1/6 with 1/8 in [equation 4.5](#) leads to the QUICK advection scheme ([Webb et al., 1998](#)). This option is not available through a namelist parameter, since the 1/6 coefficient is hard coded. Nevertheless it is quite easy to make the substitution in the `traadv_ubs.F90` module and obtain a QUICK scheme.

Note that it is straightforward to rewrite [equation 4.5](#) as follows:

$$\tau_u^{ubs} = \tau_u^{cen4} + \frac{1}{12} \begin{cases} +\tau''_i & \text{if } u_{i+1/2} \geq 0 \\ -\tau''_{i+1} & \text{if } u_{i+1/2} < 0 \end{cases} \quad (4.6)$$

or equivalently

$$u_{i+1/2} \tau_u^{ubs} = u_{i+1/2} \overline{T} - \frac{1}{6} \delta_i [\delta_{i+1/2}[T]]^{i+1/2} - \frac{1}{2} |u|_{i+1/2} \frac{1}{6} \delta_{i+1/2}[\tau''_i]$$

[equation 4.6](#) 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 [equation 4.5](#). 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 the current version of *NEMO* uses the computationally more efficient formulation [equation 4.5](#).

```

!-----
&namtra_ldf ! lateral diffusion scheme for tracers (default: NO selection)
!-----
!
! Operator type:
ln_traldf_off = .false. ! No explicit diffusion
ln_traldf_lap = .false. ! laplacian operator
ln_traldf_blp = .false. ! bilaplacian operator
!
! Direction of action:
ln_traldf_lev = .false. ! iso-level
ln_traldf_hor = .false. ! horizontal (geopotential)
ln_traldf_iso = .false. ! iso-neutral (standard operator)
ln_traldf_triad = .false. ! iso-neutral (triad operator)
!
! iso-neutral options:
ln_traldf_msc = .false. ! Method of Stabilizing Correction (both operators)
rn_slpmax = 0.01 ! slope limit (both operators)
ln_triad_iso = .false. ! pure horizontal mixing in ML (triad only)
rn_sw_triad = 1 ! =1 switching triad ; =0 all 4 triads used (triad only)
ln_botmix_triad = .false. ! lateral mixing on bottom (triad only)
!
! Coefficients:
nn_aht_ijk_t = 0 ! space/time variation of eddy coefficient:
! ! =-20 (=30) read in eddy_diffusivity_2D.nc (...3D.nc) file
! ! = 0 constant
! ! = 10 F(k) =ldf_c1d
! ! = 20 F(i, j) =ldf_c2d
! ! = 21 F(i, j, t) =Treguier et al. JPO 1997 formulation
! ! = 30 F(i, j, k) =ldf_c2d * ldf_c1d
! ! = 31 F(i, j, k, t)=F(local velocity and grid-spacing)
! ! time invariant coefficients: aht0 = 1/2 Ud*Ld (lap case)
! ! or = 1/12 Ud*Ld^3 (blp case)
rn_Ud = 0.01 ! lateral diffusive velocity [m/s] (nn_aht_ijk_t= 0, 10, 20, 30)
rn_Ld = 200.e+3 ! lateral diffusive length [m] (nn_aht_ijk_t= 0, 10)
/

```

namelist 4.2.: `&namtra_ldf`

4.1.5. QCK: QuiCKest scheme (`ln_traadv_qck`)

The Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Terms (QUICKEST) scheme proposed by Leonard (1979) is used when `ln_traadv_qck=.true.`. QUICKEST implementation can be found in the *traadv_qck.F90* module.

QUICKEST is the third order Godunov scheme which is associated with the ULTIMATE QUICKEST limiter (Leonard, 1991). 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 FCT in the vertical direction (as for the UBS case) should be implemented to restore this property.

4.2. Tracer lateral diffusion (*traldf.F90*)

Options are defined through the `&namtra_ldf` (namelist 4.2) namelist variables. They are regrouped in four items, allowing to specify (i) the type of operator used (none, laplacian, bilaplacian), (ii) the direction along which the operator acts (iso-level, horizontal, iso-neutral), (iii) some specific options related to the rotated operators (*i.e.* non-iso-level operator), and (iv) the specification of eddy diffusivity coefficient (either constant or variable in space and time). Item (iv) will be described in chapter 8. The direction along which the operators act is defined through the slope between this direction and the iso-level surfaces. The slope is computed in the *ldfslp.F90* module and will also be described in chapter 8.

The lateral diffusion of tracers is evaluated using a forward scheme, *i.e.* the tracers appearing in its expression are the *before* tracers in time, except for the pure vertical component that appears when a rotation tensor is used. This latter component is solved implicitly together with the vertical diffusion term (see chapter 2). When `ln_traldf_msc=.true.`, a Method of Stabilizing Correction is used in which the pure vertical component is split into an explicit and an implicit part (Lemarié et al., 2012).

4.2.1. Type of operator (`ln_traldf_off` , `ln_traldf_lap` , or `ln_traldf_blp`)

Three operator options are proposed and, one and only one of them must be selected:

ln_traldf_OFF=.true. no operator selected, the lateral diffusive tendency will not be applied to the tracer equation. This option can be used when the selected advection scheme is diffusive enough (MUSCL scheme for example).

ln_traldf_lap=.true. a laplacian operator is selected. This harmonic operator takes the following expression: $\mathcal{L}(T) = \nabla \cdot A_{ht} \nabla T$, where the gradient operates along the selected direction (see subsection 4.2.2), and A_{ht} is the eddy diffusivity coefficient expressed in m^2/s (see chapter 8).

ln_traldf_blp=.true. a bilaplacian operator is selected. This biharmonic operator takes the following expression: $\mathcal{B} = -\mathcal{L}(\mathcal{L}(T)) = -\nabla \cdot b \nabla (\nabla \cdot b \nabla T)$ where the gradient operates along the selected direction, and $b^2 = B_{ht}$ is the eddy diffusivity coefficient expressed in m^4/s (see chapter 8). In the code, the bilaplacian operator is obtained by calling the laplacian twice.

Both laplacian and bilaplacian operators ensure the total tracer variance decrease. Their primary role is to provide strong dissipation at the smallest scale supported by the grid while minimizing the impact on the larger scale features. The main difference between the two operators is the scale selectiveness. The bilaplacian damping time (*i.e.* its spin down time) scales like λ^{-4} for disturbances of wavelength λ (so that short waves damped more rapidly than long ones), whereas the laplacian damping time scales only like λ^{-2} .

4.2.2. Direction of action (**ln_traldf_lev** , **ln_traldf_hor** , **ln_traldf_iso** , or **ln_traldf_triad**)

The choice of a direction of action determines the form of operator used. The operator is a simple (re-entrant) laplacian acting in the (**i,j**) plane when iso-level option is used (**ln_traldf_lev=.true.**) or when a horizontal (*i.e.* geopotential) operator is demanded in z -coordinate (**ln_traldf_hor** and **ln_zco=.true.**). The associated code can be found in the `traldf_lap_blp.F90` module. The operator is a rotated (re-entrant) laplacian when the direction along which it acts does not coincide with the iso-level surfaces, that is when standard or triad iso-neutral option is used (**ln_traldf_iso** or **ln_traldf_triad = .true.**, see `traldf_iso.F90` or `traldf_triad.F90` module, resp.), or when a horizontal (*i.e.* geopotential) operator is demanded in s -coordinate (**ln_traldf_hor** and **ln_sco = .true.**)[†]. In that case, a rotation is applied to the gradient(s) that appears in the operator so that diffusive fluxes acts on the three spatial direction.

The resulting discret form of the three operators (one iso-level and two rotated one) is given in the next two sub-sections.

4.2.3. Iso-level (bi-)laplacian operator (**ln_traldf_iso**)

The laplacian diffusion operator acting along the model (i,j)-surfaces is given by:

$$D_t^{IT} = \frac{1}{b_t} \left(\delta_i \left[A_u^{IT} \frac{e_{2u} e_{3u}}{e_{1u}} \delta_{i+1/2}[T] \right] + \delta_j \left[A_v^{IT} \frac{e_{1v} e_{3v}}{e_{2v}} \delta_{j+1/2}[T] \right] \right) \quad (4.7)$$

where $b_t = e_{1t} e_{2t} e_{3t}$ is the volume of T -cells and where zero diffusive fluxes is assumed across solid boundaries, first (and third in bilaplacian case) horizontal tracer derivative are masked. It is implemented in the `tra_ldf_lap` subroutine found in the `traldf_lap_blp.F90` module. The module also contains `tra_ldf_blp`, the subroutine calling twice `tra_ldf_lap` in order to compute the iso-level bilaplacian operator.

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** or **ln_traldf_blp=.true.**, we have **ln_traldf_lev=.true.** or **ln_traldf_hor=ln_zco=.true.**. In both cases, it significantly contributes to diapycnal mixing. It is therefore never recommended, even when using it in the bilaplacian case.

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 (equation 4.7) at the bottom level require a specific treatment. They are calculated in the `zpsjde.F90` module, described in section 4.9.

[†]In this case, the standard iso-neutral operator will be automatically selected

4.2.4. Standard and triad (bi-)laplacian operator

Standard rotated (bi-)laplacian operator (*traldf_iso.F90*)

The general form of the second order lateral tracer subgrid scale physics (equation 1.17) takes the following semi-discrete space form in z - and s -coordinates:

$$D_T^{lT} = \frac{1}{b_t} \left[\begin{aligned} &\delta_i A_u^{lT} \left(\frac{e_{2u} e_{3u}}{e_{1u}} \delta_{i+1/2}[T] - e_{2u} r_{1u} \overline{\overline{\delta_{k+1/2}[T]}}^{i+1/2,k} \right) \\ &+ \delta_j A_v^{lT} \left(\frac{e_{1v} e_{3v}}{e_{2v}} \delta_{j+1/2}[T] - e_{1v} r_{2v} \overline{\overline{\delta_{k+1/2}[T]}}^{j+1/2,k} \right) \\ &+ \delta_k A_w^{lT} \left(\frac{e_{1w} e_{2w}}{e_{3w}} (r_{1w}^2 + r_{2w}^2) \delta_{k+1/2}[T] \right. \\ &\quad \left. - e_{2w} r_{1w} \overline{\overline{\delta_{i+1/2}[T]}}^{i,k+1/2} - e_{1w} r_{2w} \overline{\overline{\delta_{j+1/2}[T]}}^{j,k+1/2} \right) \end{aligned} \right] \quad (4.8)$$

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 section 8.2. At the surface, bottom and lateral boundaries, the turbulent fluxes of heat and salt are set to zero using the mask technique (see section 7.1).

The operator in equation 4.8 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 section 4.3). For computer efficiency reasons, this term is not computed in the *traldf_iso.F90* module, but in the *trazdf.F90* module where, if iso-neutral mixing is used, the vertical mixing coefficient is simply increased by $\frac{e_{1w} e_{2w}}{e_{3w}} (r_{1w}^2 + r_{2w}^2)$.

This formulation conserves the tracer but does not ensure the decrease of the tracer variance. Nevertheless the treatment performed on the slopes (see chapter 8) allows the model to run safely without any additional background horizontal diffusion (Guilyardi et al., 2001).

Note that in the partial step z -coordinate (`ln_zps=.true.`), the horizontal derivatives at the bottom level in equation 4.8 require a specific treatment. They are calculated in module `zpsjde`, described in section 4.9.

Triad rotated (bi-)laplacian operator (`ln_traldf_triad`)

An alternative scheme developed by Griffies et al. (1998) which ensures tracer variance decreases is also available in *NEMO* (`ln_traldf_triad=.true.`). A complete description of the algorithm is given in appendix D.

The lateral fourth order bilaplacian operator on tracers is obtained by applying (equation 4.7) twice. The operator requires an additional assumption on boundary conditions: both first and third derivative terms normal to the coast are set to zero.

The lateral fourth order operator formulation on tracers is obtained by applying (equation 4.8) 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.

Option for the rotated operators

<code>ln_traldf_msc</code>	Method of Stabilizing Correction (both operators)
<code>rn_slpmax</code>	Slope limit (both operators)
<code>ln_triad_iso</code>	Pure horizontal mixing in ML (triad only)
<code>rn_sw_triad</code>	=1 switching triad; = 0 all 4 triads used (triad only)
<code>ln_botmix_triad</code>	Lateral mixing on bottom (triad only)

4.3. Tracer vertical diffusion (*trazdf.F90*)

Options are defined through the `&namzdf` (namelist 9.1) namelist variables. 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 (equation 1.17) takes the following semi-discrete space form:

$$D_T^{vT} = \frac{1}{e_{3t}} \delta_k \left[\frac{A_w^{vT}}{e_{3w}} \delta_{k+1/2}[T] \right] \quad D_T^{vS} = \frac{1}{e_{3t}} \delta_k \left[\frac{A_w^{vS}}{e_{3w}} \delta_{k+1/2}[S] \right]$$

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.* `ln_zdfddm=.true.`). The way these coefficients are evaluated is given in [chapter 9](#) (ZDF). Furthermore, when iso-neutral mixing is used, both mixing coefficients are increased by $\frac{\epsilon_{1w}\epsilon_{2w}}{\epsilon_{3w}}(r_{1w}^2 + r_{2w}^2)$ to account for the vertical second derivative of [equation 4.8](#).

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 [subsection 4.4.1](#)), whilst at the bottom they are set to zero for heat and salt unless a geothermal flux forcing is prescribed as a bottom boundary condition (see [subsection 4.4.3](#)).

The large eddy coefficient found in the mixed layer together with high vertical resolution implies that there would be too restrictive constraint on the time step if we use explicit time stepping. Therefore an implicit time stepping is preferred for the vertical diffusion since it overcomes the stability constraint.

4.4. External forcing

4.4.1. Surface boundary condition (`trasbc.F90`)

The surface boundary condition for tracers is implemented in a separate module (`trasbc.F90`) instead of entering as a boundary condition on the vertical diffusion operator (as in the case of momentum). This has been found to enhance readability of the code. The two formulations are completely equivalent; the forcing terms in `trasbc` are the surface fluxes divided by the thickness of the top model layer.

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. They are both included directly in Q_{ns} , the surface heat flux, and F_{salt} , the surface salt flux (see [chapter 6](#) for further details). By doing this, the forcing formulation is the same for any tracer (including temperature and salinity).

The surface module (`sbcmod.F90` , see [chapter 6](#)) 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 [subsection 4.4.2](#)) plus the heat content associated with of the mass exchange with the atmosphere and lands.
- sfx The salt flux resulting from ice-ocean mass exchange (freezing, melting, ridging...)
- emp The mass flux exchanged with the atmosphere (evaporation minus precipitation) and possibly with the sea-ice and ice-shelves.
- rnf The mass flux associated with runoff (see [section 6.8](#) for further detail of how it acts on temperature and salinity tendencies)
- $fwfisf$ The mass flux associated with ice shelf melt, (see [section 6.9](#) for further details on how the ice shelf melt is computed and applied).

The surface boundary condition on temperature and salinity is applied as follows:

$$F^T = \frac{1}{C_p \rho_o e_{3t}|_{k=1}} \overline{Q_{ns}}^t \quad F^S = \frac{1}{\rho_o e_{3t}|_{k=1}} \overline{sfx}^t \quad (4.9)$$

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 [chapter 2](#)).

In the linear free surface case (`ln_linssh=.true.`), an additional term has to be added on both temperature and salinity. On temperature, this term remove the heat content associated with mass exchange that has been added to Q_{ns} . On salinity, this term mimics the concentration/dilution effect that would have resulted from a change in the volume of the first level. The resulting surface boundary condition is applied as follows:

$$F^T = \frac{1}{C_p \rho_o e_{3t}|_{k=1}} \overline{(Q_{ns} - C_p emp T|_{k=1})}^t \quad F^S = \frac{1}{\rho_o e_{3t}|_{k=1}} \overline{(sfx - emp S|_{k=1})}^t \quad (4.10)$$

Note that an exact conservation of heat and salt content is only achieved with non-linear free surface. In the linear free surface case, there is a small imbalance. The imbalance is larger than the imbalance associated with the Asselin time filter ([Leclair and Madec, 2009](#)). This is the reason why the modified filter is not applied in the linear free surface case (see [chapter 2](#)).

```

!-----
&namtra_qsr      ! penetrative solar radiation                (ln_traqsr =T)
!-----
!
! type of penetration                (default: NO selection)
ln_qsr_rgb      = .false.      ! RGB light penetration (Red-Green-Blue)
ln_qsr_2bd      = .false.      ! 2BD light penetration (two bands)
ln_qsr_bio      = .false.      ! bio-model light penetration
!
! RGB & 2BD choices:
rn_abs          = 0.58         ! RGB & 2BD: fraction absorbed in the very near surface
rn_si0         = 0.35         ! RGB & 2BD: shortness depth of extinction
nn_chldta      = 0           ! RGB : Chl data (=1) or cst value (=0)
rn_sil         = 23.0         ! 2BD : longest depth of extinction

cn_dir         = './'        ! root directory for the chlorophyl data location

!-----
! file name          ! frequency (hours) ! variable ! time interp. ! clim ! 'yearly' / !
! weights filename ! rotation ! land/sea mask !
! pairing ! filename ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
sn_chl      = 'chlorophyll' , -1. , 'CHLA' , .true. , .true. , 'yearly' , ''
!-----
/

```

namelist 4.3.: `&namtra_qsr`

4.4.2. Solar radiation penetration (*traqsr.F90*)

Options are defined through the `&namtra_qsr` (namelist 4.3) namelist variables. When the penetrative solar radiation option is used (`ln_traqsr=.true.`), the solar radiation penetrates the top few tens of meters of the ocean. If it is not used (`ln_traqsr=.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 [equation 1.4b](#) and the surface boundary condition is modified to take into account only the non-penetrative part of the surface heat flux:

$$\frac{\partial T}{\partial t} = \dots + \frac{1}{\rho_o C_p e_3} \frac{\partial I}{\partial k} \quad (4.11)$$

$$Q_{ns} = Q_{\text{Total}} - Q_{sr}$$

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 [equation 4.11](#) is discretized as follows:

$$\frac{1}{\rho_o C_p e_3} \frac{\partial I}{\partial k} \equiv \frac{1}{\rho_o C_p e_{3t}} \delta_k [I_w] \quad (4.12)$$

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 heat 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 4.3) 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 ([Paulson and Simpson, 1977](#)):

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

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 ([Morel \(1988\)](#), see also [figure 4.2](#)). Light absorption in the ocean depends on particle concentration and is spectrally selective. [Morel \(1988\)](#) 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, [Lengaigne et al. \(2007\)](#) 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 (600-700 nm). For each wave-band, the chlorophyll-dependent attenuation coefficient is fitted to the coefficients computed from the full spectral model of [Morel \(1988\)](#) (as modified by [Morel and](#)

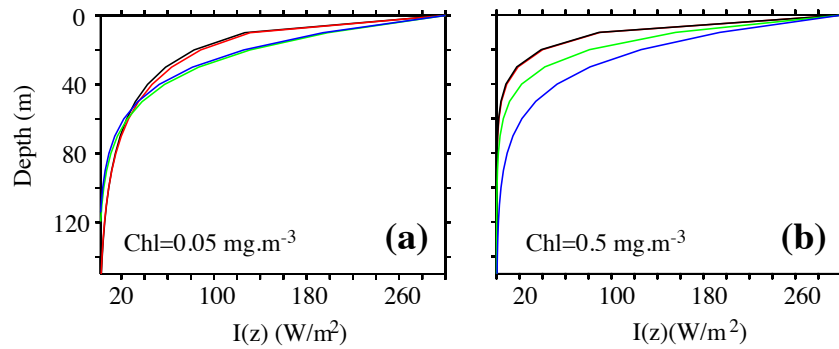


Figure 4.2.: Penetration profile of the downward solar irradiance calculated by four models. Two waveband chlorophyll-independent formulation (blue), a chlorophyll-dependent monochromatic formulation (green), 4 waveband RGB formulation (red), 61 waveband Morel (1988) formulation (black) for a chlorophyll concentration of (a) $\text{Chl}=0.05 \text{ mg.m}^{-3}$ and (b) $\text{Chl}=0.5 \text{ mg.m}^{-3}$. From Lengaigne et al. (2007).

Maritorea (2001)), assuming the same power-law relationship. As shown in figure 4.2, this formulation, called RGB (Red-Green-Blue), reproduces quite closely the light penetration profiles predicted by the full spectral 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 $g.\text{Chl}/L$ (see the routine `trc_oce_rgb` in `trc_oce.F90` module). Four types of chlorophyll can be chosen in the RGB formulation:

`nn_chldta=0` a constant 0.05 $g.\text{Chl}/L$ value everywhere;

`nn_chldta=1` an observed time varying chlorophyll deduced from satellite surface ocean color measurement spread uniformly in the vertical direction;

`nn_chldta=2` same as previous case except that a vertical profile of chlorophyll is used. Following Morel and Berthon (1989), the profile is computed from the local surface chlorophyll value;

`ln_qsr_bio=.true.` simulated time varying chlorophyll by *TOP* biogeochemical model. In this case, the RGB formulation is used to calculate both the phytoplankton light limitation in *PISCES* and the oceanic heating rate.

The trend in equation 4.12 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 ($< 200 \text{ m}$), 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).

4.4.3. Bottom boundary condition (`trabbc.F90` - `ln_trabbc`)

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 $\sim 0.1 \text{ W/m}^2$ (Stein and Stein, 1992)), 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 (Emile-Geay and Madec, 2009).

Options are defined through the `&nambbc` (namelist 4.4) namelist variables. 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 `rn_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 (figure 4.3) (Emile-Geay and Madec, 2009).

4.5. Bottom boundary layer (`trabbl.F90` - `ln_trabbl`)

Options are defined through the `&nambbl` (namelist 4.5) namelist variables. 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.

```

!-----
&nambbc      ! bottom temperature boundary condition          (default: OFF)
!-----
ln_trabbc   = .false.   ! Apply a geothermal heating at the ocean bottom
nn_geoflx   = 2         ! geothermal heat flux: = 1 constant flux
!           !           !           = 2 read variable flux [mW/m2]
rn_geoflx_cst = 86.4e-3 ! Constant value of geothermal heat flux [mW/m2]

cn_dir      = './'      ! root directory for the geothermal data location

!-----
!           !           !           !           !           !           !
!           ! file name   ! frequency (hours) ! variable ! time interp.! clim ! 'yearly' / !
! weights filename ! rotation ! land/sea mask !
!           !           !           ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
!           ! pairing ! filename !
sn_qgh      = 'geothermal_heating.nc' , -12. , 'heatflow' , .false. , .true. , 'yearly' ,
!           !           !           !           !           !           !
!           !           !           !           !           !           !
/

```

namelist 4.4.: &nambbc

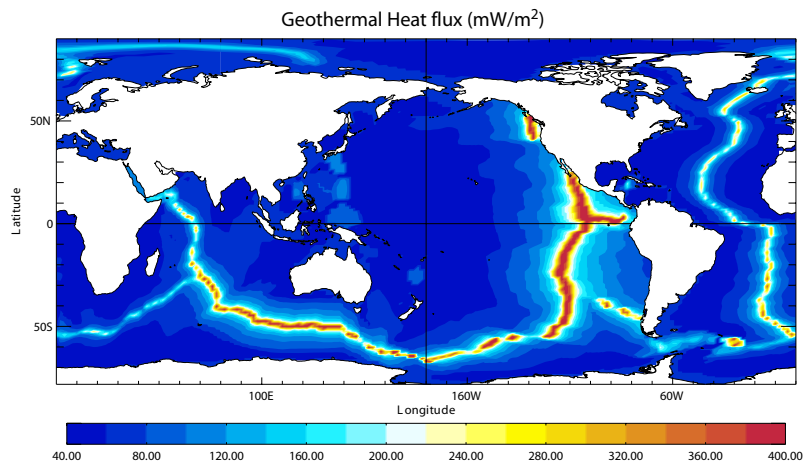


Figure 4.3.: Geothermal Heat flux (in $mW.m^{-2}$) used by Emile-Geay and Madec (2009). It is inferred from the age of the sea floor and the formulae of Stein and Stein (1992).

```

!-----
&nambb1     ! bottom boundary layer scheme                    (default: OFF)
!-----
ln_trabbl   = .false.   ! Bottom Boundary Layer parameterisation flag
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]
/

```

namelist 4.5.: &nambb1

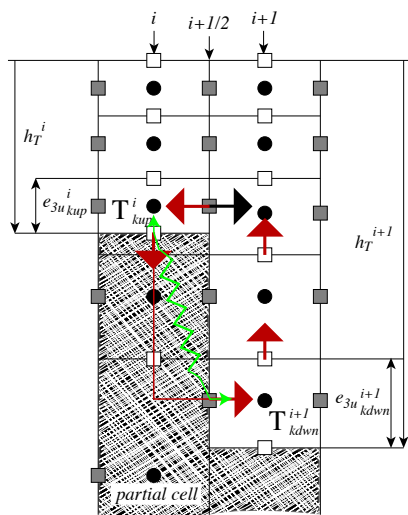


Figure 4.4.: Advective/diffusive Bottom Boundary Layer. The BBL parameterisation is activated when ρ_{kup}^i is larger than ρ_{kdown}^{i+1} . Red arrows indicate the additional overturning circulation due to the advective BBL. The transport of the downslope flow is defined either as the transport of 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 *kdown* ocean bottom cells.

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 (Willebrand et al., 2001), and the thickness of the plume is not resolved.

The idea of the bottom boundary layer (BBL) parameterisation, first introduced by Beckmann and Döscher (1997), 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 Campin and Goosse (1999).

4.5.1. Diffusive bottom boundary layer (`nn_bbl_ldf=1`)

When applying sigma-diffusion (`ln_trabbl=true` and `nn_bbl_ldf` set to 1), the diffusive flux between two adjacent cells at the ocean floor is given by

$$F_{\sigma} = A_l^{\sigma} \nabla_{\sigma} T$$

with ∇_{σ} the lateral gradient operator taken between bottom cells, and A_l^{σ} the lateral diffusivity in the BBL. Following Beckmann and Döscher (1997), the latter is prescribed with a spatial dependence, *i.e.* in the conditional form

$$A_l^{\sigma}(i, j, t) = \begin{cases} A_{bbl} & \text{if } \nabla_{\sigma} \rho \cdot \nabla H < 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.13)$$

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 equation 4.13 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 figure 4.4). In practice, this constraint is applied separately in the two horizontal directions, and the density gradient in equation 4.13 is evaluated with the log gradient formulation:

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

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

4.5.2. Advective bottom boundary layer (`nn_bbl_adv=1, 2`)

When applying an advective BBL (`nn_bbl_adv=1..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

```

!-----
&namtra_dmp ! tracer: T & S newtonian damping (default: OFF)
!-----
ln_tradmp = .false. ! add a damping term (using resto.nc coef.)
nn_zdmp = 0 ! 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 criteria)
cn_resto = 'resto.nc' ! Name of file containing restoration coeff. field (use dmp_tools to create this)
/

```

namelist 4.6.: `&namtra_dmp`

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 figure 4.4) (Beckmann and Döscher, 1997). 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.* $U \cdot \nabla H > 0$).

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 (Campin and Gosse, 1999). 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 example, the resulting transport of the downslope flow, here in the *i*-direction (figure 4.4), is simply given by the following expression:

$$u_{bbl}^{tr} = \gamma g \frac{\Delta\rho}{\rho_o} e_{1u} \min(e_{3ukup}, e_{3ukdwn})$$

where γ , expressed in seconds, is the coefficient of proportionality provided as `rn_gamdbl`, a namelist parameter, and *kup* and *kdown* 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 (Campin and Gosse, 1999).

Scalar properties are advected by this additional transport ($u_{bbl}^{tr}, v_{bbl}^{tr}$) 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 figure 4.4 where the density at level (*i*, *kup*) is larger than the one at level (*i*, *kdown*). The advective BBL scheme modifies the tracer time tendency of the ocean cells near the topographic step by the downslope flow equation 4.14, the horizontal equation 4.15 and the upward equation 4.16 return flows as follows:

$$\partial_t T_{kdown}^{do} \equiv \partial_t T_{kdown}^{do} + \frac{u_{bbl}^{tr}}{b_{t_{kdown}}^{do}} (T_{kup}^{sh} - T_{kdown}^{do}) \quad (4.14)$$

$$\partial_t T_{kup}^{sh} \equiv \partial_t T_{kup}^{sh} + \frac{u_{bbl}^{tr}}{b_{t_{kup}}^{sh}} (T_{kup}^{do} - T_{kup}^{sh}) \quad (4.15)$$

and for $k = kdown - 1, \dots, kup$:

$$\partial_t T_k^{do} \equiv \partial_t S_k^{do} + \frac{u_{bbl}^{tr}}{b_{t_k}^{do}} (T_{k+1}^{do} - T_k^{sh}) \quad (4.16)$$

where b_i is the *T*-cell volume.

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

4.6. Tracer damping (*tradmp.F90*)

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

$$\frac{\partial T}{\partial t} = \dots - \gamma(T - T_o) \quad \frac{\partial S}{\partial t} = \dots - \gamma(S - S_o) \quad (4.17)$$

where γ is the inverse of a time scale, and T_o and S_o are given temperature and salinity fields (usually a climatology). Options are defined through the `&namtra_dmp` (namelist 4.6) namelist variables. The restoring term is added when the namelist parameter `ln_tradmp` is set to true. It also requires that both `ln_tsd_init` and `ln_tsd_dmp` are set

to true in `&namtsd` (namelist 3.2) namelist as well as `sn_tem` and `sn_sal` structures are correctly set (*i.e.* that T_o and S_o are provided in input files and read using `fldread.F90`, see subsection 6.2.1). The restoring coefficient γ is a three-dimensional array read in during the `tra_dmp_init` routine. The file name is specified by the namelist variable `cn_resto`. The DMP_TOOLS are provided to allow users to generate the netcdf file.

The two main cases in which equation 4.17 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 (Sarmiento and Bryan, 1982). 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 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.

The namelist parameter `nn_zdmp` sets whether the damping should be applied in the whole water column or only below the mixed layer (defined either on a density or S_o criterion). It is common to set the damping to zero in the mixed layer as the adjustment time scale is short here (Madec et al., 1996).

For generating `resto.nc`, see the documentation for the DMP tools provided with the source code under `./tools/DMP_TOOLS`.

4.7. Tracer time evolution (`tranxt.F90`)

Options are defined through the `&namdom` (namelist 3.1) namelist variables. The general framework for tracer time stepping is a modified leap-frog scheme (Leclair and Madec, 2009), *i.e.* a three level centred time scheme associated with a Asselin time filter (cf. section 2.5):

$$\begin{aligned} (e_{3t}T)^{t+\Delta t} &= (e_{3t}T)_f^{t-\Delta t} + 2\Delta t e_{3t}^t \text{RHS}^t \\ (e_{3t}T)_f^t &= (e_{3t}T)^t + \gamma \left[(e_{3t}T)_f^{t-\Delta t} - 2(e_{3t}T)^t + (e_{3t}T)^{t+\Delta t} \right] \\ &\quad - \gamma \Delta t \left[Q^{t+\Delta t/2} - Q^{t-\Delta t/2} \right] \end{aligned} \quad (4.18)$$

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`, its default value is `10.e-3`. Note that the forcing correction term in the filter is not applied in linear free surface (`ln_linssh=.true.`) (see subsection 4.4.1). Note 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 `trazdf.F90` module. In this case only the swapping of arrays and the Asselin filtering is done in the `tranxt.F90` module.

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

4.8. Equation of state (`eosbn2.F90`)

4.8.1. Equation of seawater (`ln_teos10` , `ln_teos80` , or `ln_seos`)

The Equation Of Seawater (EOS) is an empirical nonlinear thermodynamic relationship linking seawater density, ρ , to a number of state variables, most typically temperature, salinity and pressure. Because density gradients control the pressure gradient force through the hydrostatic balance, the equation of state provides a fundamental bridge between the distribution of active tracers and the fluid dynamics. Nonlinearities of the EOS are of major importance, in particular influencing the circulation through determination of the static stability below the mixed layer, thus controlling rates of exchange between the atmosphere and the ocean interior (Roquet et al., 2015a). Therefore an accurate EOS based on either the 1980 equation of state (EOS-80, Fofonoff and Millard (1983)) or TEOS-10 (IOC and IAPSO, 2010) standards should be used anytime a simulation of the real ocean circulation is attempted (Roquet et al., 2015a). The use of TEOS-10 is highly recommended because (i) it is the new official EOS, (ii) it is more accurate, being based on an updated database of laboratory measurements, and (iii) it uses Conservative Temperature and Absolute Salinity (instead of potential temperature and practical salinity for EOS-80), both variables being more suitable for use as model variables (IOC and IAPSO, 2010; Graham and McDougall, 2013). EOS-80 is an obsolescent feature of the NEMO system, kept only for backward compatibility. For process studies, it is often convenient to use an approximation of the EOS. To that purpose, a simplified EOS (S-EOS) inspired by Vallis (2006) is also available.

```

!-----
&nameos      !   ocean Equation Of Seawater                (default: NO selection)
!-----
ln_teos10    = .false.      ! = Use TEOS-10
ln_eos80     = .false.      ! = Use EOS80
ln_seos      = .false.      ! = Use S-EOS (simplified Eq.)
!
!           ! S-EOS coefficients (ln_seos=T):
!           ! rd(T,S,Z)*rau0 = -a0*(1+.5*lambda*dT+mu*Z+nu*dS)*dT+b0*dS
rn_a0        = 1.6550e-1    ! thermal expansion coefficient
rn_b0        = 7.6554e-1    ! saline expansion coefficient
rn_lambda1   = 5.9520e-2    ! cabbeling coeff in T^2  (=0 for linear eos)
rn_lambda2   = 7.4914e-4    ! cabbeling coeff in S^2  (=0 for linear eos)
rn_mu1       = 1.4970e-4    ! thermobaric coeff. in T (=0 for linear eos)
rn_mu2       = 1.1090e-5    ! thermobaric coeff. in S (=0 for linear eos)
rn_nu        = 2.4341e-3    ! cabbeling coeff in T*S  (=0 for linear eos)
/

```

namelist 4.7.: &nameos

In the computer code, a density anomaly, $d_a = \rho/\rho_o - 1$, is computed, with ρ_o a reference density. Called *rau0* in the code, ρ_o is set in *phycst.F90* to a value of $1,026 \text{ Kg/m}^3$. 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 that value (Gill, 1982).

Options which control the EOS used are defined through the `&nameos` (namelist 4.7) namelist variables.

ln_teos10=.true. the polyTEOS10-bsq equation of seawater (Roquet et al., 2015b) is used. The accuracy of this approximation is comparable to the TEOS-10 rational function approximation, but it is optimized for a Boussinesq fluid and the polynomial expressions have simpler and more computationally efficient expressions for their derived quantities which make them more adapted for use in ocean models. Note that a slightly higher precision polynomial form is now used replacement of the TEOS-10 rational function approximation for hydrographic data analysis (IOC and IAPSO, 2010). A key point is that conservative state variables are used: Absolute Salinity (unit: g/kg , notation: S_A) and Conservative Temperature (unit: $^{\circ}C$, notation: Θ). The pressure in decibars is approximated by the depth in meters. With TEOS10, the specific heat capacity of sea water, C_p , is a constant. It is set to $C_p = 3991.86795711963 \text{ J.Kg}^{-1}.\text{K}^{-1}$, according to IOC and IAPSO (2010). Choosing polyTEOS10-bsq implies that the state variables used by the model are Θ and S_A . In particular, the initial state defined by the user have to be given as *Conservative Temperature* and *Absolute Salinity*. In addition, when using TEOS10, the Conservative SST is converted to potential SST prior to either computing the air-sea and ice-sea fluxes (forced mode) or sending the SST field to the atmosphere (coupled mode).

ln_eos80=.true. the polyEOS80-bsq equation of seawater is used. It takes the same polynomial form as the polyTEOS10, but the coefficients have been optimized to accurately fit EOS80 (Roquet, personal comm.). The state variables used in both the EOS80 and the ocean model are: the Practical Salinity (unit: psu , notation: S_p) and Potential Temperature (unit: $^{\circ}C$, notation: θ). The pressure in decibars is approximated by the depth in meters. With EOS, the specific heat capacity of sea water, C_p , is a function of temperature, salinity and pressure (Fofonoff and Millard, 1983). Nevertheless, a severe assumption is made in order to have a heat content ($C_p T_p$) which is conserved by the model: C_p is set to a constant value, the TEOS10 value.

ln_seos=.true. a simplified EOS (S-EOS) inspired by Vallis (2006) is chosen, the coefficients of which has been optimized to fit the behavior of TEOS10 (Roquet, personal comm.) (see also Roquet et al. (2015a)). It provides a simplistic linear representation of both cabbeling and thermobaricity effects which is enough for a proper treatment of the EOS in theoretical studies (Roquet et al., 2015a). With such an equation of state there is no longer a distinction between *conservative* and *potential* temperature, as well as between *absolute* and *practical* salinity. S-EOS takes the following expression:

$$d_a(T, S, z) = \frac{1}{\rho_o} \left[-a_0 (1 + 0.5 \lambda_1 T_a + \mu_1 z) * T_a + b_0 (1 - 0.5 \lambda_2 S_a - \mu_2 z) * S_a - \nu T_a S_a \right]$$

with $T_a = T - 10$; $S_a = S - 35$; $\rho_o = 1026 \text{ Kg/m}^3$

where the computer name of the coefficients as well as their standard value are given in table 4.1. In fact, when choosing S-EOS, various approximation of EOS can be specified simply by changing the associated coefficients. Setting to zero the two thermobaric coefficients (μ_1, μ_2) remove thermobaric effect from S-EOS. Setting to zero the three cabbeling coefficients ($\lambda_1, \lambda_2, \nu$) remove cabbeling effect from S-EOS. Keeping non-zero value to a_0 and b_0 provide a linear EOS function of T and S.

coeff.	computer name	S-EOS	description
a_0	rn_a0	$1.6550 \cdot 10^{-1}$	linear thermal expansion coeff.
b_0	rn_b0	$7.6554 \cdot 10^{-1}$	linear haline expansion coeff.
λ_1	rn_lambda1	$5.9520 \cdot 10^{-2}$	cabbeling coeff. in T^2
λ_2	rn_lambda2	$5.4914 \cdot 10^{-4}$	cabbeling coeff. in S^2
ν	rn_nu	$2.4341 \cdot 10^{-3}$	cabbeling coeff. in $T S$
μ_1	rn_mu1	$1.4970 \cdot 10^{-4}$	thermobaric coeff. in T
μ_2	rn_mu2	$1.1090 \cdot 10^{-5}$	thermobaric coeff. in S

Table 4.1.: Standard value of S-EOS coefficients

4.8.2. Brunt-Väisälä frequency

An accurate computation of the ocean stability (i.e. of N , the Brunt-Väisälä frequency) is of paramount importance as determine the ocean stratification and is used in several ocean parameterisations (namely TKE, GLS, Richardson number dependent vertical diffusion, enhanced vertical diffusion, non-penetrative convection, tidal mixing parameterisation, iso-neutral diffusion). In particular, N^2 has to be computed at the local pressure (pressure in decibar being approximated by the depth in meters). The expression for N^2 is given by:

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

where $(T, S) = (\Theta, S_A)$ for TEOS10, (θ, S_p) for TEOS-80, or (T, S) for S-EOS, and, α and β are the thermal and haline expansion coefficients. The coefficients are a polynomial function of temperature, salinity and depth which expression depends on the chosen EOS. They are computed through *eos_rab*, a FORTRAN function that can be found in *eosbn2.F90*.

4.8.3. Freezing point of seawater

The freezing point of seawater is a function of salinity and pressure (Fofonoff and Millard, 1983):

$$T_f(S, p) = (a + b\sqrt{S} + cS) S + dp \quad (4.19)$$

where $a = -0.0575$, $b = 1.710523 \cdot 10^{-3}$, $c = -2.154996 \cdot 10^{-4}$ and $d = -7.53 \cdot 10^{-3}$

equation 4.19 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 equation 4.19 (last term) have been dropped. The freezing point is computed through *eos_fzp*, a FORTRAN function that can be found in *eosbn2.F90*.

4.9. Horizontal derivative in zps-coordinate (zpsjde.F90)

With partial cells (`ln_zps=.true.`) at bottom and top (`ln_isfcav=.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 the hydrostatic pressure gradient calculations (*dynhpg.F90* module). The partial cell properties at the top (`ln_isfcav=.true.`) are computed in the same way as for the bottom. So, only the bottom interpolation is explained below.

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 (figure 4.5). For example, for temperature in the i -direction the needed interpolated temperature, \tilde{T} , is:

$$\tilde{T} = \begin{cases} T^{i+1} - \frac{(e_{3w}^{i+1} - e_{3w}^i)}{e_{3w}^{i+1}} \delta_k T^{i+1} & \text{if } e_{3w}^{i+1} \geq e_{3w}^i \\ T^i + \frac{(e_{3w}^{i+1} - e_{3w}^i)}{e_{3w}^i} \delta_k T^{i+1} & \text{if } e_{3w}^{i+1} < e_{3w}^i \end{cases}$$

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

$$\begin{aligned} \delta_{i+1/2} T &= \begin{cases} \tilde{T} - T^i & \text{if } e_{3w}^{i+1} \geq e_{3w}^i \\ T^{i+1} - \tilde{T} & \text{if } e_{3w}^{i+1} < e_{3w}^i \end{cases} \\ \bar{T}^{i+1/2} &= \begin{cases} (\tilde{T} - T^i)/2 & \text{if } e_{3w}^{i+1} \geq e_{3w}^i \\ (T^{i+1} - \tilde{T})/2 & \text{if } e_{3w}^{i+1} < e_{3w}^i \end{cases} \end{aligned} \quad (4.20)$$

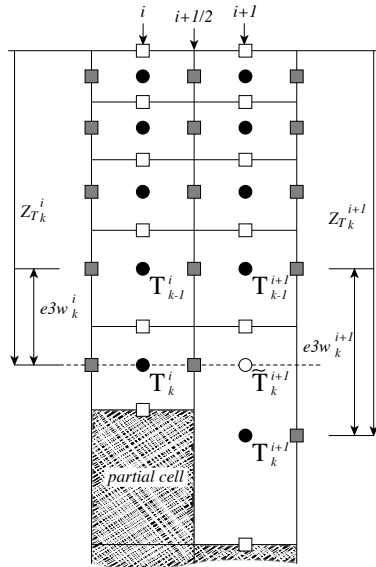


Figure 4.5.: Discretisation of the horizontal difference and average of tracers in the z -partial step coordinate (`ln_zps=.true.`) in the case $(e3w_k^{i+1} - e3w_k^i) > 0$. A linear interpolation is used to estimate $\tilde{T}_k^{i+1/2}$, the tracer value at the depth of the shallower tracer point of the two adjacent bottom T -points. The horizontal difference is then given by: $\delta_{i+1/2}T_k = \tilde{T}_k^{i+1/2} - T_k^i$ and the average by: $\bar{T}_k^{i+1/2} = (\tilde{T}_k^{i+1/2} - T_k^i)/2$.

The computation of horizontal derivative of tracers as well as of density is performed once for all at each time step in `zpsjde.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, $\tilde{\rho}$, is not the same as that used for T and S . Instead of forming a linear approximation of density, we compute $\tilde{\rho}$ 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 subsection 4.8.1):

$$\tilde{\rho} = \rho(\tilde{T}, \tilde{S}, z_u) \quad \text{where } z_u = \min(z_T^{i+1}, z_T^i)$$

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 (section 5.4) and the slopes of neutral surfaces (section 8.2).

Note that in almost all the advection schemes presented in this chapter, both averaging and differencing operators appear. Yet equation 4.20 has not been used in these schemes: in contrast to diffusion and pressure gradient computations, no correction for partial steps is applied for advection. The main motivation is to preserve the domain averaged mean variance of the advected field when using the 2^{nd} order centred scheme. Sensitivity of the advection schemes to the way horizontal averages are performed in the vicinity of partial cells should be further investigated in the near future.

Table of contents

5.1.	Sea surface height and diagnostic variables (η, ζ, χ, w)	48
5.1.1.	Horizontal divergence and relative vorticity (<i>divcur.F90</i>)	48
5.1.2.	Horizontal divergence and relative vorticity (<i>sshwzv.F90</i>)	48
5.2.	Coriolis and advection: vector invariant form	49
5.2.1.	Vorticity term (<i>dynvor.F90</i>)	49
5.2.2.	Kinetic energy gradient term (<i>dynkeg.F90</i>)	52
5.2.3.	Vertical advection term (<i>dynzad.F90</i>)	52
5.3.	Coriolis and advection: flux form	52
5.3.1.	Coriolis plus curvature metric terms (<i>dynvor.F90</i>)	52
5.3.2.	Flux form advection term (<i>dynadv.F90</i>)	53
5.4.	Hydrostatic pressure gradient (<i>dynhpg.F90</i>)	54
5.4.1.	Full step Z -coordinate (<i>ln_dynhpg_zco</i>)	54
5.4.2.	Partial step Z -coordinate (<i>ln_dynhpg_zps</i>)	54
5.4.3.	S - and Z - S -coordinates	54
5.4.4.	Ice shelf cavity	55
5.4.5.	Time-scheme (<i>ln_dynhpg_imp</i>)	55
5.5.	Surface pressure gradient (<i>dynspg.F90</i>)	56
5.5.1.	Explicit free surface (<i>ln_dynspg_exp</i>)	56
5.5.2.	Split-explicit free surface (<i>ln_dynspg_ts</i>)	56
5.5.3.	Filtered free surface (<i>dynspgflt?</i>)	57
5.6.	Lateral diffusion term and operators (<i>dynldf.F90</i>)	57
5.6.1.	Iso-level laplacian (<i>ln_dynldf_lap</i>)	59
5.6.2.	Rotated laplacian (<i>ln_dynldf_iso</i>)	59
5.6.3.	Iso-level bilaplacian (<i>ln_dynldf_bilap</i>)	60
5.7.	Vertical diffusion term (<i>dynzdf.F90</i>)	60
5.8.	External forcings	61
5.9.	Wetting and drying	61
5.9.1.	Directional limiter (<i>wet_dry.F90</i>)	62
5.9.2.	Iterative limiter (<i>wet_dry.F90</i>)	62
5.9.3.	The WAD test cases (<i>usrdef_zgr.F90</i>)	65
5.10.	Time evolution term (<i>dynnxt.F90</i>)	65

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

Using the representation described in [chapter 3](#), several semi-discrete space forms of the dynamical equations are available depending on the vertical coordinate used and on the conservation properties of the vorticity term. In all the equations presented here, the masking has been omitted for simplicity. One must be aware that all the quantities are masked fields and that each time 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 [chapter 6](#), [chapter 8](#) and [chapter 9](#), 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 (*wzvm0d* 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_dyn_ttt_xxx*, where *xxx* is a 3 or 4 letter acronym corresponding to each optional scheme. The corresponding code can be found in the *dyn_ttt_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 (*trddyn?* defined), as described in [chapter 14](#). Furthermore, the tendency terms associated with the 2D barotropic vorticity balance (when *trdvor?* is defined) can be derived from the 3D terms.

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

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

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

$$\zeta = \frac{1}{e_{1f} e_{2f}} \left(\delta_{i+1/2} [e_{2v} v] - \delta_{j+1/2} [e_{1u} u] \right) \quad (5.1)$$

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

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

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.

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

The sea surface height is given by:

$$\begin{aligned} \frac{\partial \eta}{\partial t} &\equiv \frac{1}{e_{1t} e_{2t}} \sum_k \{ \delta_i [e_{2u} e_{3u} u] + \delta_j [e_{1v} e_{3v} v] \} - \frac{emp}{\rho_w} \\ &\equiv \sum_k \chi e_{3t} - \frac{emp}{\rho_w} \end{aligned} \quad (5.2)$$

where *emp* is the surface freshwater budget (evaporation minus precipitation), expressed in Kg/m²/s (which is equal to mm/s), and $\rho_w=1,035 \text{ Kg/m}^3$ is the reference density of sea water (Boussinesq approximation). If river runoff is expressed as a surface freshwater flux (see [chapter 6](#)) then *emp* can be written as the evaporation minus precipitation, minus the


```

!-----
&namdyn_adv    !   formulation of the momentum advection                (default: NO selection)
!-----
ln_dynadv_OFF = .false. ! linear dynamics (no momentum advection)
ln_dynadv_vec = .false. ! vector form - 2nd centered scheme
nn_dynkeg     = 0       ! grad(KE) scheme: =0 C2 ; =1 Hollingsworth correction
ln_dynadv_cen2 = .false. ! flux form - 2nd order centered scheme
ln_dynadv_ubs = .false. ! flux form - 3rd order UBS      scheme
/

```

namelist 5.1.: &namdyn_adv

```

!-----
&namdyn_vor    !   Vorticity / Coriolis scheme                        (default: NO selection)
!-----
ln_dynvor_ene = .false. ! energy conserving scheme
ln_dynvor_ens = .false. ! enstrophy conserving scheme
ln_dynvor_mix = .false. ! mixed scheme
ln_dynvor_enT = .false. ! energy conserving scheme (T-point)
ln_dynvor_eeT = .false. ! energy conserving scheme (een using e3t)
ln_dynvor_eeen = .false. ! energy & enstrophy scheme
nn_eeen_e3f   = 0       ! =0 e3f = mi(mj(e3t))/4
!               ! =1 e3f = mi(mj(e3t))/mi(mj( tmask))
ln_dynvor_msk = .false. ! vorticity multiplied by fmask (=T)      ==>>> PLEASE DO NOT ACTIVATE
!               ! (f-point vorticity schemes only)
/

```

namelist 5.2.: &namdyn_vor

river runoff. The sea-surface height is evaluated using exactly the same time stepping scheme as the tracer equation [equation 4.18](#): a leapfrog scheme in combination with an Asselin time filter, *i.e.* the velocity appearing in [equation 5.2](#) 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 ([Griffies et al., 2001](#); [Leclair and Madec, 2009](#)).

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:

$$\begin{cases} w|_{k_b-1/2} = 0 & \text{where } k_b \text{ is the level just above the sea floor} \\ w|_{k+1/2} = w|_{k-1/2} + e_{3t}|_k \chi|_k - \frac{1}{2\Delta t} (e_{3t}^{t+1}|_k - e_{3t}^{t-1}|_k) \end{cases} \quad (5.3)$$

In the case of a non-linear free surface ($\nabla v \neq 0$), 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 [equation 5.3](#) 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 [equation 5.2](#)).

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 [equation 5.3](#) (see [section 3.1.3](#)).

5.2. Coriolis and advection: vector invariant form

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. Options are defined through the &namdyn_adv ([namelist 5.1](#)) namelist variables. 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 [chapter 7](#).

5.2.1. Vorticity term (dynvor.F90)

Options are defined through the &namdyn_vor ([namelist 5.2](#)) namelist variables. 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 (EEN scheme) (see [section C.5](#)). In the case of ENS, ENE or MIX schemes the land sea mask may be slightly modified to ensure the consistency of vorticity term with analytical equations (`ln_dynvor_con=.true.`). The vorticity terms are all computed in dedicated routines that can be found in the `dynvor.F90` module.

Enstrophy conserving scheme (`ln_dynvor_ens`)

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

$$\begin{cases} + \frac{1}{e_{1u}} \overline{\left(\frac{\zeta + f}{e_{3f}} \right)^i} \overline{\overline{(e_{1v} e_{3v} v)}}^{i,j+1/2} \\ - \frac{1}{e_{2v}} \overline{\left(\frac{\zeta + f}{e_{3f}} \right)^j} \overline{\overline{(e_{2u} e_{3u} u)}}^{i+1/2,j} \end{cases} \quad (5.4)$$

Energy conserving scheme (`ln_dynvor_ene`)

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

$$\begin{cases} + \frac{1}{e_{1u}} \overline{\left(\frac{\zeta + f}{e_{3f}} \right)} \overline{\overline{(e_{1v} e_{3v} v)}}^{i+1/2,j} \\ - \frac{1}{e_{2v}} \overline{\left(\frac{\zeta + f}{e_{3f}} \right)} \overline{\overline{(e_{2u} e_{3u} u)}}^{j+1/2,i} \end{cases} \quad (5.5)$$

Mixed energy/enstrophy conserving scheme (`ln_dynvor_mix`)

For the mixed energy/enstrophy conserving scheme (MIX scheme), a mixture of the two previous schemes is used. It consists of the ENS scheme ([equation 5.4](#)) for the relative vorticity term, and of the ENE scheme ([equation 5.5](#)) applied to the planetary vorticity term.

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

Energy and enstrophy conserving scheme (`ln_dynvor_een`)

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 [Arakawa and Hsu \(1990\)](#). 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 [\(Griffies et al., 1998\)](#) for the discretization of the iso-neutral diffusion operator (see [appendix C](#)).

The [Arakawa and Hsu \(1990\)](#) vorticity advection scheme for a single layer is modified for spherical coordinates as described by [Arakawa and Lamb \(1981\)](#) to obtain the EEN scheme. First consider the discrete expression of the potential vorticity, q , defined at an f -point:

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

where the relative vorticity is defined by ([equation 5.1](#)), the Coriolis parameter is given by $f = 2\Omega \sin \varphi_f$ and the layer thickness at f -points is:

$$e_{3f} = \overline{\overline{e_{3t}}}^{i+1/2,j+1/2} \quad (5.6)$$

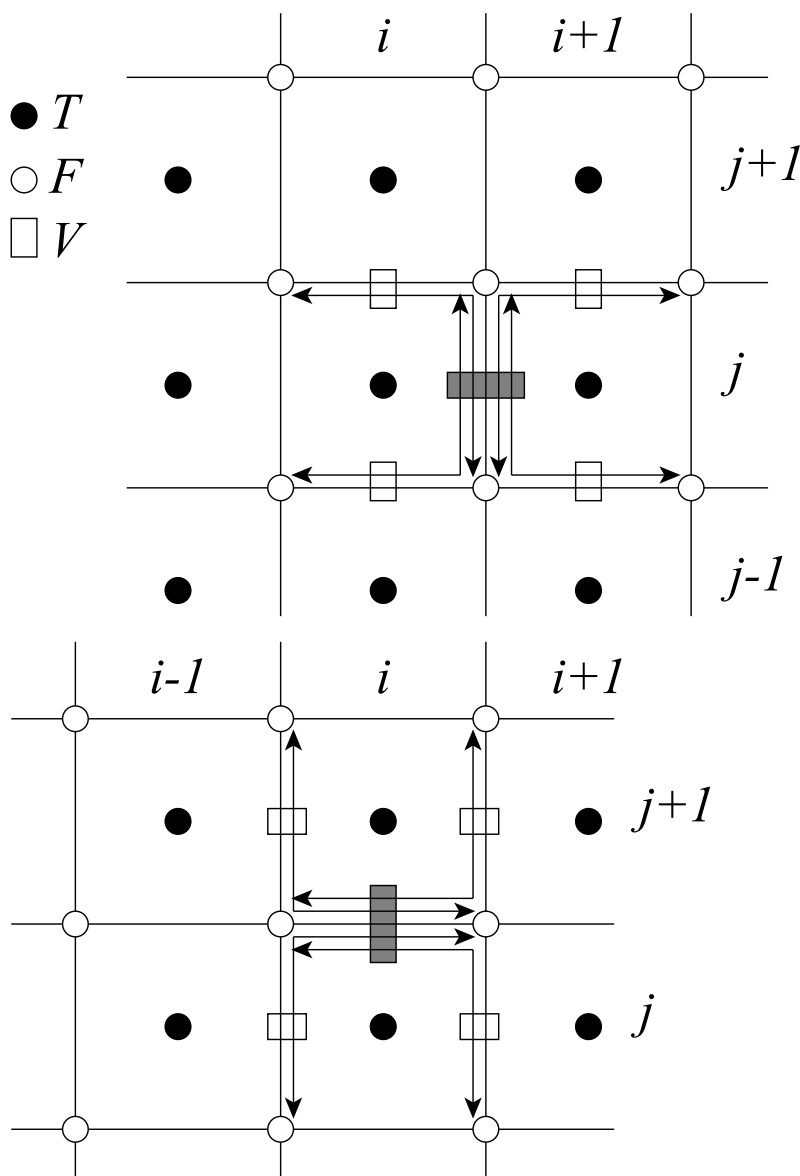


Figure 5.1.: Triads used in the energy and enstrophy conserving scheme (EEN) for u -component (upper panel) and v -component (lower panel).

A key point in [equation 5.6](#) is how the averaging in the i - and j - directions is made. It uses the sum of masked t -point vertical scale factor divided either by the sum of the four t -point masks (`nn_eeen_e3f=1`), or just by 4 (`nn_eeen_e3f=.true.`). The latter case 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 case introduces a sub-grid-scale topography at f -points (with a systematic reduction of e_{3f} when a model level intercept the bathymetry) that tends to reinforce the topography of the flow (*i.e.* the tendency of the flow to follow the isobaths) ([Penduff et al., 2007](#)).

Next, the vorticity triads, ${}^i_j Q_{j_p}^{i_p}$ can be defined at a T -point as the following triad combinations of the neighbouring potential vorticities defined at f -points ([figure 5.1](#)):

$${}^i_j Q_{j_p}^{i_p} = \frac{1}{12} \left(q_{j+j_p}^{i-i_p} + q_{j+i_p}^{i+j_p} + q_{j-j_p}^{i+i_p} \right) \quad (5.7)$$

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:

$$\begin{cases} +q e_3 v \equiv +\frac{1}{e_{1u}} \sum_{i_p, k_p} {}^i_{j_p} Q_{j_p}^{i_p} (e_{1v} e_{3v} v)_{j+j_p}^{i+1/2-i_p} \\ -q e_3 u \equiv -\frac{1}{e_{2v}} \sum_{i_p, k_p} {}^i_{j_p} Q_{j_p}^{i_p} (e_{2u} e_{3u} u)_{j+1/2-j_p}^{i+i_p} \end{cases} \quad (5.8)$$

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.* $\chi=0$) (see [section C.5](#)). 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 ([Le Sommer et al., 2009](#)). Furthermore, used in combination with a partial steps representation of bottom topography, it improves the interaction between current and topography, leading to a larger topography of the flow ([Barnier et al., 2006](#); [Penduff et al., 2007](#)).

5.2.2. Kinetic energy gradient term (*dynkeg.F90*)

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

$$\begin{cases} -\frac{1}{2 e_{1u}} \delta_{i+1/2} [\overline{u^2}^i + \overline{v^2}^j] \\ -\frac{1}{2 e_{2v}} \delta_{j+1/2} [\overline{u^2}^i + \overline{v^2}^j] \end{cases}$$

5.2.3. 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 C](#)).

$$\begin{cases} -\frac{1}{e_{1u} e_{2u} e_{3u}} \frac{\overline{e_{1t} e_{2t} w^{i+1/2} \delta_{k+1/2} [u]^k}}{e_{1t} e_{2t} w^{i+1/2} \delta_{k+1/2} [u]^k} \\ -\frac{1}{e_{1v} e_{2v} e_{3v}} \frac{\overline{e_{1t} e_{2t} w^{j+1/2} \delta_{k+1/2} [u]^k}}{e_{1t} e_{2t} w^{j+1/2} \delta_{k+1/2} [u]^k} \end{cases}$$

When `ln_dynzad_zts=.true.`, a split-explicit time stepping with 5 sub-timesteps is used on the vertical advection term. This option can be useful when the value of the timestep is limited by vertical advection ([Lemarié et al., 2015](#)). Note that in this case, a similar split-explicit time stepping should be used on vertical advection of tracer to ensure a better stability, an option which is only available with a TVD scheme (see `ln_traadv_tvd_zts` in [subsection 4.1.2](#)).

5.3. Coriolis and advection: flux form

Options are defined through the `&namdyn_adv` ([namelist 5.1](#)) namelist variables. 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 [chapter 7](#).

5.3.1. 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:

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

Any of the ([equation 5.4](#)), ([equation 5.5](#)) and ([equation 5.8](#)) schemes can be used to compute the product of the Coriolis parameter and the vorticity. However, the energy-conserving scheme ([equation 5.8](#)) has exclusively been used to date. This term is evaluated using a leapfrog scheme, *i.e.* the velocity is centred in time (*now* velocity).

5.3.2. Flux form advection term (`dynadv.F90`)

The discrete expression of the advection term is given by:

$$\left\{ \begin{array}{l} \frac{1}{e_{1u} e_{2u} e_{3u}} \left(\delta_{i+1/2} \left[\overline{e_{2u} e_{3u} u^i} u_t \right] + \delta_j \left[\overline{e_{1u} e_{3u} v^{i+1/2}} u_f \right] \right. \\ \qquad \qquad \qquad \left. + \delta_k \left[\overline{e_{1w} e_{2w} w^{i+1/2}} u_{uw} \right] \right) \\ \\ \frac{1}{e_{1v} e_{2v} e_{3v}} \left(\delta_i \left[\overline{e_{2u} e_{3u} w^{j+1/2}} v_f \right] + \delta_{j+1/2} \left[\overline{e_{1u} e_{3u} v^i} v_t \right] \right. \\ \qquad \qquad \qquad \left. + \delta_k \left[\overline{e_{1w} e_{2w} w^{j+1/2}} v_{vw} \right] \right) \end{array} \right.$$

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 [Shchepetkin and McWilliams \(2005\)](#). 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 .

CEN2: 2^{nd} order centred scheme (`ln_dynadv_cen2`)

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

$$\left\{ \begin{array}{lll} u_T^{cen2} = \bar{u}^i & u_F^{cen2} = \bar{u}^{j+1/2} & u_{uw}^{cen2} = \bar{u}^{k+1/2} \\ v_F^{cen2} = \bar{v}^{i+1/2} & v_T^{cen2} = \bar{v}^j & v_{vw}^{cen2} = \bar{v}^{k+1/2} \end{array} \right. \quad (5.9)$$

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.

UBS: Upstream Biased Scheme (`ln_dynadv_ubs`)

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:

$$u_T^{ubs} = \bar{u}^i - \frac{1}{6} \begin{cases} u''_{i-1/2} & \text{if } \overline{e_{2u} e_{3u} u^i} \geq 0 \\ u''_{i+1/2} & \text{if } \overline{e_{2u} e_{3u} u^i} < 0 \end{cases} \quad (5.10)$$

where $u''_{i+1/2} = \delta_{i+1/2} [\delta_i [u]]$. This results in a dissipatively dominant (*i.e.* hyper-diffusive) truncation error ([Shchepetkin and McWilliams, 2005](#)). The overall performance of the advection scheme is similar to that reported in [Farrow and Stevens \(1995\)](#). It is a relatively good compromise between accuracy and smoothness. It is not a *positive* scheme, meaning that false extrema are permitted. But 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_{uw}^{ubs} and u_{vw}^{ubs} in [equation 5.9](#) are used. UBS is diffusive and is associated with vertical mixing of momentum.

For stability reasons, the first term in [equation 5.10](#), 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 [Webb et al. \(1998\)](#) 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 [equation 5.10](#) leads to the QUICK advection scheme ([Webb et al., 1998](#)). This option is not available through a namelist parameter, since the $1/6$ coefficient is hard coded. Nevertheless it is quite easy to make the substitution in 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.

```

!-----
&namdyn_hpg      !   Hydrostatic pressure gradient option           (default: NO selection)
!-----
ln_hpg_zco      = .false.    !   z-coordinate - full steps
ln_hpg_zps      = .false.    !   z-coordinate - partial steps (interpolation)
ln_hpg_sco      = .false.    !   s-coordinate (standard jacobian formulation)
ln_hpg_isf      = .false.    !   s-coordinate (sco ) adapted to isf
ln_hpg_djc      = .false.    !   s-coordinate (Density Jacobian with Cubic polynomial)
ln_hpg_prj      = .false.    !   s-coordinate (Pressure Jacobian scheme)
/

```

namelist 5.3.: &namdyn_hpg

5.4. Hydrostatic pressure gradient (dynhpg.F90)

Options are defined through the &namdyn_hpg (namelist 5.3) namelist variables. 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-implicit scheme. At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied.

5.4.1. Full step Z -coordinate (ln_dynhpg_zco)

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)

$$\begin{cases} \delta_{i+1/2} [p^h] \Big|_{k=km} = \frac{1}{2} g \delta_{i+1/2} [e_{3w} \rho] \Big|_{k=km} \\ \delta_{j+1/2} [p^h] \Big|_{k=km} = \frac{1}{2} g \delta_{j+1/2} [e_{3w} \rho] \Big|_{k=km} \end{cases} \quad (5.11)$$

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

$$\begin{cases} \delta_{i+1/2} [p^h] \Big|_k = \delta_{i+1/2} [p^h] \Big|_{k-1} + \frac{1}{2} g \delta_{i+1/2} [e_{3w} \bar{\rho}^{k+1/2}] \Big|_k \\ \delta_{j+1/2} [p^h] \Big|_k = \delta_{j+1/2} [p^h] \Big|_{k-1} + \frac{1}{2} g \delta_{j+1/2} [e_{3w} \bar{\rho}^{k+1/2}] \Big|_k \end{cases} \quad (5.12)$$

Note that the $1/2$ factor in (equation 5.11) 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 (vvl? defined), the surface pressure gradient is included in equation 5.11 and equation 5.12 through the space and time variations of the vertical scale factor e_{3w} .

5.4.2. Partial step Z -coordinate (ln_dynhpg_zps)

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

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

5.4.3. S - and Z - S -coordinates

Pressure gradient formulations in an s -coordinate have been the subject of a vast number of papers (*e.g.*, Song (1998); Shchepetkin and McWilliams (2005)). A number of different pressure gradient options are coded but the ROMS-like, density Jacobian with cubic polynomial method is currently disabled whilst known bugs are under investigation.

- Traditional coding (see for example Madec et al. (1996): (`ln_dynhpg_sco=.true.`)

$$\begin{cases} -\frac{1}{\rho_o e_{1u}} \delta_{i+1/2} [p^h] + \frac{g \bar{\rho}^{i+1/2}}{\rho_o e_{1u}} \delta_{i+1/2} [z_t] \\ -\frac{1}{\rho_o e_{2v}} \delta_{j+1/2} [p^h] + \frac{g \bar{\rho}^{j+1/2}}{\rho_o e_{2v}} \delta_{j+1/2} [z_t] \end{cases} \quad (5.13)$$

Where the first term is the pressure gradient along coordinates, computed as in equation 5.11 - equation 5.12, 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}).

- Traditional coding with adaptation for ice shelf cavities (`ln_dynhpg_isf=.true.`). This scheme need the activation of ice shelf cavities (`ln_isfcav=.true.`).
- Pressure Jacobian scheme (prj) (a research paper in preparation) (`ln_dynhpg_prj=.true.`)
- Density Jacobian with cubic polynomial scheme (DJC) (Shchepetkin and McWilliams, 2005) (`ln_dynhpg_djc=.true.`) (currently disabled; under development)

Note that expression equation 5.13 is commonly used when the variable volume formulation is activated (`vv1?`) because in that case, even with a flat bottom, the coordinate surfaces are not horizontal but follow the free surface (Levier et al., 2007). The pressure jacobian scheme (`ln_dynhpg_prj=.true.`) is available as an improved option to `ln_dynhpg_sco=.true.` when `vv1?` is active. The pressure Jacobian scheme uses a constrained cubic spline to reconstruct the density profile across the water column. This method maintains the monotonicity between the density nodes. The pressure can be calculated by analytical integration of the density profile and a pressure Jacobian method is used to solve the horizontal pressure gradient. This method can provide a more accurate calculation of the horizontal pressure gradient than the standard scheme.

5.4.4. Ice shelf cavity

Beneath an ice shelf, the total pressure gradient is the sum of the pressure gradient due to the ice shelf load and the pressure gradient due to the ocean load (`ln_dynhpg_isf=.true.`).

The main hypothesis to compute the ice shelf load is that the ice shelf is in an isostatic equilibrium. The top pressure is computed integrating from surface to the base of the ice shelf a reference density profile (prescribed as density of a water at 34.4 PSU and -1.9°C) and corresponds to the water replaced by the ice shelf. This top pressure is constant over time. A detailed description of this method is described in Losch (2008).

The pressure gradient due to ocean load is computed using the expression equation 5.13 described in subsection 5.4.3.

5.4.5. Time-scheme (`ln_dynhpg_imp`)

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 (Brown and Campana, 1978; Maltrud et al., 1998). 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.`):

$$\frac{u^{t+\Delta t} - u^{t-\Delta t}}{2\Delta t} = \dots - \frac{1}{\rho_o e_{1u}} \delta_{i+1/2} [p_h^t] \quad (5.14)$$

- semi-implicit scheme (`ln_dynhpg_imp=.true.`):

$$\frac{u^{t+\Delta t} - u^{t-\Delta t}}{2\Delta t} = \dots - \frac{1}{4\rho_o e_{1u}} \delta_{i+1/2} [p_h^{t+\Delta t} + 2p_h^t + p_h^{t-\Delta t}] \quad (5.15)$$

The semi-implicit time scheme equation 5.15 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 equation 5.15 compared to that using the standard leapfrog scheme equation 5.14. Note that equation 5.15 is equivalent to applying a time filter to the pressure gradient to eliminate high frequency IGWs. Obviously, when using equation 5.15, 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

```

!-----
&namdyn_spg      !   surface pressure gradient                               (default: NO selection)
!-----
ln_dynspg_exp   = .false.   ! explicit free surface
ln_dynspg_ts    = .false.   ! split-explicit free surface
ln_bt_fw        = .true.    ! Forward integration of barotropic Eqs.
ln_bt_av        = .true.    ! Time filtering of barotropic variables
nn_btflt        = 1         ! Time filter choice = 0 None
!                                     ! = 1 Boxcar over nn_baro sub-steps
!                                     ! = 2 Boxcar over 2*nn_baro " "
ln_bt_auto      = .true.    ! Number of sub-step defined from:
rn_bt_cmax      = 0.8       ! =T : the Maximum Courant Number allowed
nn_baro         = 30        ! =F : the number of sub-step in rn_rdt seconds
rn_bt_alpha     = 0.        ! Temporal diffusion parameter (if ln_bt_av=F)
/

```

namelist 5.4.: &namdyn_spg

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:

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

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.

5.5. Surface pressure gradient (*dynspg.F90*)

Options are defined through the `&namdyn_spg` (namelist 5.4) namelist variables. The surface pressure gradient term is related to the representation of the free surface (section 1.2). The main distinction is between the fixed volume case (linear free surface) and the variable volume case (nonlinear free surface, `vv1?` is defined). In the linear free surface case (subsection 1.2.2) the vertical scale factors e_3 are fixed in time, while they are time-dependent in the nonlinear case (subsection 1.2.2). 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 ??), 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_ft.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 (section 1.2). 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_ft.F90* and not in *dynnxt.F90*.

5.5.1. Explicit free surface (`ln_dynspg_exp`)

In the explicit free surface formulation (`ln_dynspg_exp` set to true), 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 :

$$\begin{cases} -\frac{1}{e_{1u} \rho_o} \delta_{i+1/2} [\rho \eta] \\ -\frac{1}{e_{2v} \rho_o} \delta_{j+1/2} [\rho \eta] \end{cases} \quad (5.16)$$

Note that in the non-linear free surface case (*i.e.* `vv1?` 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.

5.5.2. Split-explicit free surface (`ln_dynspg_ts`)

The split-explicit free surface formulation used in *NEMO* (`ln_dynspg_ts` set to true), also called the time-splitting formulation, follows the one proposed by [Shchepetkin and McWilliams \(2005\)](#). 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 (figure 5.2). 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: $\Delta t_e = \Delta t / nn_baro$. This parameter can be optionally defined automatically (`ln_bt_nn_auto=.true.`) considering that the stability of the barotropic system is essentially controlled by external waves propagation. Maximum Courant number is in that case time independent, and easily computed online from the input bathymetry. Therefore, Δt_e is adjusted so that the Maximum allowed Courant number is smaller than `rn_bt_cmax`.

The barotropic mode solves the following equations:

$$\frac{\partial \mathbf{U}_h}{\partial t} = -f \mathbf{k} \times \mathbf{U}_h - g \nabla_h \eta - \frac{c_b^U}{H + \eta} \bar{\mathbf{U}}_h + \bar{\mathbf{G}} \quad (5.17)$$

$$\frac{\partial \eta}{\partial t} = -\nabla \cdot [(H + \eta) \mathbf{U}_h] + P - E$$

where $\bar{\mathbf{G}}$ is a forcing term held constant, containing coupling term between modes, surface atmospheric forcing as well as slowly varying barotropic terms not explicitly computed to gain efficiency. The third term on the right hand side of equation 5.17 represents the bottom stress (see section 9.4), explicitly accounted for at each barotropic iteration. Temporal discretization of the system above follows a three-time step Generalized Forward Backward algorithm detailed in Shchepetkin and McWilliams (2005). AB3-AM4 coefficients used in *NEMO* follow the second-order accurate, "multi-purpose" stability compromise as defined in Shchepetkin and McWilliams (2009) (see their figure 12, lower left).

In the default case (`ln_bt_fw=.true.`), the external mode is integrated between *now* and *after* baroclinic time-steps (figure 5.2a). To avoid aliasing of fast barotropic motions into three dimensional equations, time filtering is eventually applied on barotropic quantities (`ln_bt_av=.true.`). In that case, the integration is extended slightly beyond *after* time step to provide time filtered quantities. These are used for the subsequent initialization of the barotropic mode in the following baroclinic step. Since external mode equations written at baroclinic time steps finally follow a forward time stepping scheme, asselin filtering is not applied to barotropic quantities.

Alternatively, one can choose to integrate barotropic equations starting from *before* time step (`ln_bt_fw=.false.`). Although more computationally expensive (`nn_baro` additional iterations are indeed necessary), the baroclinic to barotropic forcing term given at *now* time step become centred in the middle of the integration window. It can easily be shown that this property removes part of splitting errors between modes, which increases the overall numerical robustness.

As far as tracer conservation is concerned, barotropic velocities used to advect tracers must also be updated at *now* time step. This implies to change the traditional order of computations in *NEMO*: most of momentum trends (including the barotropic mode calculation) updated first, tracers' after. This *de facto* makes semi-implicit hydrostatic pressure gradient (see section subsection 5.4.5) and time splitting not compatible. Advective barotropic velocities are obtained by using a secondary set of filtering weights, uniquely defined from the filter coefficients used for the time averaging (Shchepetkin and McWilliams (2005)). Consistency between the time averaged continuity equation and the time stepping of tracers is here the key to obtain exact conservation.

One can eventually choose to feedback instantaneous values by not using any time filter (`ln_bt_av=.false.`). In that case, external mode equations are continuous in time, *i.e.* they are not re-initialized when starting a new sub-stepping sequence. This is the method used so far in the POM model, the stability being maintained by refreshing at (almost) each barotropic time step advection and horizontal diffusion terms. Since the latter terms have not been added in *NEMO* for computational efficiency, removing time filtering is not recommended except for debugging purposes. This may be used for instance to appreciate the damping effect of the standard formulation on external gravity waves in idealized or weakly non-linear cases. Although the damping is lower than for the filtered free surface, it is still significant as shown by Levier et al. (2007) in the case of an analytical barotropic Kelvin wave.

5.5.3. Filtered free surface (`dynspgflt`?)

The filtered formulation follows the Roulet and Madec (2000) implementation. The extra term introduced in the equations (see subsection 1.2.2) is solved implicitly. The elliptic solvers available in the code are documented in chapter 14.

Note that in the linear free surface formulation (`vv1?` 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.

5.6. Lateral diffusion term and operators (`dynldf.F90`)

Options are defined through the `&namdyn_ldf` (namelist 5.5) namelist variables. 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 (chapter 8). The lateral diffusion of momentum is evaluated using a forward scheme, *i.e.* the velocity appearing in its expression is the *before* velocity in time, except for the pure vertical component that appears when a tensor of rotation is used. This latter term is solved implicitly together with the vertical diffusion term (see chapter 2).

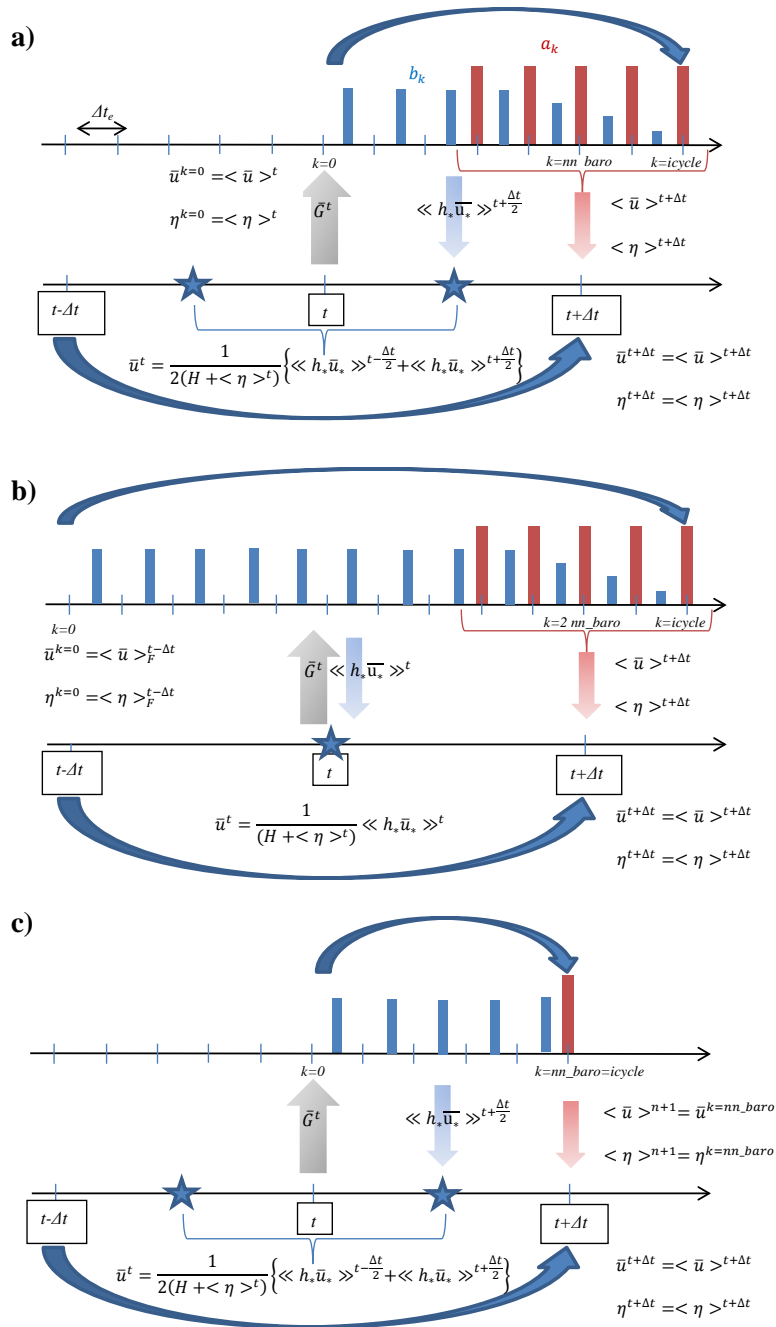


Figure 5.2.: Schematic of the split-explicit time stepping scheme for the external and internal modes. Time increases to the right. In this particular example, a boxcar averaging window over `nn_baro` barotropic time steps is used (`nn_bt_filt=1`) and `nn_baro=5`. Internal mode time steps (which are also the model time steps) are denoted by $t - \Delta t$, t and $t + \Delta t$. Variables with k superscript refer to instantaneous barotropic variables, $\langle \rangle$ and $\langle\langle \rangle\rangle$ operator refer to time filtered variables using respectively primary (red vertical bars) and secondary weights (blue vertical bars). The former are used to obtain time filtered quantities at $t + \Delta t$ while the latter are used to obtain time averaged transports to advect tracers. a) Forward time integration: `ln_bt_fw=.true.`, `ln_bt_av=.true.` . b) Centred time integration: `ln_bt_fw=.false.`, `ln_bt_av=.true.` . c) Forward time integration with no time filtering (POM-like scheme): `ln_bt_fw=.true.`, `ln_bt_av=.false.` .

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

```

!-----
&namdyn_ldf ! lateral diffusion on momentum (default: NO selection)
!-----
!
! Type of the operator :
ln_dynldf_OFF = .false. ! No operator (i.e. no explicit diffusion)
ln_dynldf_lap = .false. ! laplacian operator
ln_dynldf_blp = .false. ! bilaplacian operator
!
! Direction of action :
ln_dynldf_lev = .false. ! iso-level
ln_dynldf_hor = .false. ! horizontal (geopotential)
ln_dynldf_iso = .false. ! iso-neutral (lap only)
!
! Coefficient
nn_ahm_ijk_t = 0 ! space/time variation of eddy coefficient :
! ! =-30 read in eddy_viscosity_3D.nc file
! ! =-20 read in eddy_viscosity_2D.nc file
! ! = 0 constant
! ! = 10 F(k)=c1d
! ! = 20 F(i,j)=F(grid spacing)=c2d
! ! = 30 F(i,j,k)=c2d*c1d
! ! = 31 F(i,j,k)=F(grid spacing and local velocity)
! ! = 32 F(i,j,k)=F(local gridscale and deformation rate)
!
! time invariant coefficients : ahm = 1/2 Uv*Lv (lap case)
! or = 1/12 Uv*Lv^3 (blp case)
rn_Uv = 0.1 ! lateral viscous velocity [m/s] (nn_ahm_ijk_t= 0, 10, 20, 30)
rn_Lv = 10.e+3 ! lateral viscous length [m] (nn_ahm_ijk_t= 0, 10)
!
! Smagorinsky settings (nn_ahm_ijk_t= 32) :
rn_csmc = 3.5 ! Smagorinsky constant of proportionality
rn_minfac = 1.0 ! multiplier of theoretical lower limit
rn_maxfac = 1.0 ! multiplier of theoretical upper limit
!
! iso-neutral laplacian operator (ln_dynldf_iso=T) :
rn_ahm_b = 0.0 ! background eddy viscosity [m2/s]
/
    
```

namelist 5.5.: &namdyn_ldf

5.6.1. Iso-level laplacian operator (`ln_dynldf_lap`)

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

$$\begin{cases} D_u^{lU} = \frac{1}{e_{1u}} \delta_{i+1/2} [A_T^{lm} \chi] - \frac{1}{e_{2u} e_{3u}} \delta_j [A_f^{lm} e_{3f} \zeta] \\ D_v^{lU} = \frac{1}{e_{2v}} \delta_{j+1/2} [A_T^{lm} \chi] + \frac{1}{e_{1v} e_{3v}} \delta_i [A_f^{lm} e_{3f} \zeta] \end{cases} \quad (5.18)$$

As explained in [subsection 1.5.2](#), 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.

5.6.2. Rotated laplacian operator (`ln_dynldf_iso`)

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:

$$D_u^U = \frac{1}{e_{1u} e_{2u} e_{3u}} \left\{ \begin{aligned} & \delta_{i+1/2} \left[A_T^{lm} \left(\frac{e_{2t} e_{3t}}{e_{1t}} \delta_i[u] - e_{2t} r_{1t} \overline{\overline{\delta_{k+1/2}[u]}}^{i,k} \right) \right] \\ & + \delta_j \left[A_f^{lm} \left(\frac{e_{1f} e_{3f}}{e_{2f}} \delta_{j+1/2}[u] - e_{1f} r_{2f} \overline{\overline{\delta_{k+1/2}[u]}}^{j+1/2,k} \right) \right] \\ & + \delta_k \left[A_{uw}^{lm} \left(-e_{2u} r_{1uw} \overline{\overline{\delta_{i+1/2}[u]}}^{i+1/2,k+1/2} \right. \right. \\ & \quad \left. \left. - e_{1u} r_{2uw} \overline{\overline{\delta_{j+1/2}[u]}}^{j,k+1/2} \right. \right. \\ & \quad \left. \left. + \frac{e_{1u} e_{2u}}{e_{3uw}} (r_{1uw}^2 + r_{2uw}^2) \delta_{k+1/2}[u] \right) \right] \end{aligned} \right\} \quad (5.19)$$

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

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 (chapter 8).

5.6.3. Iso-level bilaplacian operator (ln_dynldf_bilap)

The lateral fourth order operator formulation on momentum is obtained by applying equation 5.18 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 chapter 7).

5.7. Vertical diffusion term (dynzdf.F90)

Options are defined through the &namzdf (namelist 9.1) namelist variables. 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 chapter 2). 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 equation 1.17 take the following semi-discrete space form:

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

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 chapter 9).

```

!-----
&namwad      !   Wetting and Drying (WaD)                               (default: OFF)
!-----
ln_wd_il     = .false.  !   T/F activation of iterative limiter
ln_wd_dl     = .false.  !   T/F activation of directional limiter
ln_wd_dl_bc  = .false.  !   T/F Directional limiter Baroclinic option
ln_wd_dl_rmp = .false.  !   T/F Turn on directional limiter ramp
rn_wdmin0    = 0.30    !   depth at which WaD starts
rn_wdmin1    = 0.2     !   Minimum wet depth on dried cells
rn_wdmin2    = 0.0001  !   Tolerance of min wet depth on dried cells
rn_wdld      = 2.5     !   Land elevation below which WaD is allowed
nn_wdit      = 20      !   Max iterations for WaD limiter
rn_wd_sbcdep = 5.0     !   Depth at which to taper sbc fluxes
rn_wd_sbcfra = 0.999   !   Fraction of SBC fluxes at taper depth (Must be <1)
/

```

namelist 5.6.: &namwad

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,

$$\left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_{z=1} = \frac{1}{\rho_o} \begin{pmatrix} \tau_u \\ \tau_v \end{pmatrix} \quad (5.20)$$

where (τ_u, τ_v) are the two components of the wind stress vector in the (\mathbf{i}, \mathbf{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 [chapter 6](#)).

The turbulent flux of momentum at the bottom of the ocean is specified through a bottom friction parameterisation (see [section 9.4](#))

5.8. External forcings

Besides the surface and bottom stresses (see the above section) which are introduced as boundary conditions on the vertical mixing, three other forcings may enter the dynamical equations by affecting the surface pressure gradient.

(1) When `ln_apr_dyn=.true.` (see [section 6.6](#)), the atmospheric pressure is taken into account when computing the surface pressure gradient.

(2) When `ln_tide_pot=.true.` and `ln_tide=.true.` (see [section 6.7](#)), the tidal potential is taken into account when computing the surface pressure gradient.

(3) When `nn_ice_embd=2` and LIM or CICE is used (*i.e.* when the sea-ice is embedded in the ocean), the snow-ice mass is taken into account when computing the surface pressure gradient.

5.9. Wetting and drying

There are two main options for wetting and drying code (wd): (a) an iterative limiter (il) and (b) a directional limiter (dl). The directional limiter is based on the scheme developed by [Warner et al. \(2013\)](#) for RO MS which was in turn based on ideas developed for POM by [Oey \(2006\)](#). The iterative limiter is a new scheme. The iterative limiter is activated by setting `ln_wd_il = .true.` and `ln_wd_dl = .false.`. The directional limiter is activated by setting `ln_wd_dl = .true.` and `ln_wd_il = .false.`.

The following terminology is used. The depth of the topography (positive downwards) at each (i, j) point is the quantity stored in array `ht_wd` in the *NEMO* code. The height of the free surface (positive upwards) is denoted by `ssh`. Given the sign conventions used, the water depth, h , is the height of the free surface plus the depth of the topography (*i.e.* `ssh + ht_wd`).

Both wd schemes take all points in the domain below a land elevation of `rn_wdld` to be covered by water. They require the topography specified with a model configuration to have negative depths at points where the land is higher than the topography's reference sea-level. The vertical grid in *NEMO* is normally computed relative to an initial state with zero sea surface height elevation. The user can choose to compute the vertical grid and heights in the model relative to a non-zero reference height for the free surface. This choice affects the calculation of the metrics and depths (*i.e.* the `e3t_0`, `ht_0` etc. arrays).

Points where the water depth is less than `rn_wdmin1` are interpreted as "dry". `rn_wdmin1` is usually chosen to be of order 0.05m but extreme topographies with very steep slopes require larger values for normal choices of time-step. Surface fluxes are also switched off for dry cells to prevent freezing, boiling etc. of very thin water layers. The fluxes are tapered

down using a tanh weighting function to no flux as the dry limit `rn_wdmin1` is approached. Even wet cells can be very shallow. The depth at which to start tapering is controlled by the user by setting `rn_wd_sbcddep`. The fraction (< 1) of surface fluxes to use at this depth is set by `rn_wd_sbcfra`.

Both versions of the code have been tested in six test cases provided in the `WAD_TEST_CASES` configuration and in “realistic” configurations covering parts of the north-west European shelf. All these configurations have used pure sigma coordinates. It is expected that the wetting and drying code will work in domains with more general s-coordinates provided the coordinates are pure sigma in the region where wetting and drying actually occurs.

The next sub-section describes the directional limiter and the following sub-section the iterative limiter. The final sub-section covers some additional considerations that are relevant to both schemes.

5.9.1. Directional limiter (*wet_dry.F90*)

The principal idea of the directional limiter is that water should not be allowed to flow out of a dry tracer cell (i.e. one whose water depth is less than `rn_wdmin1`).

All the changes associated with this option are made to the barotropic solver for the non-linear free surface code within `dynspg_ts`. On each barotropic sub-step the scheme determines the direction of the flow across each face of all the tracer cells and sets the flux across the face to zero when the flux is from a dry tracer cell. This prevents cells whose depth is `rn_wdmin1` or less from drying out further. The scheme does not force h (the water depth) at tracer cells to be at least the minimum depth and hence is able to conserve mass / volume.

The flux across each u -face of a tracer cell is multiplied by a factor `zuwdmask` (an array which depends on `ji` and `jj`). If the user sets `ln_wd_dl_ramp=.false.` then `zuwdmask` is 1 when the flux is from a cell with water depth greater than `rn_wdmin1` and 0 otherwise. If the user sets `ln_wd_dl_ramp=.true.` the flux across the face is ramped down as the water depth decreases from $2 * \text{rn_wdmin1}$ to `rn_wdmin1`. The use of this ramp reduced grid-scale noise in idealised test cases.

At the point where the flux across a u -face is multiplied by `zuwdmask`, we have chosen also to multiply the corresponding velocity on the “now” step at that face by `zuwdmask`. We could have chosen not to do that and to allow fairly large velocities to occur in these “dry” cells. The rationale for setting the velocity to zero is that it is the momentum equations that are being solved and the total momentum of the upstream cell (treating it as a finite volume) should be considered to be its depth times its velocity. This depth is considered to be zero at “dry” u -points consistent with its treatment in the calculation of the flux of mass across the cell face.

Warner et al. (2013) state that in their scheme the velocity masks at the cell faces for the baroclinic timesteps are set to 0 or 1 depending on whether the average of the masks over the barotropic sub-steps is respectively less than or greater than 0.5. That scheme does not conserve tracers in integrations started from constant tracer fields (tracers independent of x , y and z). Our scheme conserves constant tracers because the velocities used at the tracer cell faces on the baroclinic timesteps are carefully calculated by `dynspg_ts` to equal their mean value during the barotropic steps. If the user sets `ln_wd_dl_bc=.true.`, the baroclinic velocities are also multiplied by a suitably weighted average of `zuwdmask`.

5.9.2. Iterative limiter (*wet_dry.F90*)

Iterative flux limiter (*wet_dry.F90*)

The iterative limiter modifies the fluxes across the faces of cells that are either already “dry” or may become dry within the next time-step using an iterative method.

The flux limiter for the barotropic flow (devised by Hedong Liu) can be understood as follows:

The continuity equation for the total water depth in a column

$$\frac{\partial h}{\partial t} + \nabla \cdot (hu) = 0. \quad (5.21)$$

can be written in discrete form as

$$\frac{e_1 e_2}{\Delta t} (h_{i,j}(t_{n+1}) - h_{i,j}(t_e)) = -(\text{flxu}_{i+1,j} - \text{flxu}_{i,j} + \text{flxv}_{i,j+1} - \text{flxv}_{i,j}) \quad (5.22)$$

$$= \text{zzflx}_{i,j}. \quad (5.23)$$

In the above h is the depth of the water in the column at point (i, j) , $\text{flxu}_{i+1,j}$ is the flux out of the “eastern” face of the cell and $\text{flxv}_{i,j+1}$ the flux out of the “northern” face of the cell; t_{n+1} is the new timestep, t_e is the old timestep (either t_b or t_n) and $\Delta t = t_{n+1} - t_e$; $e_1 e_2$ is the area of the tracer cells centred at (i, j) and zzflx is the sum of the fluxes through all the faces.

The flux limiter splits the flux zzflx into fluxes that are out of the cell (zzflxp) and fluxes that are into the cell (zzflxn). Clearly

$$\text{zzflx}_{i,j} = \text{zzflxp}_{i,j} + \text{zzflxn}_{i,j}. \quad (5.24)$$

The flux limiter iteratively adjusts the fluxes $flxu$ and $flxv$ until none of the cells will “dry out”. To be precise the fluxes are limited until none of the cells has water depth less than rn_wdmin1 on step $n + 1$.

Let the fluxes on the m th iteration step be denoted by $flxu^{(m)}$ and $flxv^{(m)}$. Then the adjustment is achieved by seeking a set of coefficients, $zcoef_{i,j}^{(m)}$ such that:

$$\begin{aligned} zzflxp_{i,j}^{(m)} &= zcoef_{i,j}^{(m)} zzflxp_{i,j}^{(0)} \\ zzflxn_{i,j}^{(m)} &= zcoef_{i,j}^{(m)} zzflxn_{i,j}^{(0)} \end{aligned} \quad (5.25)$$

where the coefficients are 1.0 generally but can vary between 0.0 and 1.0 around cells that would otherwise dry.

The iteration is initialised by setting

$$zzflxp_{i,j}^{(0)} = zzflxp_{i,j}, \quad zzflxn_{i,j}^{(0)} = zzflxn_{i,j}. \quad (5.26)$$

The fluxes out of cell (i, j) are updated at the $m + 1$ th iteration if the depth of the cell on timestep t_e , namely $h_{i,j}(t_e)$, is less than the total flux out of the cell times the timestep divided by the cell area. Using (equation 5.22) this condition is

$$h_{i,j}(t_e) - rn_wdmin1 < \frac{\Delta t}{e_1 e_2} (zzflxp_{i,j}^{(m)} + zzflxn_{i,j}^{(m)}). \quad (5.27)$$

Rearranging (equation 5.27) we can obtain an expression for the maximum outward flux that can be allowed and still maintain the minimum wet depth:

$$\begin{aligned} zzflxp_{i,j}^{(m+1)} &= \left[(h_{i,j}(t_e) - rn_wdmin1 - rn_wdmin2) \frac{e_1 e_2}{\Delta t} \right. \\ &\quad \left. - zzflxn_{i,j}^{(m)} \right] \end{aligned} \quad (5.28)$$

Note a small tolerance (rn_wdmin2) has been introduced here [*Q: Why is this necessary/desirable?*]. Substituting from (equation 5.25) gives an expression for the coefficient needed to multiply the outward flux at this cell in order to avoid drying.

$$\begin{aligned} zcoef_{i,j}^{(m+1)} &= \left[(h_{i,j}(t_e) - rn_wdmin1 - rn_wdmin2) \frac{e_1 e_2}{\Delta t} \right. \\ &\quad \left. - zzflxn_{i,j}^{(m)} \right] \frac{1}{zzflxp_{i,j}^{(0)}} \end{aligned} \quad (5.29)$$

Only the outward flux components are altered but, of course, outward fluxes from one cell are inward fluxes to adjacent cells and the balance in these cells may need subsequent adjustment; hence the iterative nature of this scheme. Note, for example, that the flux across the “eastern” face of the (i, j) th cell is only updated at the $m + 1$ th iteration if that flux at the m th iteration is out of the (i, j) th cell. If that is the case then the flux across that face is into the $(i + 1, j)$ cell and that flux will not be updated by the calculation for the $(i + 1, j)$ th cell. In this sense the updates to the fluxes across the faces of the cells do not “compete” (they do not over-write each other) and one would expect the scheme to converge relatively quickly. The scheme is flux based so conserves mass. It also conserves constant tracers for the same reason that the directional limiter does.

Modification of surface pressure gradients (*dynhpg.F90*)

At “dry” points the water depth is usually close to rn_wdmin1 . If the topography is sloping at these points the sea-surface will have a similar slope and there will hence be very large horizontal pressure gradients at these points. The WAD modifies the magnitude but not the sign of the surface pressure gradients ($zhpi$ and $zhpj$) at such points by multiplying them by positive factors (zcp_x and zcp_y respectively) that lie between 0 and 1.

We describe how the scheme works for the “eastward” pressure gradient, $zhpi$, calculated at the (i, j) th u -point. The scheme uses the ht_wd depths and surface heights at the neighbouring $(i + 1, j)$ and (i, j) tracer points. zcp_x is calculated using two logicals variables, ll_tmp1 and ll_tmp2 which are evaluated for each grid column. The three possible combinations are illustrated in figure 5.3.

The first logical, ll_tmp1 , is set to true if and only if the water depth at both neighbouring points is greater than $rn_wdmin1 + rn_wdmin2$ and the minimum height of the sea surface at the two points is greater than the maximum

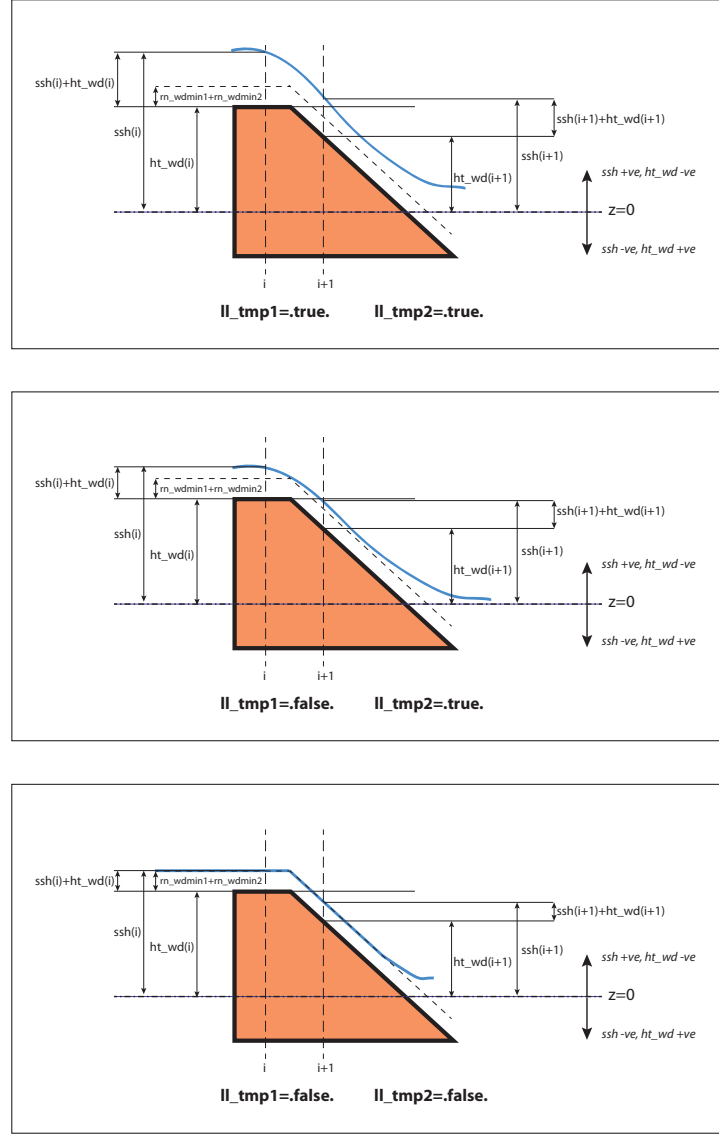


Figure 5.3.: Three possible combinations of the logical variables controlling the limiting of the horizontal pressure gradient in wetting and drying regimes

height of the topography at the two points:

$$\begin{aligned}
 ll_tmp1 = & \text{MIN}(ssh_n(j_i, j_j), ssh_n(j_i + 1, j_j)) > \\
 & \text{MAX}(-ht_wd(j_i, j_j), -ht_wd(j_i + 1, j_j)) \text{ .and.} \\
 & \text{MAX}(ssh_n(j_i, j_j) + ht_wd(j_i, j_j), \\
 & \quad ssh_n(j_i + 1, j_j) + ht_wd(j_i + 1, j_j)) > \\
 & rn_wdmin1 + rn_wdmin2
 \end{aligned} \tag{5.30}$$

The second logical, ll_tmp2 , is set to true if and only if the maximum height of the sea surface at the two points is greater than the maximum height of the topography at the two points plus $rn_wdmin1 + rn_wdmin2$

$$\begin{aligned}
 ll_tmp2 = & (\text{ABS}(ssh_n(j_i, j_j) - ssh_n(j_i + 1, j_j)) > 1.E - 12) \text{ .AND.} \\
 & (\text{MAX}(ssh_n(j_i, j_j), ssh_n(j_i + 1, j_j)) > \\
 & \quad \text{MAX}(-ht_wd(j_i, j_j), -ht_wd(j_i + 1, j_j)) + rn_wdmin1 + rn_wdmin2).
 \end{aligned} \tag{5.31}$$

If ll_tmp1 is true then the surface pressure gradient, $zphi$ at the (i, j) point is unmodified. If both logicals are false $zphi$ is set to zero.

If ll_tmp1 is true and ll_tmp2 is false then the surface pressure gradient is multiplied through by zcp_x which is the absolute value of the difference in the water depths at the two points divided by the difference in the surface heights at

the two points. Thus the sign of the sea surface height gradient is retained but the magnitude of the pressure force is determined by the difference in water depths rather than the difference in surface height between the two points. Note that dividing by the difference between the sea surface heights can be problematic if the heights approach parity. An additional condition is applied to `ll_tmp2` to ensure it is `.false.` in such conditions.

Additional considerations (`usrdef_zgr.F90`)

In the very shallow water where wetting and drying occurs the parametrisation of bottom drag is clearly very important. In order to promote stability it is sometimes useful to calculate the bottom drag using an implicit time-stepping approach.

Suitable specification of the surface heat flux in wetting and drying domains in forced and coupled simulations needs further consideration. In order to prevent freezing or boiling in uncoupled integrations the net surface heat fluxes need to be appropriately limited.

5.9.3. The WAD test cases (`usrdef_zgr.F90`)

See the WAD tests MY_DOC documentation for details of the WAD test cases.

5.10. Time evolution term (`dynnxt.F90`)

Options are defined through the `&namdom` (namelist 3.1) namelist variables. 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. chapter 2). The scheme is applied to the velocity, except when using the flux form of momentum advection (cf. section 5.3) in the variable volume case (`vv1?` defined), where it has to be applied to the thickness weighted velocity (see section A.3)

- vector invariant form or linear free surface (`ln_dynhpg_vec=.true.` ; `vv1?` not defined):

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

- flux form and nonlinear free surface (`ln_dynhpg_vec=.false.` ; `vv1?` defined):

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

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.e-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 `dynspflt.F90` module, and only array swapping and Asselin filtering is done in `dynnxt.F90`.

Surface Boundary Condition (SBC, SAS, ISF, ICB)

Table of contents

6.1.	Surface boundary condition for the ocean	68
6.2.	Input data generic interface	69
6.2.1.	Input data specification (<i>fldread.F90</i>)	69
6.2.2.	Interpolation on-the-fly	71
6.2.3.	Standalone surface boundary condition scheme (SAS)	72
6.3.	Flux formulation (<i>sbcflx.F90</i>)	73
6.4.	Bulk formulation (<i>sbcblk.F90</i>)	73
6.4.1.	Ocean-Atmosphere Bulk formulae (<i>sbcblk_algo_coare.F90, sbcblk_algo_coare3p5.F90, sbcblk_algo_ecmwf.F90, sbcblk_algo_ncar.F90</i>)	75
6.4.2.	Ice-Atmosphere Bulk formulae	75
6.5.	Coupled formulation (<i>sbcopl.F90</i>)	76
6.6.	Atmospheric pressure (<i>sbcapr.F90</i>)	77
6.7.	Surface tides (<i>sbctide.F90</i>)	77
6.8.	River runoffs (<i>sbcrrf.F90</i>)	78
6.9.	Ice shelf melting (<i>sbcisf.F90</i>)	79
6.10.	Ice sheet coupling	82
6.11.	Handling of icebergs (ICB)	83
6.12.	Interactions with waves (<i>sbcwave.F90, ln_wave</i>)	85
6.12.1.	Neutral drag coefficient from wave model (<i>ln_cdgw</i>)	86
6.12.2.	3D Stokes Drift (<i>ln_sdw & nn_sdrift</i>)	86
6.12.3.	Stokes-Coriolis term (<i>ln_stcor</i>)	87
6.12.4.	Wave modified stress (<i>ln_tauwoc & ln_tauw</i>)	87
6.13.	Miscellaneous options	87
6.13.1.	Diurnal cycle (<i>sbcscy.F90</i>)	87
6.13.2.	Rotation of vector pairs onto the model grid directions	87
6.13.3.	Surface restoring to observed SST and/or SSS (<i>sbcssr.F90</i>)	88
6.13.4.	Handling of ice-covered area (<i>sbcice_...</i>)	88
6.13.5.	Interface to CICE (<i>sbcice_ice.F90</i>)	89
6.13.6.	Freshwater budget control (<i>sbcfwb.F90</i>)	90

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

```

!-----
&namsbc      !   Surface Boundary Condition manager                (default: NO selection)
!-----
nn_fsbc      = 2          !   frequency of SBC module call
!              !   !   (control sea-ice & iceberg model call)
!              !   !   !   Type of air-sea fluxes
ln_usr       = .false.   !   user defined formulation            (T => check usrdef_sbc)
ln_flx       = .false.   !   flux formulation                (T => fill namsbc_flx )
ln_blk       = .false.   !   Bulk formulation                (T => fill namsbc_blk )
!              !   !   !   Type of coupling (Ocean/Ice/Atmosphere) :
ln_cpl       = .false.   !   atmosphere coupled formulation      ( requires key_oasis3 )
ln_mixcpl    = .false.   !   forced-coupled mixed formulation    ( requires key_oasis3 )
nn_components = 0        !   configuration of the opa-sas OASIS coupling
!              !   !   =0 no opa-sas OASIS coupling: default single executable config.
!              !   !   =1 opa-sas OASIS coupling: multi executable config., OPA component
!              !   !   =2 opa-sas OASIS coupling: multi executable config., SAS component
!              !   !   !   Sea-ice :
nn_ice       = 0          !   =0 no ice boundary condition
!              !   !   =1 use observed ice-cover                ( => fill namsbc_iif )
!              !   !   =2 or 3 automatically for SI3 or CICE      ("key_si3" or "key_cice")
!              !   !   !   except in AGRIF zoom where it has to be specified
ln_ice_embd  = .false.   !   =T embedded sea-ice (pressure + mass and salt exchanges)
!              !   !   =F levitating ice (no pressure, mass and salt exchanges)
!              !   !   !   Misc. options of sbc :
ln_traqsr    = .false.   !   Light penetration in the ocean      (T => fill namtra_qsr)
ln_dm2dc     = .false.   !   daily mean to diurnal cycle on short wave
ln_ssr       = .false.   !   Sea Surface Restoring on T and/or S  (T => fill namsbc_ssr)
nn_fwb       = 0          !   FreshWater Budget: =0 unchecked
!              !   !   =1 global mean of e-p-r set to zero at each time step
!              !   !   =2 annual global mean of e-p-r set to zero
ln_rnf       = .false.   !   runoffs                            (T => fill namsbc_rnf)
ln_apr_dyn   = .false.   !   Patm gradient added in ocean & ice Eqs. (T => fill namsbc_apr )
ln_isf       = .false.   !   ice shelf                          (T => fill namsbc_isf & namsbc_iscpl)
ln_wave      = .false.   !   Activate coupling with wave        (T => fill namsbc_wave)
ln_cdgw      = .false.   !   Neutral drag coefficient read from wave model (T => ln_wave=.true. & fill
↳ namsbc_wave)
ln_sdw       = .false.   !   Read 2D Surf Stokes Drift & Computation of 3D stokes drift (T => ln_wave=.true. &
↳ fill namsbc_wave)
nn_sdrift    = 0          !   Parameterization for the calculation of 3D-Stokes drift from the surface Stokes
↳ drift
!              !   !   = 0 Breivik 2015 parameterization: v_z=v_0*[exp(2*k*z)/(1-8*k*z)]
!              !   !   = 1 Phillips:
↳ v_z=v_o*[exp(2*k*z)-beta*sqrt(-2*k*pi*z)*erfc(sqrt(-2*k*z))]
!              !   !   = 2 Phillips as (1) but using the wave frequency from a wave model
ln_tauwoc    = .false.   !   Activate ocean stress modified by external wave induced stress (T => ln_wave=.true.
↳ & fill namsbc_wave)
ln_tauw      = .false.   !   Activate ocean stress components from wave model
ln_stcor     = .false.   !   Activate Stokes Coriolis term (T => ln_wave=.true. & ln_sdw=.true. & fill
↳ namsbc_wave)
nn_lsm       = 0          !   =0 land/sea mask for input fields is not applied (keep empty land/sea mask filename
↳ field) ,
!              !   !   =1:n number of iterations of land/sea mask application for input fields (fill
↳ land/sea mask filename field)
/

```

namelist 6.1.: &namsbc

The ocean needs seven fields as surface boundary condition:

- the two components of the surface ocean stress (τ_u , τ_v)
- the incoming solar and non solar heat fluxes (Q_{ns} , Q_{sr})
- the surface freshwater budget (emp)
- the surface salt flux associated with freezing/melting of seawater (sfx)
- the atmospheric pressure at the ocean surface (p_a)

Four different ways are available to provide the seven fields to the ocean. They are controlled by namelist &namsbc (namelist 6.1) variables:

- a bulk formulation (`ln_blk=.true.` with four possible bulk algorithms),
- a flux formulation (`ln_flx=.true.`),
- a coupled or mixed forced/coupled formulation (exchanges with a atmospheric model via the OASIS coupler), (`ln_cpl` or `ln_mixcpl=.true.`),

- a user defined formulation (`ln_usr=.true.`).

The frequency at which the forcing fields have to be updated is given by the `nn_fsbc` namelist parameter.

When the fields are supplied from data files (bulk, flux and mixed formulations), the input fields do not need to be supplied on the model grid. Instead, a file of coordinates and weights can be supplied to map the data from the input fields grid to the model points (so called "Interpolation on the Fly", see [subsection 6.2.2](#)). If the "Interpolation on the Fly" option is used, input data belonging to land points (in the native grid) should be masked or filled to avoid spurious results in proximity of the coasts, as large sea-land gradients characterize most of the atmospheric variables.

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 use of a land/sea mask for input fields (`nn_lsm=.true.`),
- 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 climatological ice-cover or a sea-ice model) (`nn_ice=0..3`),
- the addition of river runoffs as surface freshwater fluxes or lateral inflow (`ln_rnf=.true.`),
- the addition of ice-shelf melting as lateral inflow (parameterisation) or as fluxes applied at the land-ice ocean interface (`ln_isf=.true.`),
- the addition of a freshwater flux adjustment in order to avoid a mean sea-level drift (`nn_fwb=0..2`),
- the transformation of the solar radiation (if provided as daily mean) into an analytical diurnal cycle (`ln_dm2dc=.true.`),
- the activation of wave effects from an external wave model (`ln_wave=.true.`),
- a neutral drag coefficient is read from an external wave model (`ln_cdgw=.true.`),
- the Stokes drift from an external wave model is accounted for (`ln_sdw=.true.`),
- the choice of the Stokes drift profile parameterization (`nn_sdrift=0..2`),
- the surface stress given to the ocean is modified by surface waves (`ln_tauwoc=.true.`),
- the surface stress given to the ocean is read from an external wave model (`ln_tauw=.true.`),
- the Stokes-Coriolis term is included (`ln_stcor=.true.`),
- the light penetration in the ocean (`ln_traqsr=.true.` with namelist `&namtra_qsr` (namelist 4.3)),
- the atmospheric surface pressure gradient effect on ocean and ice dynamics (`ln_apr_dyn=.true.` with namelist `&namsbc_apr` (namelist 6.6)),
- the effect of sea-ice pressure on the ocean (`ln_ice_embd=.true.`).

In this chapter, we first discuss where the surface boundary conditions appear in the model equations. Then we present the three ways of providing the surface boundary conditions, 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. One of these is modification by icebergs (see [section 6.11](#)), which act as drifting sources of fresh water. Another example of modification is that due to the ice shelf melting/freezing (see [section 6.9](#)), which provides additional sources of fresh water.

6.1. Surface boundary condition for the ocean

The surface ocean stress is the stress exerted by the wind and the sea-ice on the ocean. It is applied in `dynzdf.F90` module as a surface boundary condition of the computation of the momentum vertical mixing trend (see [equation 5.20](#) in [section 5.7](#)). As such, it has to be provided as a 2D vector interpolated onto the horizontal velocity ocean mesh, *i.e.* resolved onto the model (*i,j*) direction at *u*- and *v*-points.

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 plus the heat content of the mass exchange between the ocean and sea-ice). It is applied in `trasbc.F90` module as a surface

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

Table 6.1.: Ocean variables provided to the surface module (SBC). The variable are averaged over `nn_fsbc` time-step, *i.e.* the frequency of computation of surface fluxes.

boundary condition trend of the first level temperature time evolution equation (see [equation 4.9](#) and [equation 4.10](#) in [subsection 4.4.1](#)). The latter is the penetrative part of the heat flux. It is applied as a 3D trend of the temperature equation (`traqsr.F90` module) when `ln_traqsr=.true.`. The way the light penetrates inside the water column is generally a sum of decreasing exponentials (see [subsection 4.4.2](#)).

The surface freshwater budget is provided by the `emp` field. It represents the mass flux exchanged with the atmosphere (evaporation minus precipitation) and possibly with the sea-ice and ice shelves (freezing minus melting of ice). It affects the ocean in two different ways: (*i*) it changes the volume of the ocean, and therefore appears in the sea surface height equation as a volume flux, and (*ii*) it changes the surface temperature and salinity through the heat and salt contents of the mass exchanged with atmosphere, sea-ice and ice shelves.

The ocean model provides, at each time step, to the surface module (`sbcmod.F90`) the surface currents, temperature and salinity. These variables are averaged over `nn_fsbc` time-step ([table 6.1](#)), and these averaged fields are used to compute the surface fluxes at the frequency of `nn_fsbc` time-steps.

6.2. 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 specified in *NEMO*. This task is achieved by `fldread.F90`. The module is designed with four main objectives in mind:

1. optionally provide a time interpolation of the input data every specified 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 informations.

As a result, the user has only to fill in for each variable a structure in the namelist file to define 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 three additional parameters for the 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 [subsection 6.2.1](#)), 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 [subsection 6.2.2](#)).

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 user can set the `cn_dir` to the pathway leading to the data. By default, the data are assumed to be in the same directory as the executable, so that `cn_dir='./`.

6.2.1. 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 !
↪ land/sea mask !
! ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' ! filename ! pairing !
↪ filename !
```

where

File name : the stem name of the NetCDF file to be opened. 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). [table 6.2](#) 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).

	daily or weekLL	monthly	yearly
<code>clim=.false.</code>	<code>fn_yYYYYmMMdDD.nc</code>	<code>fn_yYYYYmMM.nc</code>	<code>fn_yYYYY.nc</code>
<code>clim=.true.</code>	not possible	<code>fn_m??.nc</code>	<code>fn</code>

Table 6.2.: 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 '_N' is added to files, where 'N' is the child grid number.

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, setting 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 at the middle of the 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 exceeds the one of the file. See the above 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', 'pairing rotation' and 'land/sea mask' are associated with on-the-fly interpolation which is described in [subsection 6.2.2](#).

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 [Leclair and Madec \(2009\)](#), 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 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 readability 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 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.

6.2.2. 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(s), 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 such as CDO 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 interpolations. Prior to the interpolation, providing a land/sea mask file, the user can decide to remove land points from the input file and substitute the corresponding values with the average of the 8 neighbouring points in the native external grid. Only "sea points" are considered for the averaging. The land/sea mask file must be provided in the structure associated with the input variable. The netcdf land/sea mask variable name must be 'LSM' and must have the same horizontal and vertical dimensions as the associated variables and should be equal to 1 over land and 0 elsewhere. The procedure can be recursively applied by setting `nn_lsm > 1` in `namsrc` namelist. Note that `nn_lsm=0` forces the code to not apply the procedure, even if a land/sea mask file is supplied.

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 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:

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

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 included 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 to be not 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

```

!-----
&namcbc_sas      ! Stand-Alone Surface module: ocean data          (SAS_SRC  only)
!-----
l_sasread      = .true.      ! =T Read in file ; =F set all to 0. (see sbcssm)
ln_3d_uve      = .false.     ! specify whether we are supplying a 3D u,v and e3 field
ln_read_frq    = .false.     ! specify whether we must read frq or not

cn_dir         = './'        ! root directory for the ocean data location

!-----
! file name      ! frequency (hours) ! variable ! time interp.! clim ! 'yearly'/ !
! weights filename ! rotation ! land/sea mask !
! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
! pairing ! filename !
sn_ustp        = 'sas_grid_U' , , , 120. , 'uos' , .true. , .true. , 'yearly' ,
!-----
sn_vsp        = 'sas_grid_V' , , , 120. , 'vos' , .true. , .true. , 'yearly' ,
!-----
sn_tem        = 'sas_grid_T' , , , 120. , 'sosstst' , .true. , .true. , 'yearly' ,
!-----
sn_sal        = 'sas_grid_T' , , , 120. , 'sosaline' , .true. , .true. , 'yearly' ,
!-----
sn_ssh        = 'sas_grid_T' , , , 120. , 'sossheig' , .true. , .true. , 'yearly' ,
!-----
sn_e3t        = 'sas_grid_T' , , , 120. , 'e3t_m' , .true. , .true. , 'yearly' ,
!-----
sn_frq        = 'sas_grid_T' , , , 120. , 'frq_m' , .true. , .true. , 'yearly' ,
!-----
/

```

namelist 6.2.: &namcbc_sas

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 (irregular grid case) 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).

6.2.3. Standalone surface boundary condition scheme (SAS)

In some circumstances, it may be useful to avoid calculating the 3D temperature, salinity and velocity fields and simply read them in from a previous run or receive them from OASIS. For example:

- Multiple runs of the model are required in code development to see the effect of different algorithms in the bulk formulae.

- The effect of different parameter sets in the ice model is to be examined.
- Development of sea-ice algorithms or parameterizations.
- Spinup of the iceberg floats
- Ocean/sea-ice simulation with both models running in parallel (`ln_mixcpl=.true.`)

The Standalone Surface scheme provides this capacity. Its options are defined through the `&namsbc_sas` (namelist 6.2) namelist variables. A new copy of the model has to be compiled with a configuration based on ORCA2_SAS_LIM. However, no namelist parameters need be changed from the settings of the previous run (except perhaps `nn_date0`). In this configuration, a few routines in the standard model are overridden by new versions. Routines replaced are:

- `nemogcm.F90` : This routine initialises the rest of the model and repeatedly calls the stp time stepping routine (`step.F90`). Since the ocean state is not calculated all associated initialisations have been removed.
- `step.F90` : The main time stepping routine now only needs to call the sbc routine (and a few utility functions).
- `sbcmod.F90` : This has been cut down and now only calculates surface forcing and the ice model required. New surface modules that can function when only the surface level of the ocean state is defined can also be added (e.g. icebergs).
- `daymod.F90` : No ocean restarts are read or written (though the ice model restarts are retained), so calls to restart functions have been removed. This also means that the calendar cannot be controlled by time in a restart file, so the user must check that `nn_date0` in the model namelist is correct for his or her purposes.
- `stpctl.F90` : Since there is no free surface solver, references to it have been removed from `stp_ctl` module.
- `diawri.F90` : All 3D data have been removed from the output. The surface temperature, salinity and velocity components (which have been read in) are written along with relevant forcing and ice data.

One new routine has been added:

- `sbc_sas.F90` : This module initialises the input files needed for reading temperature, salinity and velocity arrays at the surface. These filenames are supplied in namelist `namsbc_sas`. Unfortunately, because of limitations with the `iom.F90` module, the full 3D fields from the mean files have to be read in and interpolated in time, before using just the top level. Since `fldread` is used to read in the data, Interpolation on the Fly may be used to change input data resolution.

The user can also choose in the `&namsbc_sas` (namelist 6.2) namelist to read the mean (`nn_fsbc` time-step) fraction of solar net radiation absorbed in the 1st T level using (`ln_flg=.true.`) and to provide 3D oceanic velocities instead of 2D ones (`ln_flg=.true.`). In that last case, only the 1st level will be read in.

6.3. Flux formulation (`sbcflx.F90`)

In the flux formulation (`ln_flg=.true.`), the surface boundary condition fields are directly read from input files. The user has to define in the namelist `&namsbc_flg` (namelist 6.3) 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 subsection 6.2.1 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 subsection 6.13.3 for its specification.

6.4. Bulk formulation (`sbcblk.F90`)

In the bulk formulation, the surface boundary condition fields are computed with bulk formulae using atmospheric fields and ocean (and sea-ice) variables averaged over `nn_fsbc` time-step.

The atmospheric fields used depend on the bulk formulae used. In forced mode, when a sea-ice model is used, a specific bulk formulation is used. Therefore, different bulk formulae are used for the turbulent fluxes computation over the ocean and over sea-ice surface. For the ocean, four bulk formulations are available thanks to the `Aerobulk` package (Brodeau et al. (2016)): the NCAR (formerly named CORE), COARE 3.0, COARE 3.5 and ECMWF bulk formulae. The choice is made by setting to true one of the following namelist variable: `ln_NCAR`, `ln_COARE_3p0`, `ln_COARE_3p5` and `ln_ECMWF`. For sea-ice, three possibilities can be selected: a constant transfer coefficient (1.4e-3; default value), Lüpkes et al. (2012) (`ln_Cd_L12`), and Lüpkes and Gryanik (2015) (`ln_Cd_L15`) parameterizations

```

!-----
&namsbc_flx ! surface boundary condition : flux formulation (ln_flx =T)
!-----
cn_dir      = './'      ! root directory for the fluxes data location

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

```

namelist 6.3.: &namsbc_flx

```

!-----
&namsbc_blk ! namsbc_blk generic Bulk formula (ln_blk =T)
!-----
! ! bulk algorithm :
ln_NCAR     = .false. ! "NCAR" algorithm (Large and Yeager 2008)
ln_COARE_3p0 = .false. ! "COARE 3.0" algorithm (Fairall et al. 2003)
ln_COARE_3p5 = .false. ! "COARE 3.5" algorithm (Edson et al. 2013)
ln_ECMWF    = .false. ! "ECMWF" algorithm (IFS cycle 31)
!-----
rn_zqt      = 10.      ! Air temperature & humidity reference height (m)
rn_zu       = 10.      ! Wind vector reference height (m)
ln_Cd_L12   = .false. ! air-ice drags = F(ice concentration) (Lupkes et al. 2012)
ln_Cd_L15   = .false. ! air-ice drags = F(ice concentration) (Lupkes et al. 2015)
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)
rn_efac     = 1.      ! multiplicative factor for evaporation (0. or 1.)
rn_vfac     = 0.      ! multiplicative factor for ocean & ice velocity used to
! ! calculate the wind stress (0.=absolute or 1.=relative winds)
!-----
cn_dir      = './'      ! root directory for the bulk data location

!-----
! ! file name ! frequency (hours) ! variable ! time interp.! clim ! 'yearly'/ !
! weights filename ! rotation ! land/sea mask !
! ! ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
! pairing ! filename !
sn_wndi     = 'u_10.15JUNE2009_fill' , 6. , 'U_10_MOD' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bicubic_noc.nc' , 'Uwnd' , '' , 'V_10_MOD' , .false. , .true. , 'yearly' , ''
sn_wndj     = 'v_10.15JUNE2009_fill' , 6. , 'V_10_MOD' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bicubic_noc.nc' , 'Vwnd' , '' , 'SWDN_MOD' , .false. , .true. , 'yearly' , ''
sn_qsr      = 'ncar_rad.15JUNE2009_fill' , 24. , 'SWDN_MOD' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bilinear_noc.nc' , '' , '' , 'LWDN_MOD' , .false. , .true. , 'yearly' , ''
sn_qlw      = 'ncar_rad.15JUNE2009_fill' , 24. , 'LWDN_MOD' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bilinear_noc.nc' , '' , '' , 'T_10_MOD' , .false. , .true. , 'yearly' , ''
sn_tair     = 't_10.15JUNE2009_fill' , 6. , 'T_10_MOD' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bilinear_noc.nc' , '' , '' , 'Q_10_MOD' , .false. , .true. , 'yearly' , ''
sn_humi     = 'q_10.15JUNE2009_fill' , 6. , 'Q_10_MOD' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bilinear_noc.nc' , '' , '' , 'PRC_MOD1' , .false. , .true. , 'yearly' , ''
sn_prec     = 'ncar_precip.15JUNE2009_fill' , -1. , 'PRC_MOD1' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bilinear_noc.nc' , '' , '' , 'SNOW' , .false. , .true. , 'yearly' , ''
sn_snow     = 'ncar_precip.15JUNE2009_fill' , -1. , 'SNOW' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bilinear_noc.nc' , '' , '' , 'SLP' , .false. , .true. , 'yearly' , ''
sn_slp      = 'slp.15JUNE2009_fill' , 6. , 'SLP' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bilinear_noc.nc' , '' , '' , 'taudif' , .false. , .true. , 'yearly' , ''
sn_tdif     = 'taudif_core' , 24. , 'taudif' , .false. , .true. , 'yearly' , ''
!-----
! weights_core_orca2_bilinear_noc.nc' , '' , '' , '' , '' , '' , '' , ''

```

namelist 6.4.: &namsbc_blk

Variable description	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
Mean sea-level pressure	slp	hPa	T

Common options are defined through the `&namsbc_blk` (namelist 6.4) namelist variables. The required 9 input fields are:

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.

The `sn_wndi`, `sn_wndj`, `sn_qsr`, `sn_qlw`, `sn_tair`, `sn_humi`, `sn_prec`, `sn_snow`, `sn_tdif` parameters describe the fields and the way they have to be used (spatial and temporal interpolations).

`cn_dir` is the directory of location of bulk files `ln_taudif` is the flag to specify if we use High Frequency (HF) tau information (.true.) or not (.false.) `rn_zqt` : is the height of humidity and temperature measurements (m) `rn_zu` : is the height of wind measurements (m)

Three multiplicative factors are available: `rn_pfac` and `rn_efac` allow to adjust (if necessary) the global freshwater budget by increasing/reducing the precipitations (total and snow) and or evaporation, respectively. The third one, `rn_vfac`, control to which extend the ice/ocean velocities are taken into account in the calculation of surface wind stress. Its range must be between zero and one, and it is recommended to set it to 0 at low-resolution (ORCA2 configuration).

As for the flux formulation, information about the input data required by the model is provided in the `namsbc_blk` namelist (see subsection 6.2.1).

6.4.1. Ocean-Atmosphere Bulk formulae (`sbcblk_algo_coare.F90`, `sbcblk_algo_coare3p5.F90`, `sbcblk_algo_ecmwf.F90`, `sbcblk_algo_ncar.F90`)

Four different bulk algorithms are available to compute surface turbulent momentum and heat fluxes over the ocean. COARE 3.0, COARE 3.5 and ECMWF schemes mainly differ by their roughness lengths computation and consequently their neutral transfer coefficients relationships with neutral wind.

- NCAR (`ln_NCAR=.true.`): The NCAR bulk formulae have been developed by Large and Yeager (2004). They have been designed to handle the NCAR 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 10m wind speed, air temperature and specific humidity. This Large and Yeager (2004) dataset is available through the GFDL web site. Note that substituting ERA40 to NCEP reanalysis fields does not require changes in the bulk formulae themselves. This is the so-called DRAKKAR Forcing Set (DFS) (Brodeau et al., 2010).
- COARE 3.0 (`ln_COARE_3p0=.true.`): See Fairall et al. (2003) for more details
- COARE 3.5 (`ln_COARE_3p5=.true.`): See Edson et al. (2013) for more details
- ECMWF (`ln_ECMWF=.true.`): Based on IFS (Cy31) implementation and documentation. Surface roughness lengths needed for the Obukhov length are computed following Beljaars (1995).

6.4.2. Ice-Atmosphere Bulk formulae

Surface turbulent fluxes between sea-ice and the atmosphere can be computed in three different ways:

- Constant value (`constant value Cd_ice=1.4e-3`): default constant value used for momentum and heat neutral transfer coefficients
- Lüpkes et al. (2012) (`ln_Cd_L12=.true.`): This scheme adds a dependency on edges at leads, melt ponds and flows of the constant neutral air-ice drag. After some approximations, this can be resumed to a dependency on ice concentration (A). This drag coefficient has a parabolic shape (as a function of ice concentration) starting at

```

!-----
&namcbc_cpl ! coupled ocean/atmosphere model ("key_oasis3")
!-----
nn_cplmodel = 1 ! Maximum number of models to/from which NEMO is potentially sending/receiving data
ln_usecplmask = .false. ! use a coupling mask file to merge data received from several models
! ! -> file cplmask.nc with the float variable called cplmask (jpi, jpj, nn_cplmodel)
nn_cats_cpl = 5 ! Number of sea ice categories over which coupling is to be carried out (if not 1)

!-----!
! ! description ! multiple ! vector ! vector ! vector !
! ! ! categories ! reference ! orientation ! grids !
!*** send ***
sn_snd_temp = 'weighted oce and ice' , 'no' , '' , '' , ''
sn_snd_alb = 'weighted ice' , 'no' , '' , '' , ''
sn_snd_thick = 'none' , 'no' , '' , '' , ''
sn_snd_crt = 'none' , 'no' , 'spherical' , 'eastward-northward' , 'T'
sn_snd_co2 = 'coupled' , 'no' , '' , '' , ''
sn_snd_crtw = 'none' , 'no' , '' , '' , ''
sn_snd_ifrac = 'none' , 'no' , '' , '' , ''
sn_snd_wlev = 'coupled' , 'no' , '' , '' , ''
sn_snd_cond = 'weighted ice' , 'no' , '' , '' , ''
sn_snd_thick1 = 'ice and snow' , 'no' , '' , '' , ''
sn_snd_mpnd = 'weighted ice' , 'no' , '' , '' , ''
sn_snd_sstfrz = 'coupled' , 'no' , '' , '' , ''
sn_snd_ttilyr = 'weighted ice' , 'no' , '' , '' , ''
!*** receive ***
sn_rcv_wl0m = 'none' , 'no' , '' , '' , ''
sn_rcv_tauomod = 'coupled' , 'no' , '' , '' , ''
sn_rcv_tau = 'oce only' , 'no' , 'cartesian' , 'eastward-northward' , 'U,V'
sn_rcv_dqnsdt = 'coupled' , 'no' , '' , '' , ''
sn_rcv_qsr = 'oce and ice' , 'no' , '' , '' , ''
sn_rcv_qns = 'oce and ice' , 'no' , '' , '' , ''
sn_rcv_emp = 'conservative' , 'no' , '' , '' , ''
sn_rcv_rnf = 'coupled' , 'no' , '' , '' , ''
sn_rcv_cal = 'coupled' , 'no' , '' , '' , ''
sn_rcv_co2 = 'coupled' , 'no' , '' , '' , ''
sn_rcv_hsig = 'none' , 'no' , '' , '' , ''
sn_rcv_iceflx = 'none' , 'no' , '' , '' , ''
sn_rcv_mslp = 'none' , 'no' , '' , '' , ''
sn_rcv_phioc = 'none' , 'no' , '' , '' , ''
sn_rcv_sdrfx = 'none' , 'no' , '' , '' , ''
sn_rcv_sdrfy = 'none' , 'no' , '' , '' , ''
sn_rcv_wper = 'none' , 'no' , '' , '' , ''
sn_rcv_wnum = 'none' , 'no' , '' , '' , ''
sn_rcv_wstrf = 'none' , 'no' , '' , '' , ''
sn_rcv_wdrag = 'none' , 'no' , '' , '' , ''
sn_rcv_ts_ice = 'none' , 'no' , '' , '' , ''
sn_rcv_isf = 'none' , 'no' , '' , '' , ''
sn_rcv_icb = 'none' , 'no' , '' , '' , ''
sn_rcv_tauwoc = 'none' , 'no' , '' , '' , ''
sn_rcv_tauw = 'none' , 'no' , '' , '' , ''
sn_rcv_wdrag = 'none' , 'no' , '' , '' , ''
/

```

namelist 6.5.: &namcbc_cpl

1.5e-3 for A=0, reaching 1.97e-3 for A=0.5 and going down 1.4e-3 for A=1. It is theoretically applicable to all ice conditions (not only MIZ).

- Lüpkes and Gryanik (2015) (`ln_Cd_L15=.true.`): Alternative turbulent transfer coefficients formulation between sea-ice and atmosphere with distinct momentum and heat coefficients depending on sea-ice concentration and atmospheric stability (no melt-ponds effect for now). The parameterization is adapted from ECHAM6 atmospheric model. Compared to Lupkes2012 scheme, it considers specific skin and form drags to compute neutral transfer coefficients for both heat and momentum fluxes. Atmospheric stability effect on transfer coefficient is also taken into account.

6.5. Coupled formulation (sbccpl.F90)

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 namelist, 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-MCT versions 1 to 4 (`key_oasis3`). An additional specific CPP key (`key_oa3mct_v1v2`) is needed for OASIS-3-MCT versions 1 and 2. It has been successfully used to interface *NEMO* to most of the European atmospheric GCM (ARPEGE, ECHAM, ECMWF, HadAM, HadGAM, LMDz), as well as to WRF (Weather Research and Forecasting Model).

```

!-----
&namsbc_apr      ! Atmospheric pressure used as ocean forcing          (ln_apr_dyn =T)
!-----
rn_pref          = 101000.    ! reference atmospheric pressure [N/m2]/
ln_ref_apr       = .false.    ! ref. pressure: global mean Patm (T) or a constant (F)
ln_apr_obc       = .false.    ! inverse barometer added to OBC ssh data

cn_dir = './'      ! root directory for the Patm data location

!-----
! file name          ! frequency (hours) ! variable ! time interp. ! clim ! 'yearly' / !
! weights filename ! rotation ! land/sea mask !
! pairing           ! filename          ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
sn_apr             = 'patm'           , -1. , 'soms1pre' , .true. , .true. , 'yearly' ,
!-----
/
    
```

namelist 6.6.: `&namsbc_apr`

When PISCES biogeochemical model (`key_top`) is also used in the coupled system, the whole carbon cycle is computed. In this case, CO₂ fluxes will be exchanged between the atmosphere and the ice-ocean system (and need to be activated in `&namsbc_cpl` (namelist 6.5)).

The namelist above allows control of various aspects of the coupling fields (particularly for vectors) and now allows for any coupling fields to have multiple sea ice categories (as required by LIM3 and CICE). When indicating a multi-category coupling field in `&namsbc_cpl` (namelist 6.5), the number of categories will be determined by the number used in the sea ice model. In some limited cases, it may be possible to specify single category coupling fields even when the sea ice model is running with multiple categories - in this case, the user should examine the code to be sure the assumptions made are satisfactory. In cases where this is definitely not possible, the model should abort with an error message.

6.6. Atmospheric pressure (`sbcapr.F90`)

The optional atmospheric pressure can be used to force ocean and ice dynamics (`ln_apr_dyn=.true.` , `&namsbc` (namelist 6.1) namelist). The input atmospheric forcing defined via `sn_apr` structure (`&namsbc_apr` (namelist 6.6) 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:

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

where P_{atm} is the atmospheric pressure and P_o a reference atmospheric pressure. A value of 101,000 N/m² 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 steps.

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 written in the output. This can simplify altimetry data and model comparison as inverse barometer sea surface height is usually removed from these data prior to their distribution.

When using time-splitting and BDY package for open boundaries conditions, the equivalent inverse barometer sea surface height η_{ib} can be added to BDY ssh data: `ln_apr_obc` might be set to true.

6.7. Surface tides (`sbctide.F90`)

The tidal forcing, generated by the gravity forces of the Earth-Moon and Earth-Sun systems, is activated if `ln_tide` and `ln_tide_pot` are both set to `.true.` in `&nam_tide` (namelist 6.7). This translates as an additional barotropic force in the momentum equation 1.4a such that:

$$\frac{\partial U_h}{\partial t} = \dots + g\nabla(\Pi_{eq} + \Pi_{sal})$$

where Π_{eq} stands for the equilibrium tidal forcing and Π_{sal} is a self-attraction and loading term (SAL).

The equilibrium tidal forcing is expressed as a sum over a subset of constituents chosen from the set of available tidal constituents defined in file `SBC/tide.h90` (this comprises the tidal constituents *M2*, *N2*, *2N2*, *S2*, *K2*, *K1*, *O1*, *Q1*, *P1*, *M4*, *Mf*, *Mm*, *Msqm*, *Mtm*, *S1*, *MU2*, *NU2*, *L2*, and *T2*). Individual constituents are selected by including their names in the array `clname` in `&nam_tide` (namelist 6.7) (e.g., `clname (1)='M2'`, `clname (2)='S2'` to select solely the

```

!-----
&nam_tide      !  tide parameters                                (default: OFF)
!-----
ln_tide       = .false.    ! Activate tides
ln_tide_pot   = .true.     ! use tidal potential forcing
ln_scal_load  = .false.    ! Use scalar approximation for
rn_scal_load  = 0.094      ! load potential
ln_read_load  = .false.    ! Or read load potential from file
cn_tide_load  = 'tide_LOAD_grid_T.nc' ! filename for load potential
!
ln_tide_ramp  = .false.    ! Use linear ramp for tides at startup
rdttideramp  = 0.         ! ramp duration in days
cname(1)     = 'DUMMY'     ! name of constituent - all tidal components must be set in
↳ namelist_cfg
/

```

namelist 6.7.: &nam_tide

```

!-----
&namsbc_rnf   !  runoffs                                        (ln_rnf =T)
!-----
ln_rnf_mouth  = .false.    ! specific treatment at rivers mouths
rn_hrnf       = 15.e0      ! depth over which enhanced vertical mixing is used (ln_rnf_mouth=T)
rn_avt_rnf    = 1.e-3     ! value of the additional vertical mixing coef. [m2/s] (ln_rnf_mouth=T)
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
ln_rnf_depth_ini = .false. ! compute depth at initialisation from runoff file
rn_rnf_max    = 5.735e-4   ! max value of the runoff climatologie over global domain ( ln_rnf_depth_ini =
↳ .true )
rn_dep_max    = 150.       ! depth over which runoffs is spread ( ln_rnf_depth_ini = .true )
nn_rnf_depth_file = 0     ! create (=1) a runoff depth file or not (=0)

cn_dir        = './'      ! root directory for the runoff data location

↳ !-----!
!      ! file name      ! frequency (hours) ! variable ! time interp.! clim ! 'yearly'/ !
↳ weights filename ! rotation ! land/sea mask !
!      !      !      ! (if <0 months) ! name      ! (logical) ! (T/F) ! 'monthly' !
↳ ! pairing ! filename !
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' , ''
↳ , ''
/

```

namelist 6.8.: &namsbc_rnf

tidal constituents $M2$ and $S2$). Optionally, when `ln_tide_ramp` is set to `.true.`, the equilibrium tidal forcing can be ramped up linearly from zero during the initial `rdttideramp` days of the model run.

The SAL term should in principle be computed online as it depends on the model tidal prediction itself (see [Arbic et al. \(2004\)](#) for a discussion about the practical implementation of this term). Nevertheless, the complex calculations involved would make this computationally too expensive. Here, two options are available: Π_{sal} generated by an external model can be read in (`ln_read_load=.true.`), or a “scalar approximation” can be used (`ln_scal_load=.true.`). In the latter case

$$\Pi_{sal} = \beta \eta,$$

where β (`rn_scal_load` with a default value of 0.094) is a spatially constant scalar, often chosen to minimize tidal prediction errors. Setting both `ln_read_load` and `ln_scal_load` to `.false.` removes the SAL contribution.

6.8. River runoffs (*sbc_rnf.F90*)

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*.

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 variables in `&namsbc_rnf` (namelist 6.8), `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 `sbc_rnf`. For temperature -999 is taken as missing data and the river temperature is taken to be the surface temperature at the river point. For the depth parameter a value of -1 means the river is added to the surface box only, and a value of -999 means the river is added through the entire water column. 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 in, 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 `divhor.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, 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 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 redistributed 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 runoff 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.

6.9. Ice shelf melting (`sbcisf.F90`)

The namelist variable in `&namsbc` (namelist 6.1), `nn_isf`, controls the ice shelf representation. Description and result of sensitivity test to `nn_isf` are presented in Mathiot et al. (2017). The different options are illustrated in figure 6.1.

`nn_isf=1` : The ice shelf cavity is represented (`ln_isfcav=.true.` needed). The fwf and heat flux are depending of the local water properties.

Two different bulk formulae are available:

`nn_isfblk=1` : The melt rate is based on a balance between the upward ocean heat flux and the latent heat flux at the ice shelf base. A complete description is available in Hunter (2006).

`nn_isfblk=2` : The melt rate and the heat flux are based on a 3 equations formulation (a heat flux budget at the ice base, a salt flux budget at the ice base and a linearised freezing point temperature equation). A complete description is available in Jenkins (1991).

*At least a top cells thickness of 1 meter and a 3 hours forcing frequency are required to properly represent the diurnal cycle (Bernie et al., 2005). see also figure 6.3.

```

!-----
&namsbc_isf      ! Top boundary layer (ISF)                                (ln_isfcav =T : read (ln_read_cfg=T)
!-----                                                or set or usr_def_zgr )
!
!           ! type of top boundary layer
nn_isf        = 1           ! ice shelf melting/freezing
                        ! 1 = presence of ISF ; 2 = bg03 parametrisation
                        ! 3 = rnf file for ISF ; 4 = ISF specified freshwater flux
                        ! options 1 and 4 need ln_isfcav = .true. (domzgr)
!
!           ! nn_isf = 1 or 2 cases:
rn_gammat0    = 1.e-4      ! gammat coefficient used in blk formula
rn_gammas0    = 1.e-4      ! gammas coefficient used in blk formula
!
!           ! nn_isf = 1 or 4 cases:
rn_hisf_tbl   = 30.        ! thickness of the top boundary layer (Losh et al. 2008)
!
!           ! 0 => thickness of the tbl = thickness of the first wet cell
!
!           ! nn_isf = 1 case
nn_isfblk     = 1          ! 1 ISOMIP like: 2 equations formulation (Hunter et al., 2006)
!
!           ! 2 ISOMIP+ like: 3 equations formulation (Asay-Davis et al., 2015)
nn_gammatblk  = 1          ! 0 = cst Gammat (= gammat/s)
!
!           ! 1 = velocity dependend Gamma (u* * gammat/s) (Jenkins et al. 2010)
!
!           ! 2 = velocity and stability dependend Gamma (Holland et al. 1999)
!
!-----
!
!           ! file name ! frequency (hours) ! variable ! time interp.! clim ! 'yearly'/ ! weights !
!
!           ! rotation ! land/sea mask !
!
!           ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' ! filename !
!
!           ! pairing ! filename !
!
! * nn_isf = 4 case
sn_fwfisf    = 'rnfisf' , -12. , 'sowflisf' , .false. , .true. , 'yearly' , '' ,
!
! * nn_isf = 3 case
sn_rnfisf    = 'rnfisf' , -12. , 'sofwfisf' , .false. , .true. , 'yearly' , '' ,
!
! * nn_isf = 2 and 3 cases
sn_depmax_isf = 'rnfisf' , -12. , 'sozsisfmax' , .false. , .true. , 'yearly' , '' ,
!
! * nn_isf = 2 case
sn_depmin_isf = 'rnfisf' , -12. , 'sozsisfmin' , .false. , .true. , 'yearly' , '' ,
!
! * nn_isf = 2 case
sn_Leff_isf  = 'rnfisf' , -12. , 'Leff' , .false. , .true. , 'yearly' , '' ,
!
/

```

namelist 6.9.: &namsbc_isf

Temperature and salinity used to compute the melt are the average temperature in the top boundary layer [Losch \(2008\)](#). Its thickness is defined by `rn_hisf_tbl`. The fluxes and friction velocity are computed using the mean temperature, salinity and velocity in the the first `rn_hisf_tbl` m. Then, the fluxes are spread over the same thickness (ie over one or several cells). If `rn_hisf_tbl` larger than top e_3t , there is no more feedback between the freezing point at the interface and the the top cell temperature. This can lead to super-cool temperature in the top cell under melting condition. If `rn_hisf_tbl` smaller than top e_3t , the top boundary layer thickness is set to the top cell thickness.

Each melt bulk formula depends on a exchange coefficient ($\Gamma^{T,S}$) between the ocean and the ice. There are 3 different ways to compute the exchange coefficient:

`nn_gammabl=0` : The salt and heat exchange coefficients are constant and defined by `rn_gammas0` and `rn_gammat0`.

$$\begin{aligned}\gamma^T &= rn_gammat0 \\ \gamma^S &= rn_gammas0\end{aligned}$$

This is the recommended formulation for ISOMIP.

`nn_gammabl=1` : The salt and heat exchange coefficients are velocity dependent and defined as

$$\begin{aligned}\gamma^T &= rn_gammat0 \times u_* \\ \gamma^S &= rn_gammas0 \times u_*\end{aligned}$$

where u_* is the friction velocity in the top boundary layer (ie first `rn_hisf_tbl` meters). See [Jenkins et al. \(2010\)](#) for all the details on this formulation. It is the recommended formulation for realistic application.

`nn_gammabl=2` : The salt and heat exchange coefficients are velocity and stability dependent and defined as:

$$\gamma^{T,S} = \frac{u_*}{\Gamma_{Turb} + \Gamma_{Mole}^{T,S}}$$

where u_* is the friction velocity in the top boundary layer (ie first `rn_hisf_tbl` meters), Γ_{Turb} the contribution of the ocean stability and $\Gamma_{Mole}^{T,S}$ the contribution of the molecular diffusion. See [Holland and Jenkins \(1999\)](#) for all the details on this formulation. This formulation has not been extensively tested in *NEMO* (not recommended).

`nn_isf=2` : The ice shelf cavity is not represented. The fwf and heat flux are computed using the [Beckmann and Goosse \(2003\)](#) parameterisation of isf melting. The fluxes are distributed along the ice shelf edge between the depth of the average grounding line (GL) (`sn_depmax_isf`) and the base of the ice shelf along the calving front (`sn_depmin_isf`) as in (`nn_isf=3`). The effective melting length (`sn_Leff_isf`) is read from a file.

`nn_isf=3` : The ice shelf cavity is not represented. The fwf (`sn_rnfisf`) is prescribed and distributed along the ice shelf edge between the depth of the average grounding line (GL) (`sn_depmax_isf`) and the base of the ice shelf along the calving front (`sn_depmin_isf`). The heat flux (Q_h) is computed as $Q_h = fwf \times L_f$.

`nn_isf=4` : The ice shelf cavity is opened (`ln_isfcav=true`. needed). However, the fwf is not computed but specified from file `sn_fwfisf`). The heat flux (Q_h) is computed as $Q_h = fwf \times L_f$. As in `nn_isf=1`, the fluxes are spread over the top boundary layer thickness (`rn_hisf_tbl`)

- `nn_isf=1` and `nn_isf=2` compute a melt rate based on the water mass properties, ocean velocities and depth. This flux is thus highly dependent of the model resolution (horizontal and vertical), realism of the water masses onto the shelf ...

- `nn_isf=3` and `nn_isf=4` read the melt rate from a file. You have total control of the fwf forcing. This can be useful if the water masses on the shelf are not realistic or the resolution (horizontal/vertical) are too coarse to have realistic melting or for studies where you need to control your heat and fw input.

The ice shelf melt is implemented as a volume flux as for the runoff. The fw addition due to the ice shelf melting is, at each relevant depth level, added to the horizontal divergence (*hdivn*) in the subroutine `sbc_isf_div`, called from `divhor.F90`. See the runoff section [section 6.8](#) for all the details about the divergence correction.

Figure 6.1.: Illustration of the location where the fwf is injected and whether or not the fwf is interactif or not depending of `nn_isf`.

6.10. Ice sheet coupling

Ice sheet/ocean coupling is done through file exchange at the restart step. At each restart step:

1. the ice sheet model send a new bathymetry and ice shelf draft netcdf file.
2. a new domcfg.nc file is built using the DOMAINcfg tools.
3. *NEMO* run for a specific period and output the average melt rate over the period.
4. the ice sheet model run using the melt rate outputed in step 4.
5. go back to 1.

If `ln_iscpl=.true.`, the isf draft is assume to be different at each restart step with potentially some new wet/dry cells due to the ice sheet dynamics/thermodynamics. The wetting and drying scheme applied on the restart is very simple and described below for the 6 different possible cases:

Thin a cell down : T/S/ssh are unchanged and U/V in the top cell are corrected to keep the barotropic transport (bt) constant ($bt_b = bt_n$).

Enlarge a cell : See case "Thin a cell down"

Dry a cell : mask, T/S, U/V and ssh are set to 0. Furthermore, U/V into the water column are modified to satisfy ($bt_b = bt_n$).

```

!-----
&namcbc_iscpl ! land ice / ocean coupling option (ln_isfcav =T : read (ln_read_cfg=T)
!----- or set or usr_def_zgr )
nn_drown      = 10      ! number of iteration of the extrapolation loop (fill the new wet cells)
ln_hsb        = .false. ! activate conservation module (conservation exact after a time of rn_fiscpl)
nn_fiscpl     = 43800   ! (number of time step) conservation period (maybe should be fix to the coupling
↪ frequency of restart frequency)
/

```

namelist 6.10.: &namcbc_iscpl

Wet a cell : mask is set to 1, T/S is extrapolated from neighbours, $ssh_n = ssh_b$ and U/V set to 0. If no neighbours, T/S is extrapolated from old top cell value. If no neighbours along i,j and k (both previous test failed), T/S/U/V/ssh and mask are set to 0.

Dry a column : mask, T/S, U/V are set to 0 everywhere in the column and ssh set to 0.

Wet a column : set mask to 1, T/S is extrapolated from neighbours, ssh is extrapolated from neighbours and U/V set to 0. If no neighbour, T/S/U/V and mask set to 0.

Furthermore, as the before and now fields are not compatible (modification of the geometry), the restart time step is prescribed to be an euler time step instead of a leap frog and $fields_b = fields_n$.

The horizontal extrapolation to fill new cell with realistic value is called `nn_drown` times. It means that if the grounding line retreat by more than `nn_drown` cells between 2 coupling steps, the code will be unable to fill all the new wet cells properly. The default number is set up for the MISOMIP idealised experiments. This coupling procedure is able to take into account grounding line and calving front migration. However, it is a non-conservative process. This could lead to a trend in heat/salt content and volume.

In order to remove the trend and keep the conservation level as close to 0 as possible, a simple conservation scheme is available with `ln_hsb=.true.`. The heat/salt/vol. gain/loss is diagnosed, as well as the location. A correction increment is computed and apply each time step during the next `rn_fiscpl` time steps. For safety, it is advised to set `rn_fiscpl` equal to the coupling period (smallest increment possible). The corrective increment is apply into the cell itself (if it is a wet cell), the neighbouring cells or the closest wet cell (if the cell is now dry).

6.11. Handling of icebergs (ICB)

Icebergs are modelled as lagrangian particles in *NEMO* (Marsh et al., 2015). Their physical behaviour is controlled by equations as described in Martin and Adcroft (2010). (Note that the authors kindly provided a copy of their code to act as a basis for implementation in *NEMO*). Icebergs are initially spawned into one of ten classes which have specific mass and thickness as described in the `&namberg` (namelist 6.11) namelist: `rn_initial_mass` and `rn_initial_thickness`. Each class has an associated scaling (`rn_mass_scaling`), which is an integer representing how many icebergs of this class are being described as one lagrangian point (this reduces the numerical problem of tracking every single iceberg). They are enabled by setting `ln_icebergs=.true.`

Two initialisation schemes are possible.

`nn_test_icebergs > 0` In this scheme, the value of `nn_test_icebergs` represents the class of iceberg to generate (so between 1 and 10), and `nn_test_icebergs` provides a lon/lat box in the domain at each grid point of which an iceberg is generated at the beginning of the run. (Note that this happens each time the timestep equals `nn_nit000`.) `nn_test_icebergs` is defined by four numbers in `nn_test_box` representing the corners of the geographical box: lonmin,lonmax,latmin,latmax

`nn_test_icebergs=-1` In this scheme, the model reads a calving file supplied in the `sn_icb` parameter. This should be a file with a field on the configuration grid (typically ORCA) representing ice accumulation rate at each model point. These should be ocean points adjacent to land where icebergs are known to calve. Most points in this input grid are going to have value zero. When the model runs, ice is accumulated at each grid point which has a non-zero source term. At each time step, a test is performed to see if there is enough ice mass to calve an iceberg of each class in order (1 to 10). Note that this is the initial mass multiplied by the number each particle represents (*i.e.* the scaling). If there is enough ice, a new iceberg is spawned and the total available ice reduced accordingly.

Icebergs are influenced by wind, waves and currents, bottom melt and erosion. The latter act to disintegrate the iceberg. This is either all melted freshwater, or (if `rn_bits_erosion_fraction > 0`) into melt and additionally small ice

```

!-----
&namberg      !   iceberg parameters                                     (default: OFF)
!-----
ln_icebergs = .false.      ! activate iceberg floats (force =F with "key_agrif")
!
!
! diagnostics:
ln_bergdia    = .true.     ! Calculate budgets
nn_verbose_level = 0       ! Turn on more verbose output if level > 0
nn_verbose_write = 15      ! Timesteps between verbose messages
nn_sample_rate = 1         ! Timesteps between sampling for trajectory storage
!
!
! iceberg setting:
!
! Initial mass required for an iceberg of each class
rn_initial_mass = 8.8e7, 4.1e8, 3.3e9, 1.8e10, 3.8e10, 7.5e10, 1.2e11, 2.2e11, 3.9e11, 7.4e11
!
! Proportion of calving mass to apportion to each class
rn_distribution = 0.24, 0.12, 0.15, 0.18, 0.12, 0.07, 0.03, 0.03, 0.03, 0.02
!
! Ratio between effective and real iceberg mass (non-dim)
!
! i.e. number of icebergs represented at a point
rn_mass_scaling = 2000., 200., 50., 20., 10., 5., 2., 1., 1., 1.
!
! thickness of newly calved bergs (m)
rn_initial_thickness = 40., 67., 133., 175., 250., 250., 250., 250., 250., 250.
!
!
rn_rho_bergs    = 850.     ! Density of icebergs
rn_LoW_ratio    = 1.5      ! Initial ratio L/W for newly calved icebergs
ln_operator_splitting = .true. ! Use first order operator splitting for thermodynamics
rn_bits_erosion_fraction = 0. ! Fraction of erosion melt flux to divert to bergy bits
rn_sicn_shift   = 0.       ! Shift of sea-ice concn in erosion flux (0<sicn_shift<1)
ln_passive_mode = .false.  ! iceberg - ocean decoupling
nn_test_icebergs = 10      ! Create test icebergs of this class (-1 = no)
!
! Put a test iceberg at each gridpoint in box (lon1,lon2,lat1,lat2)
rn_test_box     = 108.0, 116.0, -66.0, -58.0
ln_use_calving  = .false.  ! Use calving data even when nn_test_icebergs > 0
rn_speed_limit  = 0.       ! CFL speed limit for a berg

cn_dir         = './'      ! root directory for the calving data location

↔ !-----!-----!-----!-----!-----!-----!-----!-----!-----!-----!
! ! file name ! frequency (hours) ! variable ! time interp.! clim ! 'yearly' / !
↔ weights filename ! rotation ! land/sea mask !
! ! ! ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
↔ ! pairing ! filename !
sn_icb = 'calving' , -1. , 'calvingmask', .true. , .true. , 'yearly' , ''
↔ , '' , ''
/

```

namelist 6.11.: &namberg

```

!-----
&namsbc_wave ! External fields from wave model (ln_wave=T)
!-----
cn_dir = './' ! root directory for the waves data location

!-----!-----!-----!-----!-----!-----!-----!-----!-----!
! file name ! frequency (hours) ! variable ! time interp. ! clim ! 'yearly' / !
! weights filename ! rotation ! land/sea mask !
! ! ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
! pairing ! filename !
sn_cdg = 'sdw_ecwaves_orca2' , 6. , 'drag_coeff' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
sn_usd = 'sdw_ecwaves_orca2' , 6. , 'u_sd2d' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
sn_vsd = 'sdw_ecwaves_orca2' , 6. , 'v_sd2d' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
sn_hsw = 'sdw_ecwaves_orca2' , 6. , 'hs' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
sn_wmp = 'sdw_ecwaves_orca2' , 6. , 'wmp' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
sn_wfr = 'sdw_ecwaves_orca2' , 6. , 'wfr' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
sn_wnum = 'sdw_ecwaves_orca2' , 6. , 'wave_num' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
sn_tauwoc = 'sdw_ecwaves_orca2' , 6. , 'wave_stress' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
sn_tauwx = 'sdw_ecwaves_orca2' , 6. , 'wave_stress' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
sn_tauwy = 'sdw_ecwaves_orca2' , 6. , 'wave_stress' , .true. , .true. , 'yearly' , ''
!-----!-----!-----!-----!-----!-----!-----!-----!-----!
/

```

namelist 6.12.: &namsbc_wave

bits which are assumed to propagate with their larger parent and thus delay fluxing into the ocean. Melt water (and other variables on the configuration grid) are written into the main *NEMO* model output files.

Extensive diagnostics can be produced. Separate output files are maintained for human-readable iceberg information. A separate file is produced for each processor (independent of `ln_ctl1`). The amount of information is controlled by two integer parameters:

`nn_verbose_level` takes a value between one and four and represents an increasing number of points in the code at which variables are written, and an increasing level of obscurity.

`nn_verbose_write` is the number of timesteps between writes

Iceberg trajectories can also be written out and this is enabled by setting `nn_sample_rate > 0`. A non-zero value represents how many timesteps between writes of information into the output file. These output files are in NETCDF format. When `key_mpp_mpi` is defined, each output file contains only those icebergs in the corresponding processor. Trajectory points are written out in the order of their parent iceberg in the model's "linked list" of icebergs. So care is needed to recreate data for individual icebergs, since its trajectory data may be spread across multiple files.

6.12. Interactions with waves (*sbcwave.F90* , `ln_wave`)

Ocean waves represent the interface between the ocean and the atmosphere, so *NEMO* is extended to incorporate physical processes related to ocean surface waves, namely the surface stress modified by growth and dissipation of the oceanic wave field, the Stokes-Coriolis force and the Stokes drift impact on mass and tracer advection; moreover the neutral surface drag coefficient from a wave model can be used to evaluate the wind stress.

Physical processes related to ocean surface waves can be accounted by setting the logical variable `ln_wave=.true.` in `&namsbc` (namelist 6.1) namelist. In addition, specific flags accounting for different processes should be activated as explained in the following sections.

Wave fields can be provided either in forced or coupled mode:

forced mode : wave fields should be defined through the `&namsbc_wave` (namelist 6.12) namelist for external data names, locations, frequency, interpolation and all the miscellaneous options allowed by Input Data generic Interface (see section 6.2).

coupled mode : *NEMO* and an external wave model can be coupled by setting `ln_cpl=.true.` in `&namsbc` (namelist 6.1) namelist and filling the `&namsbc_cpl` (namelist 6.5) namelist.

6.12.1. Neutral drag coefficient from wave model (ln_cdgw)

The neutral surface drag coefficient provided from an external data source (*i.e.* a wave model), can be used by setting the logical variable `ln_cdgw=.true.` in `&nam_sbc` (namelist 6.1) namelist. Then using the routine `sbcblk_algo_ncar` and starting from the neutral drag coefficient provided, the drag coefficient is computed according to the stable/unstable conditions of the air-sea interface following [Large and Yeager \(2004\)](#).

6.12.2. 3D Stokes Drift (ln_sdw & nn_sdrift)

The Stokes drift is a wave driven mechanism of mass and momentum transport ([Stokes, 2009](#)). It is defined as the difference between the average velocity of a fluid parcel (Lagrangian velocity) and the current measured at a fixed point (Eulerian velocity). As waves travel, the water particles that make up the waves travel in orbital motions but without a closed path. Their movement is enhanced at the top of the orbit and slowed slightly at the bottom, so the result is a net forward motion of water particles, referred to as the Stokes drift. An accurate evaluation of the Stokes drift and the inclusion of related processes may lead to improved representation of surface physics in ocean general circulation models. The Stokes drift velocity U_{st} in deep water can be computed from the wave spectrum and may be written as:

$$U_{st} = \frac{16\pi^3}{g} \int_0^\infty \int_{-\pi}^\pi (\cos\theta, \sin\theta) f^3 S(f, \theta) e^{2kz} d\theta df$$

where: θ is the wave direction, f is the wave intrinsic frequency, $S(f, \theta)$ is the 2D frequency-direction spectrum, k is the mean wavenumber defined as: $k = \frac{2\pi}{\lambda}$ (being λ the wavelength).

In order to evaluate the Stokes drift in a realistic ocean wave field, the wave spectral shape is required and its computation quickly becomes expensive as the 2D spectrum must be integrated for each vertical level. To simplify, it is customary to use approximations to the full Stokes profile. Three possible parameterizations for the calculation for the approximate Stokes drift velocity profile are included in the code through the `nn_sdrift` parameter once provided the surface Stokes drift $U_{st|z=0}$ which is evaluated by an external wave model that accurately reproduces the wave spectra and makes possible the estimation of the surface Stokes drift for random directional waves in realistic wave conditions:

nn_sdrift = 0 : exponential integral profile parameterization proposed by [Breivik et al. \(2014\)](#):

$$U_{st} \cong U_{st|z=0} \frac{e^{-2k_e z}}{1 - 8k_e z}$$

where k_e is the effective wave number which depends on the Stokes transport T_{st} defined as follows:

$$k_e = \frac{|U_{st|z=0}|}{|T_{st}|} \quad \text{and} \quad T_{st} = \frac{1}{16} \bar{\omega} H_s^2$$

where H_s is the significant wave height and ω is the wave frequency.

nn_sdrift = 1 : velocity profile based on the Phillips spectrum which is considered to be a reasonable estimate of the part of the spectrum mostly contributing to the Stokes drift velocity near the surface ([Breivik et al., 2016](#)):

$$U_{st} \cong U_{st|z=0} \left[\exp(2k_p z) - \beta \sqrt{-2\pi k_p z} \operatorname{erf} \left(\sqrt{-2k_p z} \right) \right]$$

where erf is the complementary error function and k_p is the peak wavenumber.

nn_sdrift = 2 : velocity profile based on the Phillips spectrum as for `nn_sdrift = 1` but using the wave frequency from a wave model.

The Stokes drift enters the wave-averaged momentum equation, as well as the tracer advection equations and its effect on the evolution of the sea-surface height η is considered as follows:

$$\frac{\partial \eta}{\partial t} = -\nabla_h \int_{-H}^{\eta} (U + U_{st}) dz$$

The tracer advection equation is also modified in order for Eulerian ocean models to properly account for unresolved wave effect. The divergence of the wave tracer flux equals the mean tracer advection that is induced by the three-dimensional Stokes velocity. The advective equation for a tracer c combining the effects of the mean current and sea surface waves can be formulated as follows:

$$\frac{\partial c}{\partial t} = -(U + U_{st}) \cdot \nabla c$$

6.12.3. Stokes-Coriolis term (`ln_stcor`)

In a rotating ocean, waves exert a wave-induced stress on the mean ocean circulation which results in a force equal to $U_{st}\mathcal{C}f$, where f is the Coriolis parameter. This additional force may have impact on the Ekman turning of the surface current. In order to include this term, once evaluated the Stokes drift (using one of the 3 possible approximations described in subsection 6.12.2), `ln_stcor=.true.` has to be set.

6.12.4. Wave modified stress (`ln_tauwoc` & `ln_tauw`)

The surface stress felt by the ocean is the atmospheric stress minus the net stress going into the waves (Janssen et al., 2013). Therefore, when waves are growing, momentum and energy is spent and is not available for forcing the mean circulation, while in the opposite case of a decaying sea state, more momentum is available for forcing the ocean. Only when the sea state is in equilibrium, the ocean is forced by the atmospheric stress, but in practice, an equilibrium sea state is a fairly rare event. So the atmospheric stress felt by the ocean circulation $\tau_{oc,a}$ can be expressed as:

$$\tau_{oc,a} = \tau_a - \tau_w$$

where τ_a is the atmospheric surface stress; τ_w is the atmospheric stress going into the waves defined as:

$$\tau_w = \rho g \int \frac{dk}{c_p} (S_{in} + S_{nl} + S_{diss})$$

where: c_p is the phase speed of the gravity waves, S_{in} , S_{nl} and S_{diss} are three source terms that represent the physics of ocean waves. The first one, S_{in} , describes the generation of ocean waves by wind and therefore represents the momentum and energy transfer from air to ocean waves; the second term S_{nl} denotes the nonlinear transfer by resonant four-wave interactions; while the third term S_{diss} describes the dissipation of waves by processes such as white-capping, large scale breaking eddy-induced damping.

The wave stress derived from an external wave model can be provided either through the normalized wave stress into the ocean by setting `ln_tauwoc=.true.`, or through the zonal and meridional stress components by setting `ln_tauw=.true.`.

6.13. Miscellaneous options

6.13.1. Diurnal cycle (`sbcscy.F90`)

Bernie et al. (2005) 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. Nevertheless, it is possible to obtain a reasonable diurnal cycle of the SST knowing only short wave flux (SWF) at high frequency (Bernie et al., 2007). 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 Bernie et al. (2007) reconstruction algorithm is available in *NEMO* by setting `ln_dm2dc=.true.` (a `&namsbc` (namelist 6.1) namelist variable) when using a bulk formulation (`ln_blk=.true.`) or the flux formulation (`ln_flg=.true.`). The reconstruction is performed in the `sbcscy.F90` module. The detail of the algorithm used can be found in the appendix A of Bernie et al. (2007). The algorithm preserves the daily mean incoming SWF as the reconstructed SWF at a given time step is the mean value of the analytical cycle over this time step (figure 6.2). The use of diurnal cycle reconstruction requires the input SWF to be daily (*i.e.* a frequency of 24 hours and a time interpolation set to true in `sn_qsr` namelist parameter). Furthermore, it is recommended to have a least 8 surface module time steps per day, that is $\Delta t_{nn_fsc} < 10,800 s = 3 h$. An example of reconstructed SWF is given in figure 6.3 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.

6.13.2. Rotation of vector pairs onto the model grid directions

When using a flux (`ln_flg=.true.`) or bulk (`ln_blk=.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

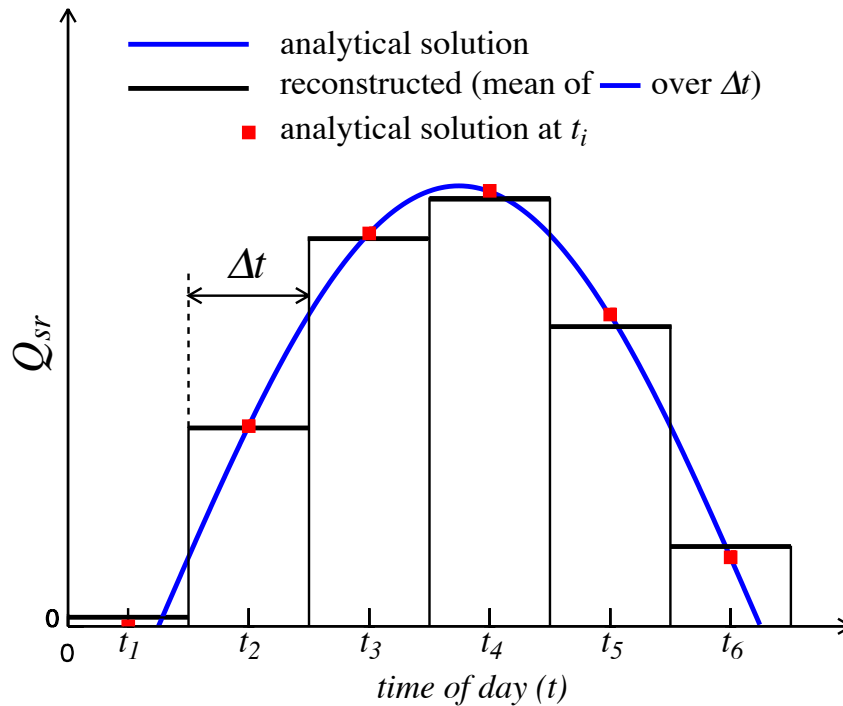


Figure 6.2.: Example of reconstruction of the diurnal cycle variation of short wave flux from daily mean values. The reconstructed diurnal cycle (black line) is chosen as the mean value of the analytical cycle (blue line) over a time step, not as the mid time step value of the analytically cycle (red square). From [Bernie et al. \(2007\)](#).

so on. The extra characters used in the strings are arbitrary. The `rot_rep` routine from the `geo2ocean.F90` module is used to perform the rotation.

6.13.3. Surface restoring to observed SST and/or SSS (`sbcssr.F90`)

Options are defined through the `&nam_sbc_ssr` (namelist 6.13) namelist variables. On forced mode using a flux formulation (`ln_flux=.true.`), a feedback term *must* be added to the surface heat flux Q_{ns}^o :

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

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 $-40 \text{ W/m}^2/\text{K}$. 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 :

$$emp = emp_o + \gamma_s^{-1} e_{3t} \frac{(S|_{k=1} - SSS_{Obs})}{S|_{k=1}} \quad (6.1)$$

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 ([Steele et al., 2001](#))), $S|_{k=1}$ 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 [equation 6.1](#) as the atmosphere does not care about ocean surface salinity ([Madec and Delécluse, 1997](#)). The SSS restoring term should be viewed as a flux correction on freshwater fluxes to reduce the uncertainties we have on the observed freshwater budget.

6.13.4. 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 found in `&nam_sbc` (namelist 6.1) namelist.

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.

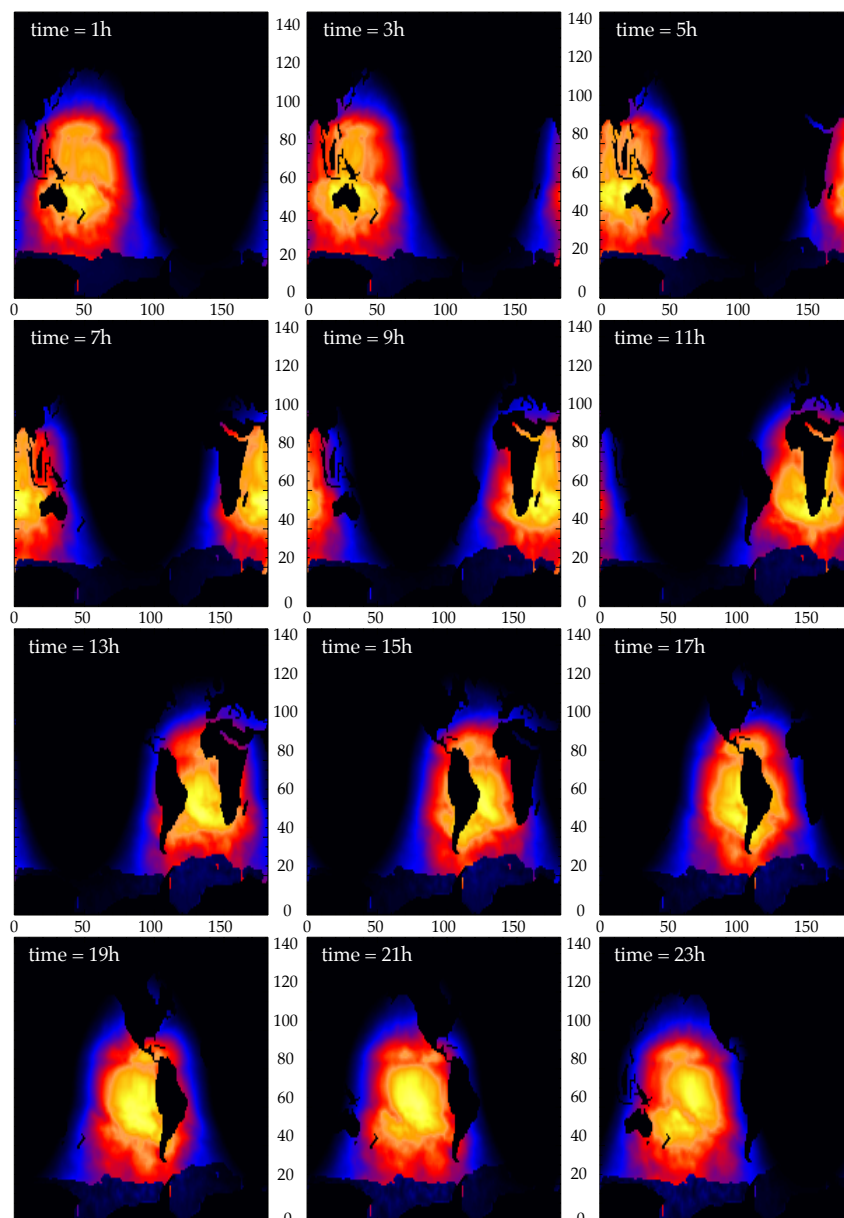


Figure 6.3.: Example of reconstruction of the diurnal cycle variation of short wave flux from daily mean values on an ORCA2 grid with a time sampling of 2 hours (from 1am to 11pm). The display is on (i,j) plane.

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 a 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 averaged ocean fluxes. Note that the activation of a sea-ice model is done by defining a CPP key (**key_si3** or **key_cice**). The activation automatically overwrites the read value of `nn_ice` to its appropriate value (*i.e.* 2 for SI3 or 3 for CICE).

6.13.5. Interface to CICE (*sbcice_cice.F90*)

It is possible to couple a regional or global *NEMO* configuration (without AGRIF) to the CICE sea-ice model by using **key_cice**. The CICE code can be obtained from LANL and the additional 'hadgem3' drivers will be required, even with the latest code release. Input grid files consistent with those used in *NEMO* will also be needed, and CICE CPP keys **ORCA_GRID**, **CICE_IN_NEMO** and **coupled** should be used (seek advice from UKMO if necessary). Currently,

```

!-----
&namsbc_ssr ! surface boundary condition : sea surface restoring (ln_ssr =T)
!-----
nn_sstr      = 0      ! add a retroaction term to the surface heat flux (=1) or not (=0)
rn_dqdt     = -40.   ! magnitude of the retroaction on temperature [W/m2/K]
nn_sssr     = 0      ! add a damping term to the surface freshwater flux (=2)
!           ! or to SSS only (=1) or no damping term (=0)
rn_deds     = -166.67 ! 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]

cn_dir      = './'   ! root directory for the SST/SSS data location

↵ !-----!-----!-----!-----!-----!-----!-----!-----!-----!
!           ! file name           ! frequency (hours) ! variable ! time interp.! clim ! 'yearly' / !
↵ weights filename ! rotation ! land/sea mask !
!           !           ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
↵ ! pairing ! filename !
sn_sst      = 'sst_data' , , 24. , 'sst' , .false. , .false. , 'yearly' ,
↵ '' , '' , '' ,
sn_sss      = 'sss_data' , , -1. , 'sss' , .true. , .true. , 'yearly' ,
↵ '' , '' , '' ,
/

```

namelist 6.13.: &namsbc_ssr

the code is only designed to work when using the NCAR forcing option for *NEMO* (with *calc_strair*=*.true.* and *calc_Tsfc*=*.true.* in the CICE name-list), or alternatively when *NEMO* is coupled to the HadGAM3 atmosphere model (with *calc_strair*=*.false.* and *calc_Tsfc*=*false*). The code is intended to be used with *nn_fsbc* set to 1 (although coupling ocean and ice less frequently should work, it is possible the calculation of some of the ocean-ice fluxes needs to be modified slightly - the user should check that results are not significantly different to the standard case).

There are two options for the technical coupling between *NEMO* and CICE. The standard version allows complete flexibility for the domain decompositions in the individual models, but this is at the expense of global gather and scatter operations in the coupling which become very expensive on larger numbers of processors. The alternative option (using **key_nemocice_decomp** for both *NEMO* and CICE) ensures that the domain decomposition is identical in both models (provided domain parameters are set appropriately, and *processor_shape* = *square-ice* and *distribution_wght* = *block* in the CICE name-list) and allows much more efficient direct coupling on individual processors. This solution scales much better although it is at the expense of having more idle CICE processors in areas where there is no sea ice.

6.13.6. 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 freshwater budget are proposed:

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 *EM-Pave_old.dat* file. As the model uses the Boussinesq approximation, the annual mean fresh water budget is simply evaluated from the change in the mean sea level at January the first and saved in the *EMPav.dat* file.

Lateral Boundary Condition (LBC)

Table of contents

7.1.	Boundary condition at the coast (<code>rn_shlat</code>)	92
7.2.	Model domain boundary condition (<code>jperio</code>)	93
7.2.1.	Closed, cyclic (<code>=0, 1, 2, 7</code>)	93
7.2.2.	North-fold (<code>=3, 6</code>)	93
7.3.	Exchange with neighbouring processors (<code>lbclnk.F90, lib_mpp.F90</code>)	95
7.4.	Unstructured open boundary conditions (BDY)	97
7.4.1.	Namelists	98
7.4.2.	Flow relaxation scheme	100
7.4.3.	Flather radiation scheme	101
7.4.4.	Orlanski radiation scheme	101
7.4.5.	Relaxation at the boundary	101
7.4.6.	Boundary geometry	102
7.4.7.	Input boundary data files	102
7.4.8.	Volume correction	102
7.4.9.	Tidal harmonic forcing	103

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

```

!-----
&namlbc      ! lateral momentum boundary condition          (default: NO selection)
!-----
!
! free slip ! partial slip ! no slip ! strong slip
rn_shlat    = -9999. ! shlat = 0 ! 0 < shlat < 2 ! shlat = 2 ! 2 < shlat
ln_vorlat   = .false. ! consistency of vorticity boundary condition with analytical Eqs.
/

```

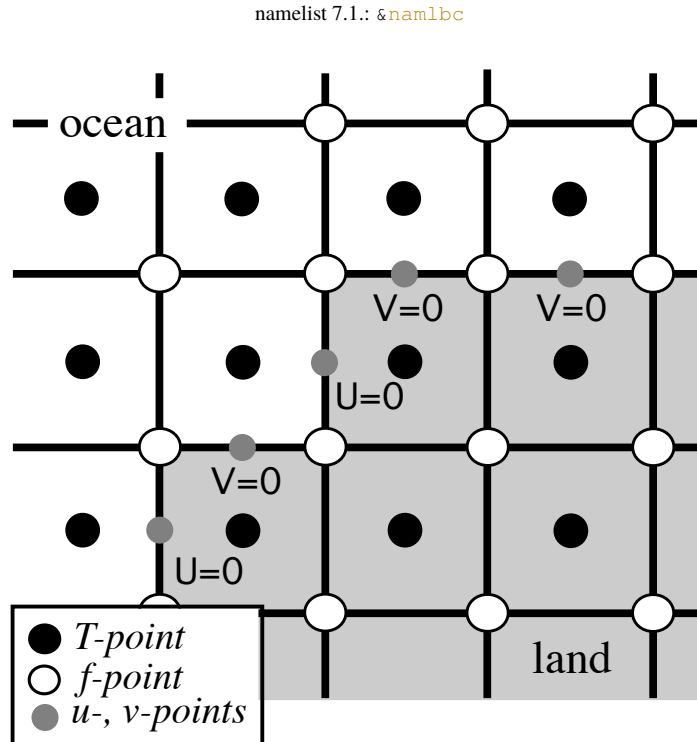


Figure 7.1.: Lateral boundary (thick line) at T-level. The velocity normal to the boundary is set to zero.

7.1. Boundary condition at the coast (`rn_shlat`)

Options are defined through the `&namlbc` (namelist 7.1) namelist variables. 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,

$$\frac{A^{iT}}{e_1} \frac{\partial T}{\partial i} \equiv \frac{A_u^{iT}}{e_{1u}} \delta_{i+1/2} [T] \text{ mask}_u$$

(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) (figure 7.1).

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

different types of lateral boundary condition are available, controlled by the value of the `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 is zero at the coast, so the vorticity: `mask_f` array is set to zero inside the land and just at the coast (figure 7.2-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 (figure 7.2-b). Therefore, the vorticity along the coastlines is given by:

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

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` :

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

"partial" free-slip boundary condition ($0 < \text{rn_shlat} < 2$) 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 (figure 7.2-c). This can be selected by providing a value of `mask_f` strictly inbetween 0 and 2.

"strong" no-slip boundary condition ($2 < \text{rn_shlat}$) the viscous boundary layer is assumed to be smaller than half the grid size (figure 7.2-d). The friction is thus larger than in the no-slip case.

Note that when the bottom topography is entirely represented by the s -coordinates (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.

7.2. Model domain boundary condition (`jperio`)

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

7.2.1. Closed, cyclic (`jperio = 0, 1, 2, 7`)

The choice of closed or cyclic model domain boundary condition is made by setting `jperio` to 0, 1, 2 or 7 in namelist `&namcfg` (namelist 15.1). 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.

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 (figure 7.3-a). Whatever flows out of the eastern (western) end of the basin enters the western (eastern) end.

For cyclic north-south boundary (`jperio = 2`), first and last columns are set to zero (closed) whilst the first row is set to the value of the last-but-one row and the last row to the value of the second one (figure 7.3-a). Whatever flows out of the northern (southern) end of the basin enters the southern (northern) end.

Bi-cyclic east-west and north-south boundary (`jperio = 7`) combines cases 1 and 2.

7.2.2. North-fold (`jperio = 3, 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 (figure 15.1, and thus requires a specific treatment illustrated in figure 7.4. Further information can be found in `lbcnfd.F90` module which applies the north fold boundary condition.

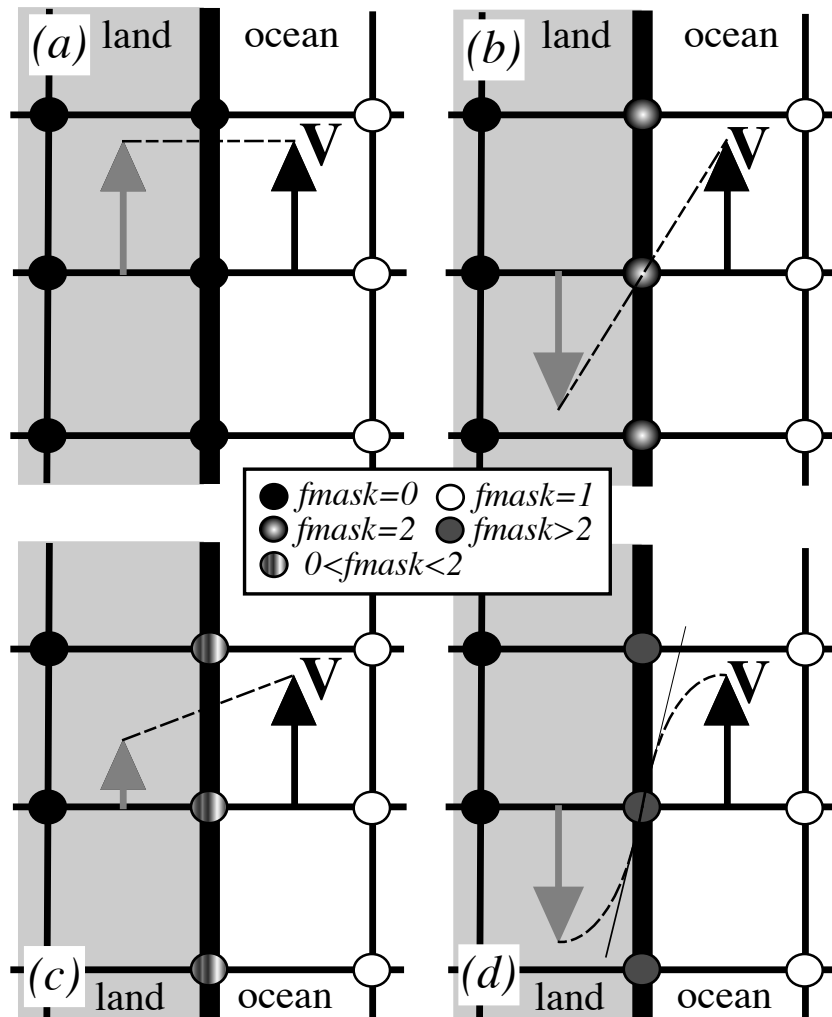


Figure 7.2.: Lateral boundary conditions (a) free-slip ($rn_shlat=0$); (b) no-slip ($rn_shlat=2$); (c) "partial" free-slip ($0 < rn_shlat < 2$) and (d) "strong" no-slip ($2 < rn_shlat$). Implied "ghost" velocity inside land area is display in grey.

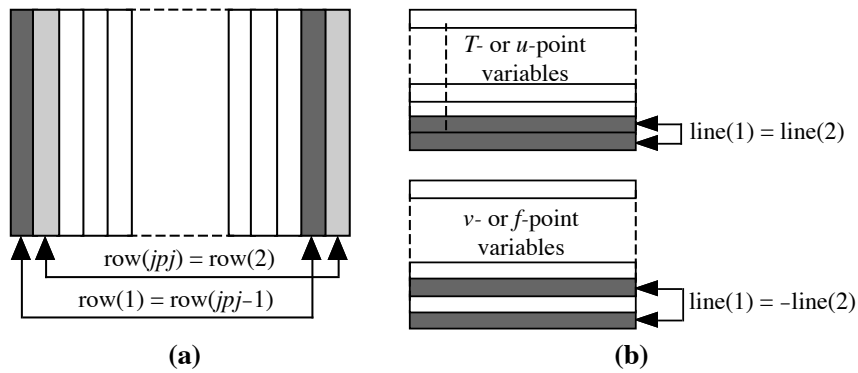


Figure 7.3.: Setting of (a) east-west cyclic (b) symmetric across the Equator boundary conditions

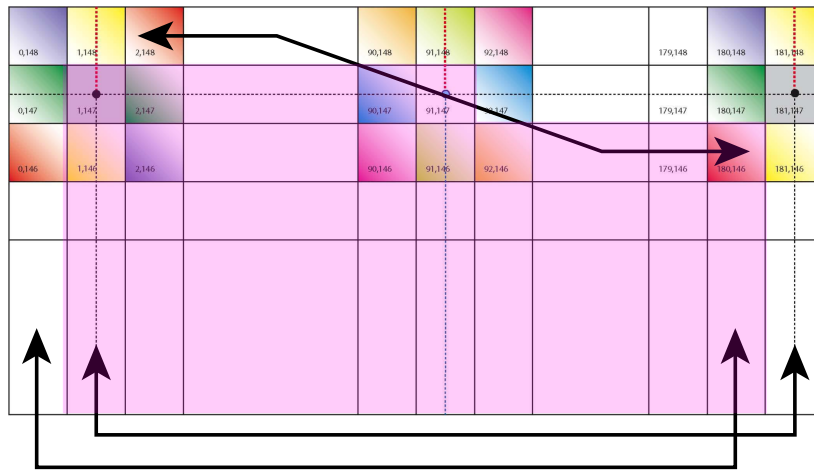


Figure 7.4.: North fold boundary with a T -point pivot and cyclic east-west boundary condition ($jperio = 4$), as used in ORCA 2°, 1/4° and 1/12°. Pink shaded area corresponds to the inner domain mask (see text).

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

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

```

!-----
&nammpp      !   Massively Parallel Processing                               ("key_mpp_mpi")
!-----
ln_listonly = .false. ! do nothing else than listing the best domain decompositions (with land domains
↳ suppression)
!
ln_nnogather = .true. ! if T: the largest number of cores tested is defined by max(mppsize, jpni*jpnj)
! activate code to avoid mpi_allgather use at the northfold
jpni        = 0       ! number of processors following i (set automatically if < 1), see also ln_listonly =
↳ T
jpnj        = 0       ! number of processors following j (set automatically if < 1), see also ln_listonly =
↳ T
/

```

namelist 7.2.: &nammpp

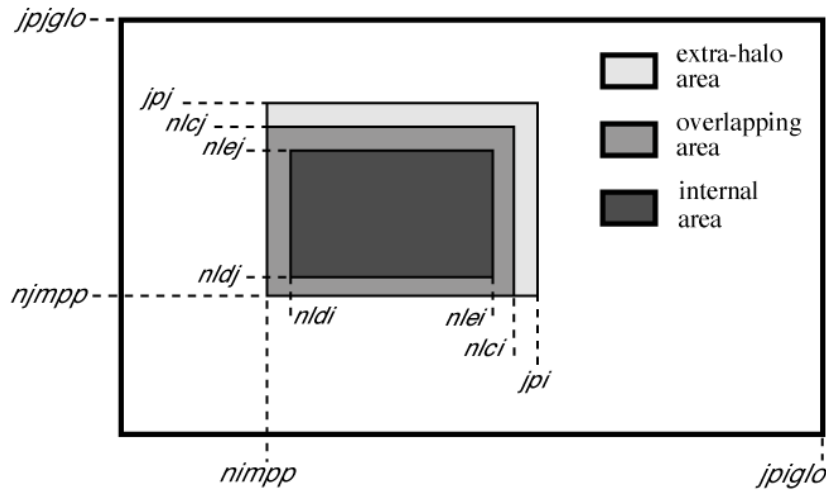


Figure 7.5.: Positioning of a sub-domain when massively parallel processing is used

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). Communications are first done according to the east-west direction and next according to the north-south direction. There is no specific communications for the corners. The communication is done through the Message Passing Interface (MPI) and requires `key_mpp_mpi`. Use also `key_mpi2` if MPI3 is not available on your computer. The data exchanges between processors are required at the very place where lateral domain boundary conditions are set in the mono-domain computation: the `lbc_lnk` routine (found in `lbclnk.F90` module) which manages such conditions is interfaced with routines found in `lib_mpp.F90` module. The output file `communication_report.txt` provides the list of which routines do how many communications during 1 time step of the model.

In *NEMO*, the splitting is regular and arithmetic. The total number of subdomains corresponds to the number of MPI processes allocated to *NEMO* when the model is launched (*i.e.* `mpirun -np x ./nemo` will automatically give `x` subdomains). The *i*-axis is divided by `jpni` and the *j*-axis by `jpnj`. These parameters are defined in `&nammpp` (`namelist 7.2`) namelist. If `jpni` and `jpnj` are `< 1`, they will be automatically redefined in the code to give the best domain decomposition (see below).

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 `jpj`, `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 (`nn_hls=1`, in `par_oce.F90`, and must be kept to one until further notice). The whole domain dimensions are named `jpiglo`, `jjglo` and `jpk`. The relationship between the whole domain and a sub-domain is:

$$jpi = (jpiglo - 2 \times nn_hls + (jpni - 1)) / jpni + 2 \times nn_hls$$

$$jpj = (jjglo - 2 \times nn_hls + (jpnj - 1)) / jpnj + 2 \times nn_hls$$

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 (figure 7.5). Note that since the version 4, there is no more extra-halo area as defined in figure 7.5 so `jpj` is now always equal to `nldi` and `jpj` equal to `nlej`.

An element of T_l , a local array (subdomain) corresponds to an element of T_g , a global array (whole domain) by the relationship:

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

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

The 1-d arrays `mig(1 : jpi)` and `mjj(1 : jpj)`, defined in `dom_glo` routine (`domain.F90` module), should be used to get global domain indices from local domain indices. The 1-d arrays, `mi0(1 : jpiglo)`, `mi1(1 : jpiglo)` and `mj0(1 : jjglo)`, `mj1(1 : jjglo)` have the reverse purpose and should be used to define loop indices expressed in

global domain indices (see examples in `dtastd.F90` module).

The *NEMO* model computes equation terms with the help of mask arrays (0 on land points and 1 on sea points). It is therefore possible that an MPI subdomain contains only land points. To save resources, we try to suppress from the computational domain as much land subdomains as possible. For example if N_{mpi} processes are allocated to NEMO, the domain decomposition will be given by the following equation:

$$N_{mpi} = jpn_i \times jpn_j - N_{land} + N_{useless}$$

N_{land} is the total number of land subdomains in the domain decomposition defined by `jpn_i` and `jpn_j`. $N_{useless}$ is the number of land subdomains that are kept in the computational domain in order to make sure that N_{mpi} MPI processes are indeed allocated to a given subdomain. The values of N_{mpi} , `jpn_i`, `jpn_j`, N_{land} and $N_{useless}$ are printed in the output file `ocean.output`. $N_{useless}$ must, of course, be as small as possible to limit the waste of resources. A warning is issued in `ocean.output` if $N_{useless}$ is not zero. Note that non-zero value of $N_{useless}$ is usually required when using AGRIF as, up to now, the parent grid and each of the child grids must use all the N_{mpi} processes.

If the domain decomposition is automatically defined (when `jpn_i` and `jpn_j` are < 1), the decomposition chosen by the model will minimise the sub-domain size (defined as $max_{all\ domains}(jpi \times jpj)$) and maximize the number of eliminated land subdomains. This means that no other domain decomposition (a set of `jpn_i` and `jpn_j` values) will use less processes than $(jpn_i \times jpn_j - N_{land})$ and get a smaller subdomain size. In order to specify N_{mpi} properly (minimize $N_{useless}$), you must run the model once with `ln_list` activated. In this case, the model will start the initialisation phase, print the list of optimum decompositions (N_{mpi} , `jpn_i` and `jpn_j`) in `ocean.output` and directly abort. The maximum value of N_{mpi} tested in this list is given by $max(N_{MPI_tasks}, jpn_i \times jpn_j)$. For example, run the model on 40 nodes with `ln_list` activated and `jpn_i = 10000` and `jpn_j = 1`, will print the list of optimum domains decomposition from 1 to about 10000.

Processors are numbered from 0 to $N_{mpi} - 1$. Subdomains containing some ocean points are numbered first from 0 to $jpn_i * jpn_j - N_{land} - 1$. The remaining $N_{useless}$ land subdomains are numbered next, which means that, for a given (`jpn_i`, `jpn_j`), the numbers attributed to the ocean subdomains do not vary with $N_{useless}$.

When land processors are eliminated, the value corresponding to these locations in the model output files is undefined. `ln_mskland` must be activated in order to avoid Not a Number values in output files. Note that it is better to not eliminate land processors when creating a meshmask file (*i.e.* when setting a non-zero value to `nn_msh`).

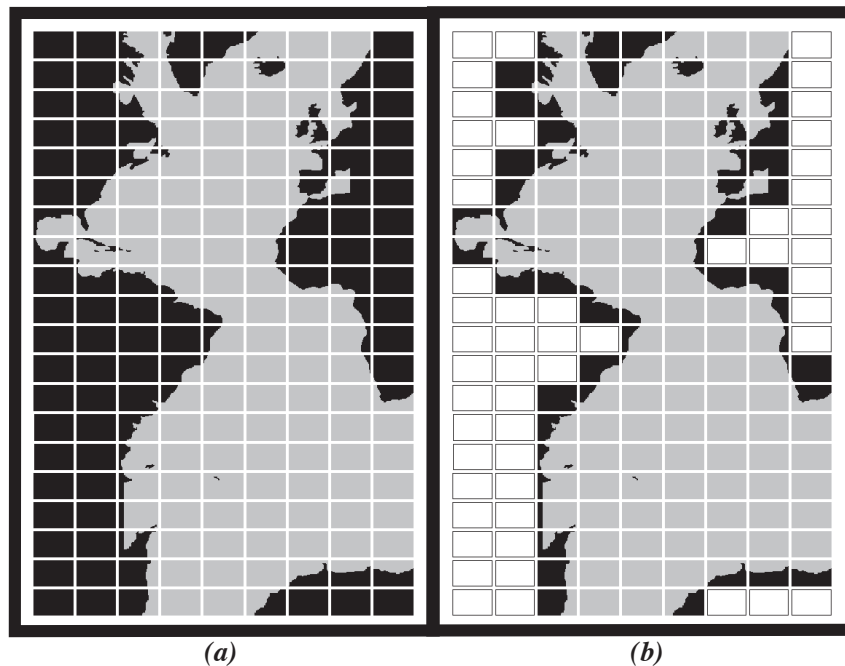


Figure 7.6.: Example of Atlantic domain defined for the CLIPPER projet. Initial grid is composed of 773 x 1236 horizontal points. (a) the domain is split onto 9 *times* 20 subdomains (`jpn_i=9`, `jpn_j=20`). 52 subdomains are land areas. (b) 52 subdomains are eliminated (white rectangles) and the resulting number of processors really used during the computation is `jpn_ij=128`.

7.4. Unstructured open boundary conditions (BDY)

Options are defined through the `&nambdy` (namelist 7.3) and `&nambdy_dta` (namelist 7.4) namelist variables. The BDY module is the core implementation of open boundary conditions for regional configurations on ocean temperature,

```

!-----
&nambdy      ! unstructured open boundaries                (default: OFF)
!-----
ln_bdy       = .false.  ! Use unstructured open boundaries
nb_bdy       = 0        ! number of open boundary sets
ln_coords_file = .true.  ! =T : read bdy coordinates from file
cn_coords_file = 'coordinates.bdy.nc' ! bdy coordinates files
ln_mask_file  = .false.  ! =T : read mask from file
cn_mask_file  = ''      ! name of mask file (if ln_mask_file=.TRUE.)
cn_dyn2d     = 'none'   !
nn_dyn2d_dta = 0       ! = 0, bdy data are equal to the initial state
!               ! = 1, bdy data are read in 'bdydata .nc' files
!               ! = 2, use tidal harmonic forcing data from files
!               ! = 3, use external data AND tidal harmonic forcing
cn_dyn3d     = 'none'   !
nn_dyn3d_dta = 0       ! = 0, bdy data are equal to the initial state
!               ! = 1, bdy data are read in 'bdydata .nc' files
cn_tra       = 'none'   !
nn_tra_dta   = 0       ! = 0, bdy data are equal to the initial state
!               ! = 1, bdy data are read in 'bdydata .nc' files
cn_ice       = 'none'   !
nn_ice_dta   = 0       ! = 0, bdy data are equal to the initial state
!               ! = 1, bdy data are read in 'bdydata .nc' files
!
ln_tra_dmp   = .false.  ! open boudaries conditions for tracers
ln_dyn3d_dmp = .false.  ! open boundary condition for baroclinic velocities
rn_time_dmp  = 1.       ! Damping time scale in days
rn_time_dmp_out = 1.    ! Outflow damping time scale
nn_rimwidth  = 10       ! width of the relaxation zone
ln_vol       = .false.  ! total volume correction (see nn_volctl parameter)
nn_volctl1  = 1        ! = 0, the total water flux across open boundaries is zero
/

```

namelist 7.3.: &nambdy

salinity, barotropic-baroclinic velocities, ice-snow concentration, thicknesses, temperatures, salinity and melt ponds concentration and thickness.

The BDY module was modelled on the OBC module (see *NEMO* 3.4) and shares many features and a similar coding structure (Chanut, 2005). The specification of the location of the open boundary is completely flexible and allows any type of setup, from regular boundaries to irregular contour (it includes the possibility to set an open boundary able to follow an isobath). Boundary data files used with versions of *NEMO* prior to Version 3.4 may need to be re-ordered to work with this version. See the section on the Input Boundary Data Files for details.

7.4.1. Namelists

The BDY module is activated by setting `ln_bdy=.true.`. It is possible to define more than one boundary “set” and apply different boundary conditions to each set. The number of boundary sets is defined by `nb_bdy`. Each boundary set can be either defined as a series of straight line segments directly in the namelist (`ln_coords_file=.false.`, and a namelist block `&nambdy_index(??)` must be included for each set) or read in from a file (`ln_coords_file=.true.`, and a “*coordinates.bdy.nc*” file must be provided). The *coordinates.bdy* file is analogous to the usual *NEMO* “*coordinates.nc*” file. In the example above, there are two boundary sets, the first of which is defined via a file and the second is defined in the namelist. For more details of the definition of the boundary geometry see section subsection 7.4.6.

For each boundary set a boundary condition has to be chosen for the barotropic solution (“u2d”: sea-surface height and barotropic velocities), for the baroclinic velocities (“u3d”), for the active tracers* (“tra”), and for sea-ice (“ice”). For each set of variables one has to choose an algorithm and the boundary data (set resp. by `cn_tra` and `nn_tra_dta` for tracers).

The choice of algorithm is currently as follows:

- 'none': No boundary condition applied. So the solution will “see” the land points around the edge of the edge of the domain.
- 'specified': Specified boundary condition applied (only available for baroclinic velocity and tracer variables).
- 'neumann': Value at the boundary are duplicated (No gradient). Only available for baroclinic velocity and tracer variables.
- 'frs': Flow Relaxation Scheme (FRS) available for all variables.

*The BDY module does not deal with passive tracers at this version

```

!-----
&nambdy_dta      ! open boundaries - external data                (see nam_bdy)
!-----
ln_zinterp      = .false.      ! T if a vertical interpolation is required. Variables gdep[tuv] and e3[tuv] must
!                               ! exist in the file
ln_full_vel     = .false.      ! automatically defined to T if the number of vertical levels in bdy dta /= jpk
!                               ! T if [uv]3d are "full" velocities and not only its baroclinic components
!                               ! in this case, baroclinic and barotropic velocities will be recomputed -> [uv]2d
!                               ! not needed
cn_dir          = 'bdydt/'     ! root directory for the BDY data location

!-----
!                               ! file name      ! frequency (hours) ! variable ! time interp.! clim ! 'yearly'/ !
! weights filename ! rotation ! land/sea mask !
!                               ! (if <0 months) ! name      ! (logical) ! (T/F) ! 'monthly' !
!-----
! pairing ! filename !
bn_ssh     = 'amml2_bdyT_u2d' ,      24.      , 'sossh eig' , .true. , .false. , 'daily' ,
!-----
bn_u2d     = 'amml2_bdyU_u2d' ,      24.      , 'vobtcrtx' , .true. , .false. , 'daily' ,
!-----
bn_v2d     = 'amml2_bdyV_u2d' ,      24.      , 'vobtcrtx' , .true. , .false. , 'daily' ,
!-----
bn_u3d     = 'amml2_bdyU_u3d' ,      24.      , 'vozocrtx' , .true. , .false. , 'daily' ,
!-----
bn_v3d     = 'amml2_bdyV_u3d' ,      24.      , 'vomecrtx' , .true. , .false. , 'daily' ,
!-----
bn_tem     = 'amml2_bdyT_tra' ,      24.      , 'votemper' , .true. , .false. , 'daily' ,
!-----
bn_sal     = 'amml2_bdyT_tra' ,      24.      , 'vosaline' , .true. , .false. , 'daily' ,
!-----
!* for si3
bn_a_i     = 'amml2_bdyT_ice' ,      24.      , 'siconc' , .true. , .false. , 'daily' ,
!-----
bn_h_i     = 'amml2_bdyT_ice' ,      24.      , 'sithic' , .true. , .false. , 'daily' ,
!-----
bn_h_s     = 'amml2_bdyT_ice' ,      24.      , 'snthic' , .true. , .false. , 'daily' ,
!-----
bn_t_i     = 'NOT USED' ,      24.      , 'sitemp' , .true. , .false. , 'daily' ,
!-----
bn_t_s     = 'NOT USED' ,      24.      , 'sntemp' , .true. , .false. , 'daily' ,
!-----
bn_tsu     = 'NOT USED' ,      24.      , 'sittop' , .true. , .false. , 'daily' ,
!-----
bn_s_i     = 'NOT USED' ,      24.      , 'sisalt' , .true. , .false. , 'daily' ,
!-----
! melt ponds (be careful, bn_aip is the pond concentration (not fraction), so it differs from rn_iceapnd)
bn_aip     = 'NOT USED' ,      24.      , 'siapnd' , .true. , .false. , 'daily' ,
!-----
bn_hip     = 'NOT USED' ,      24.      , 'sihpnd' , .true. , .false. , 'daily' ,
!-----
! if bn_t_i etc are "not used", then define arbitrary temperatures and salinity and ponds
rn_ice_tem = 270.      ! arbitrary temperature of incoming sea ice
rn_ice_sal = 10.      ! -- salinity --
rn_ice_age  = 30.      ! -- age --
rn_ice_apnd = 0.2     ! -- pond fraction = a_ip/a_i --
rn_ice_hpnd = 0.05    ! -- pond depth --
/

```

namelist 7.4.: &nambdy_dta

'**Orlanski**': Orlanski radiation scheme (fully oblique) for barotropic, baroclinic and tracer variables.

'**Orlanski_npo**': Orlanski radiation scheme for barotropic, baroclinic and tracer variables.

'**flather**': Flather radiation scheme for the barotropic variables only.

The boundary data is either set to initial conditions (`nn_tra_dta=0`) or forced with external data from a file (`nn_tra_dta=1`). In case the 3d velocity data contain the total velocity (ie, baroclinic and barotropic velocity), the bdy code can derived baroclinic and barotropic velocities by setting `ln_full_vel=.true.` For the barotropic solution there is also the option to use tidal harmonic forcing either by itself (`nn_dyn2d_dta=2`) or in addition to other external data (`nn_dyn2d_dta=3`).

If not set to initial conditions, sea-ice salinity, temperatures and melt ponds data at the boundary can either be read in a file or defined as constant (by `rn_ice_sal` , `rn_ice_tem` , `rn_ice_apnd` , `rn_ice_hpnd`). Ice age is constant and defined by `rn_ice_age` .

If external boundary data is required then the `&nambdy_dta` (namelist 7.4) namelist must be defined. One `&nambdy_dta` (namelist 7.4) namelist is required for each boundary set, adopting the same order of indexes in which the boundary sets are defined in `nambdy`. In the example given, two boundary sets have been defined. The first one is reading data file in the `&nambdy_dta` (namelist 7.4) namelist shown above and the second one is using data from initial condition (no namelist block needed). The boundary data is read in using the `fldread` module, so the `&nambdy_dta` (namelist 7.4) namelist is in the format required for `fldread`. For each required variable, the filename, the frequency of the files and the frequency of the data in the files are given. Also whether or not time-interpolation is required and whether the data is climatological (time-cyclic) data. For sea-ice salinity, temperatures and melt ponds, reading the files are skipped and constant values are used if filenames are defined as 'NOT USED'.

There is currently an option to vertically interpolate the open boundary data onto the native grid at run-time. If `nn_bdy_jpk < -1`, it is assumed that the lateral boundary data are already on the native grid. However, if `nn_bdy_jpk` is set to the number of vertical levels present in the boundary data, a bilinear interpolation onto the native grid will be triggered at runtime. For this to be successful the additional variables: `gdept`, `gdepu`, `gdepv`, `e3t`, `e3u` and `e3v`, are required to be present in the lateral boundary files. These correspond to the depths and scale factors of the input data, the latter used to make any adjustment to the velocity fields due to differences in the total water depths between the two vertical grids.

In the example of given namelists, two boundary sets are defined. The first set is defined via a file and applies FRS conditions to temperature and salinity and Flather conditions to the barotropic variables. No condition specified for the baroclinic velocity and sea-ice. External data is provided in daily files (from a large-scale model). Tidal harmonic forcing is also used. The second set is defined in a namelist. FRS conditions are applied on temperature and salinity and climatological data is read from initial condition files.

7.4.2. Flow relaxation scheme

The Flow Relaxation Scheme (FRS) (Davies, 1976; Engedahl, 1995), 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

$$\Phi(d) = \alpha(d)\Phi_e(d) + (1 - \alpha(d))\Phi_m(d) \quad d = 1, N$$

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:

$$-\frac{1}{\tau}(\Phi - \Phi_e)$$

where the relaxation time scale τ is given by a function of α and the model time step Δt :

$$\tau = \frac{1 - \alpha}{\alpha} \Delta t$$

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:

$$\alpha(d) = 1 - \tanh\left(\frac{d-1}{2}\right), \quad d = 1, N$$

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.

7.4.3. Flather radiation scheme

The [Flather \(1994\)](#) scheme is a radiation condition on the normal, depth-mean transport across the open boundary. It takes the form

$$U = U_e + \frac{c}{h} (\eta - \eta_e), \quad (7.1)$$

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 [equation 7.1](#) is a spatial gradient across the model boundary, so that η_e is defined on the T points with $nbr = 1$ and η is defined on the T points with $nbr = 2$. U and U_e are defined on the U or V points with $nbr = 1$, *i.e.* between the two T grid points.

7.4.4. Orlanski radiation scheme

The Orlanski scheme is based on the algorithm described by ([Marchesiello et al., 2001](#)), hereafter MMS.

The adaptive Orlanski condition solves a wave plus relaxation equation at the boundary:

$$\frac{\partial \phi}{\partial t} + c_x \frac{\partial \phi}{\partial x} + c_y \frac{\partial \phi}{\partial y} = -\frac{1}{\tau} (\phi - \phi^{ext}) \quad (7.2)$$

where ϕ is the model field, x and y refer to the normal and tangential directions to the boundary respectively, and the phase velocities are diagnosed from the model fields as:

$$c_x = -\frac{\partial \phi}{\partial t} \frac{\partial \phi / \partial x}{(\partial \phi / \partial x)^2 + (\partial \phi / \partial y)^2} \quad (7.3)$$

$$c_y = -\frac{\partial \phi}{\partial t} \frac{\partial \phi / \partial y}{(\partial \phi / \partial x)^2 + (\partial \phi / \partial y)^2} \quad (7.4)$$

(As noted by MMS, this is a circular diagnosis of the phase speeds which only makes sense on a discrete grid). Equation ([equation 7.2](#)) is defined adaptively depending on the sign of the phase velocity normal to the boundary c_x . For c_x outward, we have

$$\tau = \tau_{out} \quad (7.5)$$

For c_x inward, the radiation equation is not applied:

$$\tau = \tau_{in} ; c_x = c_y = 0 \quad (7.6)$$

Generally the relaxation time scale at inward propagation points (`rn_time_dmp`) is set much shorter than the time scale at outward propagation points (`rn_time_dmp_out`) so that the solution is constrained more strongly by the external data at inward propagation points. See [subsection 7.4.5](#) for detailed on the spatial shape of the scaling.

The ‘‘normal propagation of oblique radiation’’ or NPO approximation (called '`orlanski_npo`') involves assuming that c_y is zero in equation ([equation 7.2](#)), but including this term in the denominator of equation ([equation 7.3](#)). Both versions of the scheme are options in BDY. Equations ([equation 7.2](#)) - ([equation 7.6](#)) correspond to equations (13) - (15) and (2) - (3) in MMS.

7.4.5. Relaxation at the boundary

In addition to a specific boundary condition specified as `cn_tra` and `cn_dyn3d`, relaxation on baroclinic velocities and tracers variables are available. It is control by the namelist parameter `ln_tra_dmp` and `ln_dyn3d_dmp` for each boundary set.

The relaxation time scale value (`rn_time_dmp` and `rn_time_dmp_out`, τ) are defined at the boundaries itself. This time scale (α) is weighted by the distance (d) from the boundary over `nn_rimwidth` cells (N):

$$\alpha = \frac{1}{\tau} \left(\frac{N+1-d}{N} \right)^2, \quad d = 1, N$$

The same scaling is applied in the Orlanski damping.

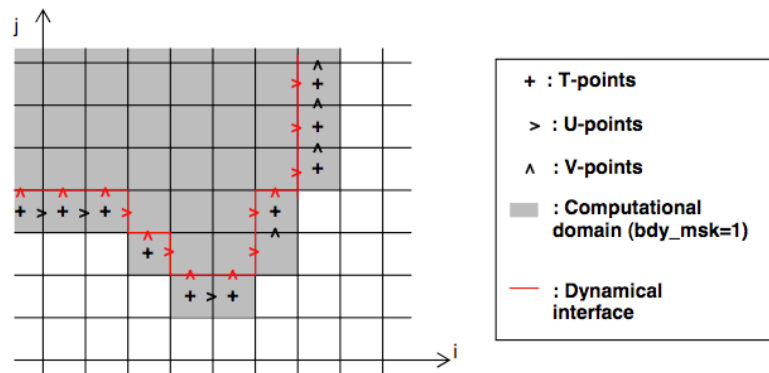


Figure 7.7.: Example of geometry of unstructured open boundary

7.4.6. Boundary geometry

Each open boundary set is defined as a list of points. The information is stored in the arrays *nbi*, *nbj*, and *nbr* in the *idx_bdy* structure. The *nbi* and *nbj* arrays define the local (*i*, *j*) indexes of each point in the boundary zone and the *nbr* array defines the discrete distance from the boundary: *nbr* = 1 means that the boundary point is next to the edge of the model domain, while *nbr* > 1 means that the boundary point is increasingly further away from the edge of the model domain. A set of *nbi*, *nbj*, and *nbr* arrays is defined for each of the *T*, *U* and *V* grids. figure 7.7 shows an example of an irregular boundary.

The boundary geometry for each set may be defined in a namelist *nambdy_index* or by reading in a “*coordinates.bdy.nc*” file. The *nambdy_index* namelist defines a series of straight-line segments for north, east, south and west boundaries. One *nambdy_index* namelist block is needed for each boundary condition defined by indexes. For the northern boundary, *nbdysegn* gives the number of segments, *jpjnob* gives the *j* index for each segment and *jpindt* and *jpinf* give the start and end *i* indices for each segment with similar for the other boundaries. These segments define a list of *T* grid points along the outermost row of the boundary (*nbr* = 1). The code deduces the *U* and *V* points and also the points for *nbr* > 1 if *nn_rimwidth* > 1.

The boundary geometry may also be defined from a “*coordinates.bdy.nc*” file. figure 7.8 gives an example of the header information from such a file, based on the description of geometrical setup given above. The file should contain the index arrays for each of the *T*, *U* and *V* grids. The arrays must be in order of increasing *nbr*. Note that the *nbi*, *nbj* values in the file are global values and are converted to local values in the code. Typically this file will be used to generate external boundary data via interpolation and so will also contain the latitudes and longitudes of each point as shown. However, this is not necessary to run the model.

For some choices of irregular boundary the model domain may contain areas of ocean which are not part of the computational domain. For example, if an open boundary is defined along an isobath, say at the shelf break, then the areas of ocean outside of this boundary will need to be masked out. This can be done by reading a mask file defined as *cn_mask_file* in the *nam_bdy* namelist. Only one mask file is used even if multiple boundary sets are defined.

7.4.7. Input boundary data files

The data files contain the data arrays in the order in which the points are defined in the *nbi* and *nbj* arrays. 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.

From Version 3.4 there are new restrictions on the order in which the boundary points are defined (and therefore restrictions on the order of the data in the file). In particular:

1. The data points must be in order of increasing *nbr*, ie. all the *nbr* = 1 points, then all the *nbr* = 2 points etc.
2. All the data for a particular boundary set must be in the same order. (Prior to 3.4 it was possible to define barotropic data in a different order to the data for tracers and baroclinic velocities).

These restrictions mean that data files used with versions of the model prior to Version 3.4 may not work with Version 3.4 onwards. A FORTRAN utility *bdy_reorder* exists in the TOOLS directory which will re-order the data in old BDY data files.

7.4.8. Volume correction

There is an option to force the total volume in the regional model to be constant. This is controlled by the *ln_vol* parameter in the namelist. A value of *ln_vol* = *.false.* indicates that this option is not used. Two options to control

```

netcdf med12.obc.coordinates {
dimensions:
    yb = 1 ;
    xbt = 3218 ;
    xbu = 3200 ;
    xbv = 3201 ;
variables:
    int nbit(yb, xbt) ;
    int nbiu(yb, xbu) ;
    int nbiv(yb, xbv) ;
    int nbjt(yb, xbt) ;
    int nbju(yb, xbu) ;
    int nbjv(yb, xbv) ;
    int nbrt(yb, xbt) ;
    int nbru(yb, xbu) ;
    int nbrv(yb, xbv) ;
    float elt(yb, xbt) ;
        elt:units = "metres" ;
    float elu(yb, xbu) ;
        elu:units = "metres" ;
    float elv(yb, xbv) ;
        elv:units = "metres" ;
    float e2t(yb, xbt) ;
        e2t:units = "metres" ;
    float e2u(yb, xbu) ;
        e2u:units = "metres" ;
    float e2v(yb, xbv) ;
        e2v:units = "metres" ;
    float glamt(yb, xbt) ;
        glamt:units = "degrees_east" ;
    float glamu(yb, xbu) ;
        glamu:units = "degrees_east" ;
    float glamv(yb, xbv) ;
        glamv:units = "degrees_east" ;
    float gphit(yb, xbt) ;
        gphit:units = "degrees_north" ;
    float gphiu(yb, xbu) ;
        gphiu:units = "degrees_north" ;
    float gp hiv(yb, xbv) ;
        gp hiv:units = "degrees_north" ;

// global attributes:
    :file_name = "med12.obc.coordinates.reorder.nc" ;
    :rimwidth = 9 ;
    :NCO = "3.9.9" ;
}

```

Figure 7.8.: Example of the header for a *coordinates.bdy.nc* file

the volume are available (`nn_volctl`). If `nn_volctl=0` then a correction is applied to the normal barotropic velocities around the boundary at each timestep to ensure that the integrated volume flow through the boundary is zero. If `nn_volctl=1` 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.

If more than one boundary set is used then volume correction is applied to all boundaries at once.

7.4.9. Tidal harmonic forcing

Tidal forcing at open boundaries requires the activation of surface tides (i.e., in `&nam_tide` (namelist 6.7), `ln_tide` needs to be set to `.true.` and the required constituents need to be activated by including their names in the `clname` array; see section 6.7). Specific options related to the reading in of the complex harmonic amplitudes of elevation (SSH) and barotropic velocity (u,v) at open boundaries are defined through the `&nambdy_tide` (namelist 7.5) namelist parameters.

The tidal harmonic data at open boundaries can be specified in two different ways, either on a two-dimensional grid covering the entire model domain or along open boundary segments; these two variants can be selected by setting `ln_bdytide_2ddta` to `.true.` or `.false.`, respectively. In either case, the real and imaginary parts of SSH and the two barotropic velocity components for each activated tidal constituent `tname` have to be provided separately: when two-dimensional data is used, variables `tname_z1` and `tname_z2` for real and imaginary SSH, respectively, are expected in input file `filtide` with suffix `_grid_T.nc`, variables `tname_u1` and `tname_u2` for real and imaginary u, respectively,

```

!-----
&nambdy_tide ! tidal forcing at open boundaries (default: OFF)
!-----
fildtide      = 'bdydt/amm12_bdytide_' ! file name root of tidal forcing files
ln_bdytide_2ddta = .false.           !
ln_bdytide_conj = .false.           !
/

```

namelist 7.5.: &nambdy_tide

are expected in input file `fildtide` with suffix `_grid_U.nc`, and `tname_v1` and `tname_v2` for real and imaginary v , respectively, are expected in input file `fildtide` with suffix `_grid_V.nc`; when data along open boundary segments is used, variables $z1$ and $z2$ (real and imaginary part of SSH) are expected to be available from file `fildtide` with suffix `tname_grid_T.nc`, variables $u1$ and $u2$ (real and imaginary part of u) are expected to be available from file `fildtide` with suffix `tname_grid_U.nc`, and variables $v1$ and $v2$ (real and imaginary part of v) are expected to be available from file `fildtide` with suffix `tname_grid_V.nc`. If `ln_bdytide_conj` is set to `.true.`, the data is expected to be in complex conjugate form.

Note that the barotropic velocity components are assumed to be defined on the native model grid and should be rotated accordingly when they are converted from their definition on a different source grid. To do so, the u , v amplitudes and phases can be converted into tidal ellipses, the grid rotation added to the ellipse inclination, and then converted back (care should be taken regarding conventions of the direction of rotation).

Table of contents

8.1. Lateral mixing operators	106
8.1.1. No lateral mixing (<code>ln_traldf_OFF</code> & <code>ln_dynldf_OFF</code>)	106
8.1.2. Laplacian mixing (<code>ln_traldf_lap</code> & <code>ln_dynldf_lap</code>)	106
8.1.3. Bilaplacian mixing (<code>ln_traldf_blp</code> & <code>ln_dynldf_blp</code>)	106
8.2. Direction of lateral mixing (<code>ldfslp.F90</code>)	106
8.2.1. Slopes for tracer geopotential mixing in the <i>s</i> -coordinate	106
8.2.2. Slopes for tracer iso-neutral mixing	107
8.2.3. Slopes for momentum iso-neutral mixing	108
8.3. Lateral mixing coefficient (<code>nn_aht_ijk_t</code> & <code>nn_ahm_ijk_t</code>)	109
8.3.1. Mixing coefficients read from file (<code>=-20, -30</code>)	110
8.3.2. Constant mixing coefficients (<code>=0</code>)	110
8.3.3. Vertically varying mixing coefficients (<code>=10</code>)	110
8.3.4. Mesh size dependent mixing coefficients (<code>=20</code>)	110
8.3.5. Mesh size and depth dependent mixing coefficients (<code>=30</code>)	110
8.3.6. Velocity dependent mixing coefficients (<code>=31</code>)	111
8.3.7. Deformation rate dependent viscosities (<code>nn_ahm_ijk_t=32</code>)	111
8.3.8. About space and time varying mixing coefficients	111
8.4. Eddy induced velocity (<code>ln_ldfeiv</code>)	111
8.5. Mixed layer eddies (<code>ln_mle</code>)	112

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

The lateral physics terms in the momentum and tracer equations have been described in [equation 1.17](#) and their discrete formulation in [section 4.2](#) and [section 5.6](#)). In this section we further discuss each lateral physics option. Choosing one lateral physics scheme means for the user defining, (1) the type of operator used (laplacian or bilaplacian operators, or no lateral mixing term); (2) the direction along which the lateral diffusive fluxes are evaluated (model level, geopotential or isopycnal surfaces); and (3) the space and time variations of the eddy coefficients. These three aspects of the lateral diffusion are set through namelist parameters (see the `&namtra_ldf` ([namelist 4.2](#)) and `&namdyn_ldf` ([namelist 5.5](#)) below). Note that this chapter describes the standard implementation of iso-neutral tracer mixing. Griffies's implementation, which is used if `ln_traldf_triad=.true.`, is described in [appendix D](#)

8.1. Lateral mixing operators

We remind here the different lateral mixing operators that can be used. Further details can be found in [subsection 4.2.1](#) and [section 5.6](#).

8.1.1. No lateral mixing (`ln_traldf_OFF` & `ln_dynldf_OFF`)

It is possible to run without explicit lateral diffusion on tracers (`ln_traldf_OFF=.true.`) and/or momentum (`ln_dynldf_OFF=.true.`). The latter option is even recommended if using the UBS advection scheme on momentum (`ln_dynadv_ubs=.true.`, see [section 5.3.2](#)) and can be useful for testing purposes.

8.1.2. Laplacian mixing (`ln_traldf_lap` & `ln_dynldf_lap`)

Setting `ln_traldf_lap=.true.` and/or `ln_dynldf_lap=.true.` enables a second order diffusion on tracers and momentum respectively. Note that in *NEMO 4*, one can not combine Laplacian and Bilaplacian operators for the same variable.

8.1.3. Bilaplacian mixing (`ln_traldf_blp` & `ln_dynldf_blp`)

Setting `ln_traldf_blp=.true.` and/or `ln_dynldf_blp=.true.` enables a fourth order diffusion on tracers and momentum respectively. It is implemented by calling the above Laplacian operator twice. We stress again that from *NEMO 4*, the simultaneous use Laplacian and Bilaplacian operators is not allowed.

8.2. Direction of lateral mixing (`ldfslp.F90`)

A direction for lateral mixing has to be defined when the desired operator does not act along the model levels. This occurs when (a) horizontal mixing is required on tracer or momentum (`ln_traldf_hor` or `ln_dynldf_hor`) in *s*- or mixed *s-z*- coordinates, and (b) isoneutral mixing is required whatever the vertical coordinate is. This direction of mixing is defined by its slopes in the *i*- and *j*-directions at the face of the cell of the quantity to be diffused. For a tracer, this leads to the following four slopes: r_{1u} , r_{1w} , r_{2v} , r_{2w} (see [equation 4.8](#)), while for momentum the slopes are r_{1t} , r_{1uw} , r_{2f} , r_{2vw} for *u* and r_{1f} , r_{1vw} , r_{2t} , r_{2vw} for *v*.

8.2.1. Slopes for tracer geopotential mixing in the *s*-coordinate

In *s*-coordinates, geopotential mixing (*i.e.* horizontal mixing) r_1 and r_2 are the slopes between the geopotential and computational surfaces. Their discrete formulation is found by locally solving [equation 4.8](#) when the diffusive fluxes in the three directions are set to zero and *T* is assumed to be horizontally uniform, *i.e.* a linear function of z_T , the depth of a *T*-point.

$$\begin{aligned}
 r_{1u} &= \frac{e_{3u}}{\left(e_{1u} \overline{\overline{e_{3w}^{i+1/2, k}}} \right)} \delta_{i+1/2}[z_t] && \approx \frac{1}{e_{1u}} \delta_{i+1/2}[z_t] \\
 r_{2v} &= \frac{e_{3v}}{\left(e_{2v} \overline{\overline{e_{3w}^{j+1/2, k}}} \right)} \delta_{j+1/2}[z_t] && \approx \frac{1}{e_{2v}} \delta_{j+1/2}[z_t] \\
 r_{1w} &= \frac{1}{e_{1w}} \overline{\overline{\delta_{i+1/2}[z_t]}}^{i, k+1/2} && \approx \frac{1}{e_{1w}} \delta_{i+1/2}[z_{uw}] \\
 r_{2w} &= \frac{1}{e_{2w}} \overline{\overline{\delta_{j+1/2}[z_t]}}^{j, k+1/2} && \approx \frac{1}{e_{2w}} \delta_{j+1/2}[z_{vw}]
 \end{aligned} \tag{8.1}$$

These slopes are computed once in `ldf_slp_init` when `ln_sco=.true.`, and either `ln_traldf_hor=.true.` or `ln_dynldf_hor=.true.`

8.2.2. Slopes for tracer iso-neutral mixing

In iso-neutral mixing r_1 and r_2 are the slopes between the iso-neutral and computational surfaces. Their formulation does not depend on the vertical coordinate used. Their discrete formulation is found using the fact that the diffusive fluxes of locally referenced potential density (*i.e.* *insitu* density) vanish. So, substituting T by ρ in [equation 4.8](#) and setting the diffusive fluxes in the three directions to zero leads to the following definition for the neutral slopes:

$$\begin{aligned}
 r_{1u} &= \frac{e_{3u}}{e_{1u}} \frac{\delta_{i+1/2}[\rho]}{\overline{\delta_{k+1/2}[\rho]}^{i+1/2, k}} \\
 r_{2v} &= \frac{e_{3v}}{e_{2v}} \frac{\delta_{j+1/2}[\rho]}{\overline{\delta_{k+1/2}[\rho]}^{j+1/2, k}} \\
 r_{1w} &= \frac{e_{3w}}{e_{1w}} \frac{\overline{\delta_{i+1/2}[\rho]}^{i, k+1/2}}{\overline{\delta_{k+1/2}[\rho]}} \\
 r_{2w} &= \frac{e_{3w}}{e_{2w}} \frac{\overline{\delta_{j+1/2}[\rho]}^{j, k+1/2}}{\overline{\delta_{k+1/2}[\rho]}}
 \end{aligned} \tag{8.2}$$

As the mixing is performed along neutral surfaces, the gradient of ρ in [equation 8.2](#) has to be evaluated at the same local pressure (which, in decibars, is approximated by the depth in meters in the model). Therefore [equation 8.2](#) cannot be used as such, but further transformation is needed depending on the vertical coordinate used:

z -coordinate with full step: in [equation 8.2](#) 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 [McDougall \(1987\)](#) (see [subsection 4.8.2](#)).

z -coordinate with partial step: this case is identical to the full step case except that at partial step level, the *horizontal* density gradient is evaluated as described in [section 4.9](#).

s - or hybrid s - z - coordinate: in the current release of *NEMO*, iso-neutral mixing is only employed for s -coordinates if the Griffies scheme is used (`ln_tralddf_triad=.true.`; see [appendix D](#)). In other words, iso-neutral mixing will only be accurately represented with a linear equation of state (`ln_seos=.true.`). In the case of a "true" equation of state, the evaluation of i and j derivatives in [equation 8.2](#) will include a pressure dependent part, leading to the wrong evaluation of the neutral slopes.

Note: The solution for s -coordinate passes through the use of different (and better) expression for the constraint on iso-neutral fluxes. Following [Griffies \(2004\)](#), instead of specifying directly that there is a zero neutral diffusive flux of locally referenced potential density, we stay in the T - S plane and consider the balance between the neutral direction diffusive fluxes of potential temperature and salinity:

$$\alpha \mathbf{F}(T) = \beta \mathbf{F}(S)$$

This constraint leads to the following definition for the slopes:

$$\begin{aligned}
 r_{1u} &= \frac{e_{3u}}{e_{1u}} \frac{\alpha_u \delta_{i+1/2}[T] - \beta_u \delta_{i+1/2}[S]}{\alpha_u \overline{\delta_{k+1/2}[T]}^{i+1/2, k} - \beta_u \overline{\delta_{k+1/2}[S]}^{i+1/2, k}} \\
 r_{2v} &= \frac{e_{3v}}{e_{2v}} \frac{\alpha_v \delta_{j+1/2}[T] - \beta_v \delta_{j+1/2}[S]}{\alpha_v \overline{\delta_{k+1/2}[T]}^{j+1/2, k} - \beta_v \overline{\delta_{k+1/2}[S]}^{j+1/2, k}} \\
 r_{1w} &= \frac{e_{3w}}{e_{1w}} \frac{\alpha_w \overline{\delta_{i+1/2}[T]}^{i, k+1/2} - \beta_w \overline{\delta_{i+1/2}[S]}^{i, k+1/2}}{\alpha_w \overline{\delta_{k+1/2}[T]} - \beta_w \overline{\delta_{k+1/2}[S]}} \\
 r_{2w} &= \frac{e_{3w}}{e_{2w}} \frac{\alpha_w \overline{\delta_{j+1/2}[T]}^{j, k+1/2} - \beta_w \overline{\delta_{j+1/2}[S]}^{j, k+1/2}}{\alpha_w \overline{\delta_{k+1/2}[T]} - \beta_w \overline{\delta_{k+1/2}[S]}}
 \end{aligned}$$

where α and β , the thermal expansion and saline contraction coefficients introduced in [subsection 4.8.2](#), have to be evaluated at the three velocity points. In order to save computation time, they should be approximated by the mean of their values at T -points (for example in the case of α : $\alpha_u = \overline{\alpha_T}^{i+1/2}$, $\alpha_v = \overline{\alpha_T}^{j+1/2}$ and $\alpha_w = \overline{\alpha_T}^{k+1/2}$).

Note that such a formulation could be also used in the z -coordinate and z -coordinate with partial steps cases.

This implementation is a rather old one. It is similar to the one proposed by Cox (1987), except for the background horizontal diffusion. Indeed, the Cox (1987) 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 (Weaver and Eby, 1997; Griffies et al., 1998). Griffies's scheme is now available in NEMO if `ln_traldf_triad=.true.`; see appendix D. Here, another strategy is presented (Lazar, 1997): a local filtering of the iso-neutral slopes (made on 9 grid-points) prevents the development of grid point noise generated by the iso-neutral diffusion operator (figure 8.1). 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 ($2 \Delta x$) 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 Griffies et al. (1998) operator which has that property.

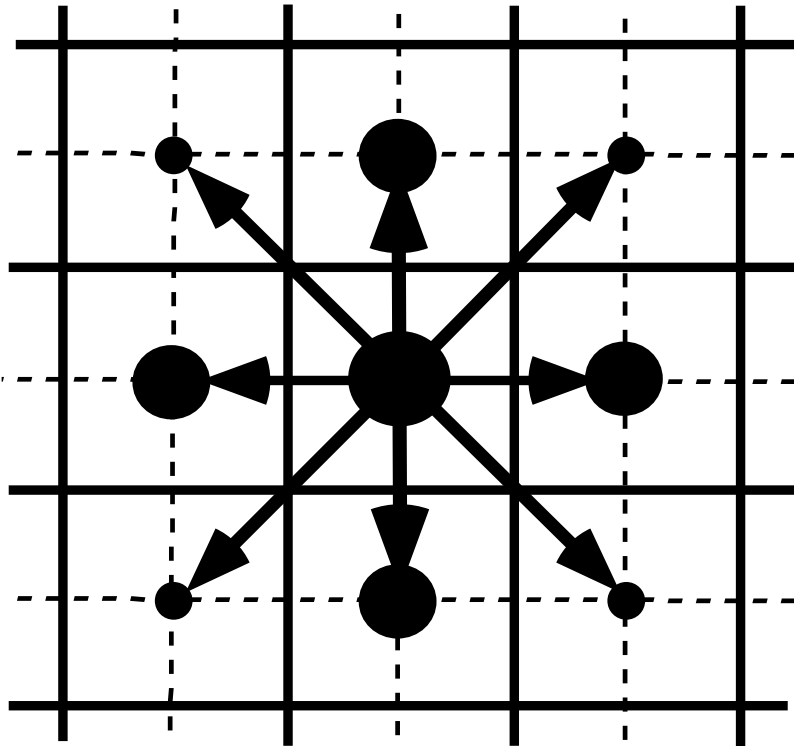


Figure 8.1.: Averaging procedure for isopycnal slope computation

For numerical stability reasons (Cox, 1987; Griffies, 2004), the slopes must also be bounded by the namelist scalar `rn_slpmax` (usually 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). The way slopes are tapered has been checked. Not sure that this is still add here a discussion about the flattening of the slopes, vs tapering the coefficient.

8.2.3. Slopes for momentum iso-neutral mixing

The iso-neutral diffusion operator on momentum is the same as the one used on tracers but applied to each component of the velocity separately (see equation 5.19 in section subsection 5.6.2). 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.* equation 8.1 and equation 8.2:

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

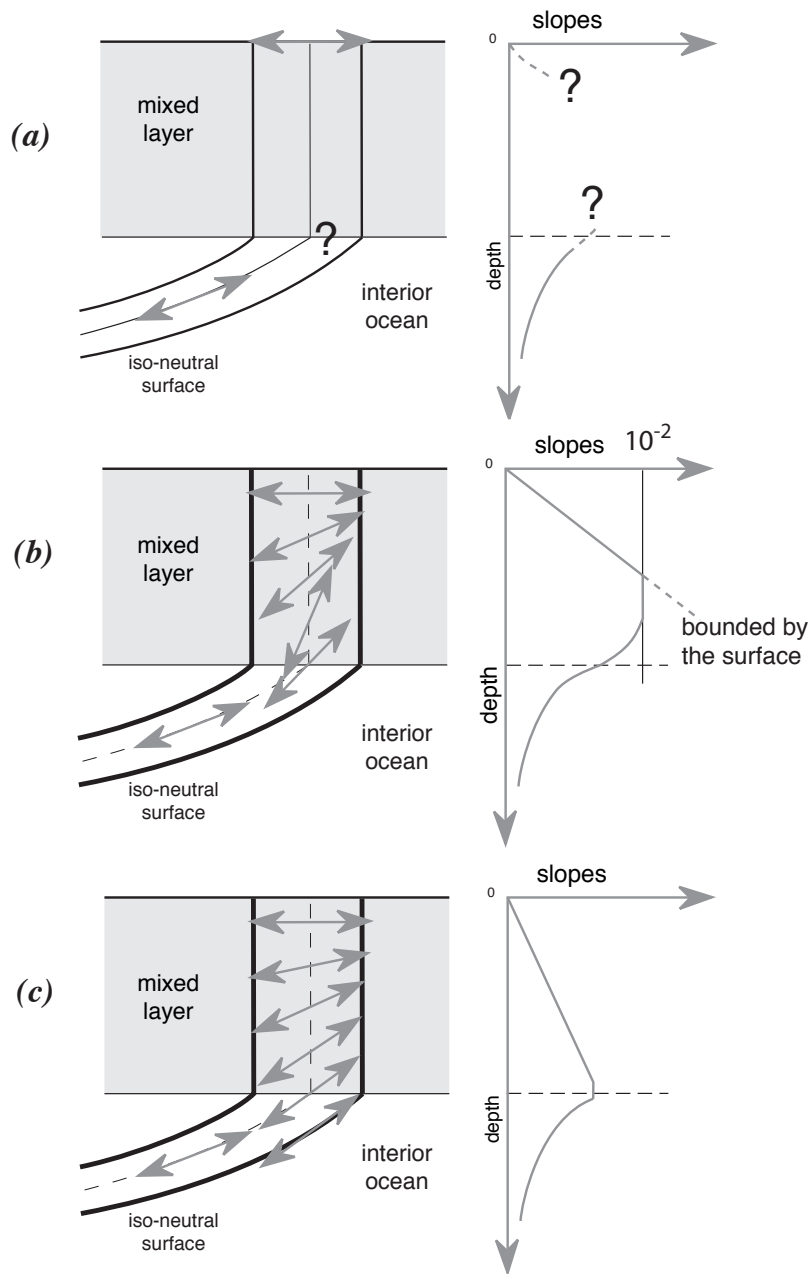


Figure 8.2.: Vertical profile of the slope used for lateral mixing in the mixed layer: (a) in the real ocean the slope is the iso-neutral slope in the ocean interior, which has to be adjusted at the surface boundary *i.e.* it must tend to zero at the surface since there is no mixing across the air-sea interface: wall boundary condition). Nevertheless, the profile between the surface zero value and the interior iso-neutral one is unknown, and especially the value at the base of the mixed layer; (b) profile of slope using a linear tapering of the slope near the surface and imposing a maximum slope of 1/100; (c) profile of slope actually used in *NEMO*: a linear decrease of the slope from zero at the surface to its ocean interior value computed just below the mixed layer. Note the huge change in the slope at the base of the mixed layer between (b) and (c).

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 section 7.1).

8.3. Lateral mixing coefficient (`nn_aht_ijk_t` & `nn_ahm_ijk_t`)

The specification of the space variation of the coefficient is made in modules `ldftra.F90` and `ldfdyn.F90`. The way the mixing coefficients are set in the reference version can be described as follows:

8.3.1. Mixing coefficients read from file (*nn_aht_ijk_t*=-20, -30 & *nn_ahm_ijk_t*=-20, -30)

Mixing coefficients can be read from file if a particular geographical variation is needed. For example, in the ORCA2 global ocean model, the laplacian viscosity operator uses $A^l = 4.10^4 \text{ m}^2/\text{s}$ poleward of 20° north and south and decreases linearly to $A^l = 2.10^3 \text{ m}^2/\text{s}$ at the equator (Madec et al., 1996; Delécluse and Madec, 1999). Similar modified horizontal variations can be found with the Antarctic or Arctic sub-domain options of ORCA2 and ORCA05. The provided fields can either be 2d (*nn_aht_ijk_t*=-20, *nn_ahm_ijk_t*=-20) or 3d (*nn_aht_ijk_t*=-30, *nn_ahm_ijk_t*=-30). They must be given at U, V points for tracers and T, F points for momentum (see table 8.1).

Namelist parameter	Input filename	dimensions	variable names
<i>nn_ahm_ijk_t</i> =-20	<code>eddy_viscosity_2D.nc</code>	(<i>i, j</i>)	<i>ahmt_2d</i> , <i>ahmf_2d</i>
<i>nn_aht_ijk_t</i> =-20	<code>eddy_diffusivity_2D.nc</code>	(<i>i, j</i>)	<i>ahtu_2d</i> , <i>ahtv_2d</i>
<i>nn_ahm_ijk_t</i> =-30	<code>eddy_viscosity_3D.nc</code>	(<i>i, j, k</i>)	<i>ahmt_3d</i> , <i>ahmf_3d</i>
<i>nn_aht_ijk_t</i> =-30	<code>eddy_diffusivity_3D.nc</code>	(<i>i, j, k</i>)	<i>ahtu_3d</i> , <i>ahtv_3d</i>

Table 8.1.: Description of expected input files if mixing coefficients are read from NetCDF files

8.3.2. Constant mixing coefficients (*nn_aht_ijk_t*=0 & *nn_ahm_ijk_t*=0)

If constant, mixing coefficients are set thanks to a velocity and a length scales (U_{scl} , L_{scl}) such that:

$$A_o^l = \begin{cases} \frac{1}{2} U_{scl} L_{scl} & \text{for laplacian operator} \\ \frac{1}{12} U_{scl} L_{scl}^3 & \text{for bilaplacian operator} \end{cases} \quad (8.3)$$

U_{scl} and L_{scl} are given by the namelist parameters *rn_Ud*, *rn_Uv*, *rn_Ld* and *rn_Lv*.

8.3.3. Vertically varying mixing coefficients (*nn_aht_ijk_t*=10 & *nn_ahm_ijk_t*=10)

In the vertically varying case, a hyperbolic variation of the lateral mixing coefficient is introduced in which the surface value is given by equation 8.3, the bottom value is 1/4 of the surface value, and the transition takes place around $z=500$ m with a width of 200 m. This profile is hard coded in module *ldfc1d_c2d.F90*, but can be easily modified by users.

8.3.4. Mesh size dependent mixing coefficients (*nn_aht_ijk_t*=20 & *nn_ahm_ijk_t*=20)

In that case, the horizontal variation of the eddy coefficient depends on the local mesh size and the type of operator used:

$$A_l = \begin{cases} \frac{1}{2} U_{scl} \max(e_1, e_2) & \text{for laplacian operator} \\ \frac{1}{12} U_{scl} \max(e_1, e_2)^3 & \text{for bilaplacian operator} \end{cases} \quad (8.4)$$

where U_{scl} is a user defined velocity scale (*rn_Ud*, *rn_Uv*). This variation is intended to reflect the lesser need for subgrid scale eddy mixing where the grid size is smaller in the domain. It was introduced in the context of the DYNAMO modelling project (Willebrand et al., 2001). Note that such a grid scale dependance of mixing coefficients significantly increases the range of stability of model configurations presenting large changes in grid spacing 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 section 2.3), especially when using a bilaplacian operator.

CASE *nn_aht_ijk_t* = 21 to be added

8.3.5. Mesh size and depth dependent mixing coefficients (*nn_aht_ijk_t*=30 & *nn_ahm_ijk_t*=30)

The 3D space variation of the mixing coefficient is simply the combination of the 1D and 2D cases above, *i.e.* a hyperbolic tangent variation with depth associated with a grid size dependence of the magnitude of the coefficient.

8.3.6. Flow dependent mixing coefficients (`nn_aht_ijk_t=31` & `nn_ahm_ijk_t=31`)

In that case, the eddy coefficient is proportional to the local velocity magnitude so that the Reynolds number $Re = |U|e/A_l$ is constant (and here hardcoded to 12): **JC comment: The Reynolds is effectively set to 12 in the code for both operators but shouldn't it**

$$A_l = \begin{cases} \frac{1}{12}|U|e & \text{for laplacian operator} \\ \frac{1}{12}|U|e^3 & \text{for bilaplacian operator} \end{cases} \quad (8.5)$$

8.3.7. Deformation rate dependent viscosities (`nn_ahm_ijk_t=32`)

This option refers to the (Smagorinsky, 1963) scheme which is here implemented for momentum only. Smagorinsky chose as a characteristic time scale T_{smag} the deformation rate and for the lengthscale L_{smag} the maximum wavenumber possible on the horizontal grid, e.g.:

$$T_{smag}^{-1} = \sqrt{(\partial_x u - \partial_y v)^2 + (\partial_y u + \partial_x v)^2}$$

$$L_{smag} = \frac{1}{\pi} \frac{e_1 e_2}{e_1 + e_2} \quad (8.6)$$

Introducing a user defined constant C (given in the namelist as `rn_csmc`), one can deduce the mixing coefficients as follows:

$$A_{smag} = \begin{cases} C^2 T_{smag}^{-1} L_{smag}^2 & \text{for laplacian operator} \\ \frac{C^2}{8} T_{smag}^{-1} L_{smag}^4 & \text{for bilaplacian operator} \end{cases} \quad (8.7)$$

For stability reasons, upper and lower limits are applied on the resulting coefficient (see section 2.3) so that:

$$C_{min} \frac{1}{2} |U|e < A_{smag} < C_{max} \frac{e^2}{8\Delta t} \quad \text{for laplacian operator}$$

$$C_{min} \frac{1}{12} |U|e^3 < A_{smag} < C_{max} \frac{e^4}{64\Delta t} \quad \text{for bilaplacian operator} \quad (8.8)$$

where C_{min} and C_{max} are adimensional namelist parameters given by `rn_minfac` and `rn_maxfac` respectively.

8.3.8. About space and time varying mixing coefficients

The following points are relevant when the eddy coefficient varies spatially:

(1) the momentum diffusion operator acting along model level surfaces is written in terms of curl and divergent components of the horizontal current (see subsection 1.5.2). Although the eddy coefficient could be set to different values in these two terms, this option is not currently 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 (section C.7).

8.4. Eddy induced velocity (`ln_ldfeiv`)

When Gent and McWilliams (1990) diffusion is used (`ln_ldfeiv=.true.`), 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.* equation 8.1 is used in z -coordinates, and the sum equation 8.1 + equation 8.2 in s -coordinates.

If isopycnal mixing is used in the standard way, *i.e.* `ln_traldf_triad=.false.`, the eddy induced velocity is given by:

$$u^* = \frac{1}{e_{2u} e_{3u}} \delta_k \left[e_{2u} A_{uw}^{eiv} \overline{r_{1w}}^{i+1/2} \right]$$

$$v^* = \frac{1}{e_{1u} e_{3v}} \delta_k \left[e_{1v} A_{vw}^{eiv} \overline{r_{2w}}^{j+1/2} \right] \quad (8.9)$$

$$w^* = \frac{1}{e_{1w} e_{2w}} \left\{ \delta_i \left[e_{2u} A_{uw}^{eiv} \overline{r_{1w}}^{i+1/2} \right] + \delta_j \left[e_{1v} A_{vw}^{eiv} \overline{r_{2w}}^{j+1/2} \right] \right\}$$

```

!-----
&namtra_eiv ! eddy induced velocity param. (default: OFF)
!-----
ln_ldfeiv = .false. ! use eddy induced velocity parameterization
!
! Coefficients:
nn_aei_ijk_t = 0 ! space/time variation of eddy coefficient:
! ! =-20 (=30) read in eddy_induced_velocity_2D.nc (..._3D.nc) file
! ! = 0 constant
! ! = 10 F(k) =ldf_c1d
! ! = 20 F(i,j) =ldf_c2d
! ! = 21 F(i,j,t) =Treguier et al. JPO 1997 formulation
! ! = 30 F(i,j,k) =ldf_c2d * ldf_c1d
! ! time invariant coefficients: aei0 = 1/2 Ue*Le
rn_Ue = 0.02 ! lateral diffusive velocity [m/s] (nn_aht_ijk_t= 0, 10, 20, 30)
rn_Le = 200.e+3 ! lateral diffusive length [m] (nn_aht_ijk_t= 0, 10)
!
ln_ldfeiv_dia = .false. ! diagnose eiv stream function and velocities
/

```

namelist 8.1.: `&namtra_eiv`

```

!-----
&namtra_mle ! mixed layer eddy parametrisation (Fox-Kemper) (default: OFF)
!-----
ln_mle = .false. ! (T) use the Mixed Layer Eddy (MLE) parameterisation
rn_ce = 0.06 ! magnitude of the MLE (typical value: 0.06 to 0.08)
nn_mle = 1 ! MLE type: =0 standard Fox-Kemper ; =1 new formulation
rn_lf = 5.e+3 ! typical scale of mixed layer front (meters) (case rn_mle=0)
rn_time = 172800. ! time scale for mixing momentum across the mixed layer (seconds) (case rn_mle=0)
rn_lat = 20. ! reference latitude (degrees) of MLE coef. (case rn_mle=1)
nn_mld_uv = 0 ! space interpolation of MLD at u- & v-pts (0=min,1=averaged,2=max)
nn_conv = 0 ! =1 no MLE in case of convection ; =0 always MLE
rn_rho_c_mle = 0.01 ! delta rho criterion used to calculate MLD for FK
/

```

namelist 8.2.: `&namtra_mle`

where A^{eiv} is the eddy induced velocity coefficient whose value is set through `nn_aei_ijk_t` `&namtra_eiv` (namelist 8.1) namelist parameter. The three components of the eddy induced velocity are computed in `ldf_eiv_trp` and added to the eulerian velocity in `tra_adv` where tracer advection is performed. 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 section 4.1) and not just the 2^{nd} order advection scheme as in previous releases of OPA (Madec et al., 1998). 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. The value of the eddy induced mixing coefficient and its space variation is controlled in a similar way as for lateral mixing coefficient described in the preceding subsection (`nn_aei_ijk_t`, `rn_Ue`, `rn_Le` namelist parameters). **CASE `nn_aei_ijk_t = 21` to be added**

In case of setting `ln_traldf_triad=.true.`, a skew form of the eddy induced advective fluxes is used, which is described in appendix D.

8.5. Mixed layer eddies (`ln_mle`)

If `ln_mle=.true.` in `&namtra_mle` (namelist 8.2) namelist, a parameterization of the mixing due to unresolved mixed layer instabilities is activated (Fox-Kemper et al. (2008)). Additional transport is computed in `ldf_mle_trp` and added to the eulerian transport in `tra_adv` as done for eddy induced advection.

TBC

Table of contents

9.1. Vertical mixing	114
9.1.1. Constant (<code>ln_zdfcst</code>)	114
9.1.2. Richardson number dependent (<code>ln_zdfric</code>)	114
9.1.3. TKE turbulent closure scheme (<code>ln_zdftke</code>)	115
9.1.4. GLS: Generic Length Scale (<code>ln_zdfgls</code>)	119
9.1.5. OSM: OSMOSIS boundary layer scheme (<code>ln_zdfosm = .true.</code>)	120
9.1.6. Discrete energy conservation for TKE and GLS schemes	123
9.2. Convection	124
9.2.1. Non-penetrative convective adjustment (<code>ln_tranpc</code>)	125
9.2.2. Enhanced vertical diffusion (<code>ln_zdfevd</code>)	125
9.2.3. Handling convection with turbulent closure schemes (<code>ln_zdf_{tke, gls, osm}</code>)	126
9.3. Double diffusion mixing (<code>ln_zdfddm</code>)	126
9.4. Bottom and top friction (<code>zdfdrg.F90</code>)	127
9.4.1. Linear top/bottom friction (<code>ln_lin</code>)	128
9.4.2. Non-linear top/bottom friction (<code>ln_non_lin</code>)	129
9.4.3. Log-layer top/bottom friction (<code>ln_loglayer</code>)	129
9.4.4. Explicit top/bottom friction (<code>ln_drgimp=.false.</code>)	129
9.4.5. Implicit top/bottom friction (<code>ln_drgimp=.true.</code>)	130
9.4.6. Bottom friction with split-explicit free surface	130
9.5. Internal wave-driven mixing (<code>ln_zdfiwm</code>)	131
9.6. Surface wave-induced mixing (<code>ln_zdfswm</code>)	132
9.7. Adaptive-implicit vertical advection (<code>ln_zad_Aimp</code>)	132
9.7.1. Adaptive-implicit vertical advection in the OVERFLOW test-case	133

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

```

!-----
&namzdf      !   vertical physics manager                               (default: NO selection)
!-----
!
!   ! adaptive-implicit vertical advection
ln_zad_Aimp = .false.      ! Courant number dependent scheme (Shchepetkin 2015)
!
!   ! type of vertical closure (required)
ln_zdfcst   = .false.      ! constant mixing
ln_zdftric  = .false.      ! local Richardson dependent formulation (T => fill namzdf_ric)
ln_zdf_tke  = .false.      ! Turbulent Kinetic Energy closure (T => fill namzdf_tke)
ln_zdfgls   = .false.      ! Generic Length Scale closure (T => fill namzdf_gls)
ln_zdfosm   = .false.      ! OSMOSIS BL closure (T => fill namzdf_osm)
!
!   ! convection
ln_zdfevd   = .false.      ! enhanced vertical diffusion
nn_evdm     = 0             ! apply on tracer (=0) or on tracer and momentum (=1)
rn_evd      = 100.         ! mixing coefficient [m2/s]
ln_zdfnpc   = .false.      ! Non-Penetrative Convective algorithm
nn_npc      = 1            ! frequency of application of npc
nn_npcp     = 365          ! npc control print frequency
!
!   ! double diffusive mixing
ln_zdfddm   = .false.      ! double diffusive mixing
rn_avts     = 1.e-4        ! maximum avs (vertical mixing on salinity)
rn_hsbfr    = 1.6          ! heat/salt buoyancy flux ratio
!
!   ! gravity wave-driven vertical mixing
ln_zdfiwm   = .false.      ! internal wave-induced mixing (T => fill namzdf_iwm)
ln_zdfsww   = .false.      ! surface wave-induced mixing (T => ln_wave=ln_sdw=T )
!
!   ! coefficients
rn_avm0     = 1.2e-4        ! vertical eddy viscosity [m2/s] (background Kz if ln_zdfcst=F)
rn_avt0     = 1.2e-5        ! vertical eddy diffusivity [m2/s] (background Kz if ln_zdfcst=F)
nn_avb      = 0             ! profile for background avt & avm (=1) or not (=0)
nn_havtb    = 0             ! horizontal shape for avtb (=1) or not (=0)
/

```

namelist 9.1.: &namzdf

9.1. Vertical mixing

The discrete form of the ocean subgrid scale physics has been presented in [section 4.3](#) and [section 5.7](#). 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 [chapter 6](#)), while at the bottom they are set to zero for heat and salt, unless a geothermal flux forcing is prescribed as a bottom boundary condition (*i.e.* `ln_trabbc` defined, see [subsection 4.4.3](#)), and specified through a bottom friction parameterisation for momentum (see [section 9.4](#)).

In this section we briefly discuss the various choices offered to compute the vertical eddy viscosity and diffusivity coefficients, A_u^{vm} , A_v^{vm} and A^{vT} (A^{vS}), defined at uw -, vw - and w - points, respectively (see [section 4.3](#) and [section 5.7](#)). 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 GLS or OSMOSIS formulation). The computation of these coefficients is initialized in the `zdfphy.F90` module and performed in the `zdftric.F90`, `zdfike.F90` or `zdfgls.F90` or `zdfosm.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.

9.1.1. Constant (`ln_zdfcst`)

Options are defined through the `&namzdf` ([namelist 9.1](#)) namelist variables. When `ln_zdfcst` is defined, the momentum and tracer vertical eddy coefficients are set to constant values over the whole ocean. This is the crudest way to define the vertical ocean physics. It is recommended to use this option only in process studies, not in basin scale simulations. Typical values used in this case are:

$$A_u^{vm} = A_v^{vm} = 1.2 \cdot 10^{-4} \text{ m}^2 \cdot \text{s}^{-1}$$

$$A^{vT} = A^{vS} = 1.2 \cdot 10^{-5} \text{ m}^2 \cdot \text{s}^{-1}$$

These values are set through the `rn_avm0` and `rn_avt0` namelist parameters. In all cases, do not use values smaller than those associated with the molecular viscosity and diffusivity, that is $\sim 10^{-6} \text{ m}^2 \cdot \text{s}^{-1}$ for momentum, $\sim 10^{-7} \text{ m}^2 \cdot \text{s}^{-1}$ for temperature and $\sim 10^{-9} \text{ m}^2 \cdot \text{s}^{-1}$ for salinity.

9.1.2. Richardson number dependent (`ln_zdftric`)

```

!-----
&namzdf_ric ! richardson number dependent vertical diffusion (ln_zdf_ric =T)
!-----
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
ln_mldw = .false. ! enhanced mixing in the Ekman layer
rn_ekmfc = 0.7 ! Factor in the Ekman depth Equation
rn_mldmin = 1.0 ! minimum allowable mixed-layer depth estimate (m)
rn_mldmax = 1000.0 ! maximum allowable mixed-layer depth estimate (m)
rn_wtmix = 10.0 ! vertical eddy viscosity coeff [m2/s] in the mixed-layer
rn_wvmix = 10.0 ! vertical eddy diffusion coeff [m2/s] in the mixed-layer
/
    
```

namelist 9.2.: &namzdf_ric

When `ln_zdf_ric=.true.`, a local Richardson number dependent formulation for the vertical momentum and tracer eddy coefficients is set through the `&namzdf_ric` (namelist 9.2) namelist variables. 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 Pacanowski and Philander (1981), the following formulation has been implemented:

$$\begin{cases} A^{vT} = \frac{A_{ric}^{vT}}{(1+a Ri)^n} + A_b^{vT} \\ A^{vm} = \frac{A^{vT}}{(1+a Ri)} + A_b^{vm} \end{cases}$$

where $Ri = N^2 / (\partial_z \mathbf{U}_h)^2$ is the local Richardson number, N is the local Brunt-Vaisälä frequency (see subsection 4.8.2), A_b^{vT} and A_b^{vm} are the constant background values set as in the constant case (see subsection 9.1.1), and $A_{ric}^{vT} = 10^{-4} m^2 \cdot s^{-1}$ 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.

A simple mixing-layer model to transfer and dissipate the atmospheric forcings (wind-stress and buoyancy fluxes) can be activated setting the `ln_mldw=.true.` in the namelist.

In this case, the local depth of turbulent wind-mixing or "Ekman depth" $h_e(x, y, t)$ is evaluated and the vertical eddy coefficients prescribed within this layer.

This depth is assumed proportional to the "depth of frictional influence" that is limited by rotation:

$$h_e = Ek \frac{u^*}{f_0}$$

where, Ek is an empirical parameter, u^* is the friction velocity and f_0 is the Coriolis parameter.

In this similarity height relationship, the turbulent friction velocity:

$$u^* = \sqrt{\frac{|\tau|}{\rho_o}}$$

is computed from the wind stress vector $|\tau|$ and the reference density ρ_o . The final h_e is further constrained by the adjustable bounds `rn_mldmin` and `rn_mldmax`. Once h_e is computed, the vertical eddy coefficients within h_e are set to the empirical values `rn_wtmix` and `rn_wvmix` (Lermusiaux, 2001).

9.1.3. TKE turbulent closure scheme (`ln_zdftke`)

The vertical eddy viscosity and diffusivity coefficients are computed from a TKE turbulent closure model based on a prognostic equation for \bar{e} , the turbulent kinetic energy, and a closure assumption for the turbulent length scales. This turbulent closure model has been developed by Bougeault and Lacarrere (1989) in the atmospheric case, adapted by Gaspar et al. (1990) for the oceanic case, and embedded in OPA, the ancestor of NEMO, by Blanke and Delécluse (1993) for equatorial Atlantic simulations. Since then, significant modifications have been introduced by Madec et al. (1998) in both the implementation and the formulation of the mixing length scale. The time evolution of \bar{e} is the result of the production of \bar{e} through vertical shear, its destruction through stratification, its vertical diffusion, and its dissipation of Kolmogorov (1942) type:

$$\frac{\partial \bar{e}}{\partial t} = \frac{K_m}{e_3^2} \left[\left(\frac{\partial u}{\partial k} \right)^2 + \left(\frac{\partial v}{\partial k} \right)^2 \right] - K_\rho N^2 + \frac{1}{e_3} \frac{\partial}{\partial k} \left[\frac{A^{vm}}{e_3} \frac{\partial \bar{e}}{\partial k} \right] - c_\epsilon \frac{\bar{e}^{3/2}}{l_\epsilon} \quad (9.1)$$

```

!-----
&namzdf_tke ! turbulent eddy kinetic dependent vertical diffusion (ln_zdf_tke =T)
!-----
rn_ediff    = 0.1 ! coef. for vertical eddy coef. (avt=rn_ediff*mxl*sqrt(e) )
rn_ediss    = 0.7 ! coef. of the Kolmogoroff dissipation
rn_ebb      = 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]
rn_bshear   = 1.e-20 ! background shear (>0) currently a numerical threshold (do not change it)
nn_pdl      = 1 ! Prandtl number function of richarson number (=1, avt=pdl(Ri)*avm) or not (=0,
↳ avt=avm)
nn_mxl      = 2 ! mixing length: = 0 bounded by the distance to surface and bottom
! ! !
! ! !
! ! !
ln_mxl0     = .true. ! surface mixing length scale = F(wind stress) (T) or not (F)
rn_mxl0     = 0.04 ! surface buoyancy length scale minimum value
ln_drg      = .false. ! top/bottom friction added as boundary condition of TKE
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 NIWs
! ! !
! ! !
! ! !
rnEFR       = 0.05 ! fraction of surface tke value which penetrates below the ML (ln_cpl=T)
nn_htau     = 1 ! type of exponential decrease of tke penetration below the ML
! ! !
! ! !
rn_eice     = 4 ! below sea ice: =0 ON ; =4 OFF when ice fraction > 1/4
/

```

namelist 9.3.: &namzdf_tke

$$K_m = C_k l_k \sqrt{\bar{\epsilon}}$$

$$K_\rho = A^{vm} / P_{rt}$$

where N is the local Brunt-Vaisälä frequency (see subsection 4.8.2), l_ϵ and l_k 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 (Gaspar et al., 1990). They are set through namelist parameters `nn_ediff` and `nn_ediss`. P_{rt} can be set to unity or, following Blanke and Delécluse (1993), be a function of the local Richardson number, R_i :

$$P_{rt} = \begin{cases} 1 & \text{if } R_i \leq 0.2 \\ 5 R_i & \text{if } 0.2 \leq R_i \leq 2 \\ 10 & \text{if } 2 \leq R_i \end{cases}$$

The choice of P_{rt} is controlled by the `nn_pdl` namelist variable.

At the sea surface, the value of $\bar{\epsilon}$ is prescribed from the wind stress field as $\bar{\epsilon}_o = e_{bb} |\tau| / \rho_o$, with e_{bb} the `rn_ebb` namelist parameter. The default value of e_{bb} is 3.75. (Gaspar et al., 1990)), however a much larger value can be used when taking into account the surface wave breaking (see below Eq. equation 9.4). The bottom value of TKE is assumed to be equal to the value of the level just above. The time integration of the $\bar{\epsilon}$ 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 $\bar{\epsilon}$ is used (`rn_emin` namelist parameter). Following Gaspar et al. (1990), the cut-off value is set to $\sqrt{2}/2 \cdot 10^{-6} \text{ m}^2 \cdot \text{s}^{-2}$. This allows the subsequent formulations to match that of Gargett (1984) 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 9.1) namelist, see subsection 9.1.1).

Turbulent length scale

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

$$l_k = l_\epsilon = \sqrt{2\bar{\epsilon}} / N \quad (9.2)$$

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`). Blanke and Delécluse (1993) notice that this simplification has two major drawbacks: it makes no sense for locally

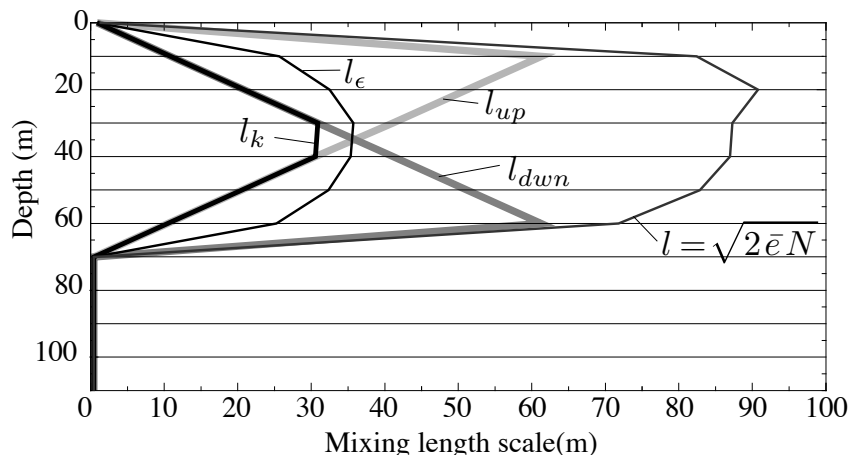


Figure 9.1.: Illustration of the mixing length computation

unstable stratification and the computation no longer uses all the information contained in the vertical density profile. To overcome these drawbacks, Madec et al. (1998) introduces the `nn_mxl=2, 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 equation 9.2 and then bounded such that:

$$\frac{1}{e_3} \left| \frac{\partial l}{\partial k} \right| \leq 1 \quad \text{with } l = l_k = l_\epsilon \quad (9.3)$$

equation 9.3 means that the vertical variations of the length scale cannot be larger than the variations of depth. It provides a better approximation of the Gaspar et al. (1990) formulation while being much less time consuming. In particular, it allows the length scale to be limited not only by the distance to the surface or to the ocean bottom but also by the distance to a strongly stratified portion of the water column such as the thermocline (figure 9.1). In order to impose the equation 9.3 constraint, we introduce two additional length scales: l_{up} and l_{down} , the upward and downward length scales, and evaluate the dissipation and mixing length scales as (and note that here we use numerical indexing):

$$l_{up}^{(k)} = \min \left(l^{(k)}, l_{up}^{(k+1)} + e_{3t}^{(k)} \right) \quad \text{from } k = 1 \text{ to } jpk$$

$$l_{down}^{(k)} = \min \left(l^{(k)}, l_{down}^{(k-1)} + e_{3t}^{(k-1)} \right) \quad \text{from } k = jpk \text{ to } 1$$

where $l^{(k)}$ is computed using equation 9.2, i.e. $l^{(k)} = \sqrt{2\bar{\epsilon}^{(k)}/N^{2(k)}}$.

In the `nn_mxl=2` case, the dissipation and mixing length scales take the same value: $l_k = l_\epsilon = \min(l_{up}, l_{down})$, while in the `nn_mxl=3` case, the dissipation and mixing turbulent length scales are give as in Gaspar et al. (1990):

$$l_k = \sqrt{l_{up} l_{down}}$$

$$l_\epsilon = \min(l_{up}, l_{down})$$

At the ocean surface, a non zero length scale is set through the `rn_mxl0` 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 (Craig and Banner, 1994) leads to a 0.04 m, the default value of `rn_mxl0`. In the ocean interior a minimum length scale is set to recover the molecular viscosity when $\bar{\epsilon}$ reach its minimum value ($1.10^{-6} = C_k l_{min} \sqrt{\bar{\epsilon}_{min}}$).

Surface wave breaking parameterization

Following Mellor and Blumberg (2004), 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 Mellor and Blumberg (2004) modifications acts on surface length scale and TKE values and air-sea drag coefficient. The latter concerns the bulk formulae and is not discussed here.

Following Craig and Banner (1994), the boundary condition on surface TKE value is :

$$\bar{\epsilon}_o = \frac{1}{2} (15.8 \alpha_{CB})^{2/3} \frac{|\tau|}{\rho_o} \quad (9.4)$$

where α_{CB} is the Craig and Banner (1994) constant of proportionality which depends on the "wave age", ranging from 57 for mature waves to 146 for younger waves (Mellor and Blumberg, 2004). The boundary condition on the turbulent length

scale follows the Charnock's relation:

$$l_o = \kappa \beta \frac{|\tau|}{g \rho_o} \quad (9.5)$$

where $\kappa = 0.40$ is the von Karman constant, and β is the Charnock's constant. Mellor and Blumberg (2004) suggest $\beta = 2.10^5$ the value chosen by Stacey (1999) 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_mx10=.true.`, applies equation 9.5 as the surface boundary condition on the length scale, with β hard coded to the Stacey's value. Note that a minimal threshold of `rn_emin0 = 10-4 m2.s-2` (namelist parameters) is applied on the 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, Craik and Leibovich (1976). 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 (Axell, 2002) for a $k - \epsilon$ turbulent closure. The parameterization, tuned against large-eddy simulation, includes the whole effect of LC in an extra source term of TKE, P_{LC} . The presence of P_{LC} in equation 9.1, the TKE equation, is controlled by setting `ln_lc` to `.true.` in the `&namzdf_tke` (namelist 9.3) namelist.

By making an analogy with the characteristic convective velocity scale (e.g., D'Alessio et al. (1998)), P_{LC} is assumed to be :

$$P_{LC}(z) = \frac{w_{LC}^3(z)}{H_{LC}}$$

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*. 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 :

$$w_{LC} = \begin{cases} c_{LC} u_s \sin(-\pi z / H_{LC}) & \text{if } -z \leq H_{LC} \\ 0 & \text{otherwise} \end{cases}$$

where $c_{LC} = 0.15$ has been chosen by (Axell, 2002) 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 (Axell, 2002).

The H_{LC} is estimated in a similar way as the turbulent length scale of TKE equations: H_{LC} is the 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

$$- \int_{-H_{LC}}^0 N^2 z dz = \frac{1}{2} u_s^2$$

Mixing just below the mixed layer

Vertical mixing parameterizations commonly used in ocean general circulation models tend to produce mixed-layer depths that are too shallow during summer months and windy conditions. This bias is particularly acute over the Southern Ocean. To overcome this systematic bias, an ad hoc parameterization is introduced into the TKE scheme Rodgers et al. (2014). The parameterization is an empirical one, *i.e.* not derived from theoretical considerations, but rather is meant to account for observed processes that affect the density structure of the oceans planetary boundary layer that are not explicitly captured by default in the TKE scheme (*i.e.* near-inertial oscillations and ocean swells and waves).

When using this parameterization (*i.e.* when `nn_etau=1`), the TKE input to the ocean (S) imposed by the winds in the form of near-inertial oscillations, swell and waves is parameterized by equation 9.4 the standard TKE surface boundary condition, plus a depth depend one given by:

$$S = (1 - f_i) f_r e_s e^{-z/h_\tau} \quad (9.6)$$

where z is the depth, e_s is TKE surface boundary condition, f_r is the fraction of the surface TKE that penetrates in the ocean, h_τ is a vertical mixing length scale that controls exponential shape of the penetration, and f_i is the ice concentration (no penetration if $f_i = 1$, *i.e.* if the ocean is entirely covered by sea-ice). The value of f_r , usually a few percents, is

*Following Li and Garrett (1993), the surface Stoke drift velocity may be expressed as $u_s = 0.016 |U_{10m}|$. Assuming an air density of $\rho_a = 1.22 \text{ Kg/m}^3$ and a drag coefficient of $1.5 \cdot 10^{-3}$ give the expression used of u_s as a function of the module of surface stress

```

!-----
&namzdf_gls      !   GLS vertical diffusion                               (ln_zdfgls =T)
!-----
rn_emin          = 1.e-7   !   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     = 0.267   !   galperin limit
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
rn_hsro          = 0.02   !   Minimum surface roughness
rn_frac_hs       = 1.3    !   Fraction of wave height as roughness (if nn_z0_met>1)
nn_z0_met        = 2      !   Method for surface roughness computation (0/1/2/3)
!               !   =3 requires ln_wave=T
nn_bc_surf       = 1      !   surface condition (0/1=Dir/Neum)
nn_bc_bot        = 1      !   bottom 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)
/
    
```

namelist 9.4.: &namzdf_gls

specified through `rn_efr` namelist parameter. The vertical mixing length scale, h_τ , can be set as a 10 m uniform value (`nn_etau=0`) or a latitude dependent value (varying from 0.5 m at the Equator to a maximum value of 30 m at high latitudes (`nn_etau=1`)).

Note that two other option exist, `nn_etau=2, 3`. They correspond to applying [equation 9.6](#) only at the base of the mixed layer, or to using the high frequency part of the stress to evaluate the fraction of TKE that penetrates the ocean. Those two options are obsolescent features introduced for test purposes. They will be removed in the next release.

In presence of Sea Ice, the value of this mixing can be modulated by the `rn_eice` namelist parameter. This parameter varies from 0 for no effect to 4 to suppress the TKE input into the ocean when Sea Ice concentration is greater than 25%.

9.1.4. GLS: Generic Length Scale (`ln_zdfgls`)

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 ([Umlauf and Burchard, 2003, 2005](#)). 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.table 9.1](#) allows to recover a number of well-known turbulent closures (k - kl ([Mellor and Yamada, 1982](#)), k - ϵ ([Rodi, 1987](#)), k - ω ([Wilcox, 1988](#)) among others ([Umlauf and Burchard, 2003; Kantha and Carniel, 2003](#))). The GLS scheme is given by the following set of equations:

$$\frac{\partial \bar{e}}{\partial t} = \frac{K_m}{\sigma_e e_3} \left[\left(\frac{\partial u}{\partial k} \right)^2 + \left(\frac{\partial v}{\partial k} \right)^2 \right] - K_\rho N^2 + \frac{1}{e_3} \frac{\partial}{\partial k} \left[\frac{K_m}{e_3} \frac{\partial \bar{e}}{\partial k} \right] - \epsilon \quad (9.7)$$

$$\frac{\partial \psi}{\partial t} = \frac{\psi}{\bar{e}} \left\{ \frac{C_1 K_m}{\sigma_\psi e_3} \left[\left(\frac{\partial u}{\partial k} \right)^2 + \left(\frac{\partial v}{\partial k} \right)^2 \right] - C_3 K_\rho N^2 - C_2 \epsilon Fw \right\} + \frac{1}{e_3} \frac{\partial}{\partial k} \left[\frac{K_m}{e_3} \frac{\partial \psi}{\partial k} \right]$$

$$K_m = C_\mu \sqrt{\bar{e}} l$$

$$K_\rho = C_{\mu'} \sqrt{\bar{e}} l$$

$$\epsilon = C_{0\mu} \frac{\bar{e}^{3/2}}{l}$$

where N is the local Brunt-Vaisälä frequency (see [subsection 4.8.2](#)) and ϵ the dissipation rate. The constants $C_1, C_2, C_3, \sigma_e, \sigma_\psi$ and the wall function (Fw) depends of the choice of the turbulence model. Four different turbulent models are pre-defined ([table 9.1](#)). They are made available through the `nn_clo` 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 κz_b (κ is the Von Karman constant and z_b the rugosity length scale) value near physical boundaries (logarithmic boundary layer law). C_μ and $C_{\mu'}$ are calculated from stability function proposed by [Galperin et al. \(1988\)](#), or by [Kantha and Clayson \(1994\)](#) or one of the two functions suggested by [Canuto et al. \(2001\)](#) (`nn_stab_func=0, 3`, resp.). The value of $C_{0\mu}$ depends on 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_bc_surf` and `nn_bc_bot`, resp. As for TKE closure, the wave effect on the mixing is considered when

	$k - kl$	$k - \epsilon$	$k - \omega$	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	-	-	-

Table 9.1.: Set of predefined GLS parameters, or equivalently predefined turbulence models available with `ln_zdfgls=.true.` and controlled by the `nn_clos` namelist variable in `&namzdf_gls` (namelist 9.4).

```

!-----
&namzdf_osm      !   OSM vertical diffusion                               (ln_zdfosm =T)
!-----
ln_use_osm_la = .false.      ! Use namelist rn_osm_la
rn_osm_la     = 0.3          ! Turbulent Langmuir number
rn_osm_dstokes = 5.         ! Depth scale of Stokes drift (m)
nn_ave        = 0           ! choice of horizontal averaging on avt, avmu, avmv
ln_dia_osm    = .true.      ! output OSMOSIS-OBL variables
rn_osm_hbl0   = 10.         ! initial hbl value
ln_kpprimix   = .true.      ! Use KPP-style Ri# mixing below BL
rn_riinfy     = 0.7         ! Highest local Ri_g permitting shear instability
rn_difri      = 0.005       ! max Ri# diffusivity at Ri_g = 0 (m^2/s)
ln_convmix    = .true.      ! Use convective instability mixing below BL
rn_difconv    = 1.          ! diffusivity when unstable below BL (m2/s)
nn_osm_wave   = 0           ! Method used to calculate Stokes drift
!                   ! = 2: Use ECMWF wave fields
!                   ! = 1: Pierson Moskowitz wave spectrum
!                   ! = 0: Constant La# = 0.3
/

```

namelist 9.5.: `&namzdf_osm`

`rn_crban` > 0. (Craig and Banner, 1994; Mellor and Blumberg, 2004). The `rn_crban` namelist parameter is α_{CB} in equation 9.4 and `rn_charn` provides the value of β in equation 9.5.

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 $l_{max} = c_{lim} \sqrt{2\bar{\epsilon}}/N$. A value of $c_{lim} = 0.53$ is often used (Galperin et al., 1988). Umlauf and Burchard (2005) 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 fluxes. 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 subsection 9.1.6 (Burchard, 2002). Evaluation of the 4 GLS turbulent closure schemes can be found in Warner et al. (2005) in ROMS model and in Refrayr. et al. (2015) for the NEMO model.

9.1.5. OSM: OSMOSIS boundary layer scheme (`ln_zdfosm`)

Namelist choices Most of the namelist options refer to how to specify the Stokes surface drift and penetration depth. There are three options:

`nn_osm_wave=0` Default value in `namelist_ref`. In this case the Stokes drift is assumed to be parallel to the surface wind stress, with magnitude consistent with a constant turbulent Langmuir number $La_t = rn_m_la$ i.e. $u_{s0} = \tau / (La_t^2 \rho_0)$. Default value of `rn_m_la` is 0.3. The Stokes penetration depth $\delta = rn_osm_dstokes$; this has default value of 5 m.

`nn_osm_wave=1` In this case the Stokes drift is assumed to be parallel to the surface wind stress, with magnitude as in the classical Pierson-Moskowitz wind-sea spectrum. Significant wave height and wave-mean period taken from this spectrum are used to calculate the Stokes penetration depth, following the approach set out in Breivik et al. (2014).

`nn_osm_wave=2` In this case the Stokes drift is taken from ECMWF wave model output, though only the component parallel to the wind stress is retained. Significant wave height and wave-mean period from ECMWF wave model output are used to calculate the Stokes penetration depth, again following Breivik et al. (2014).

Figure 9.2.: The structure of the entraining boundary layer. (a) Mean buoyancy profile. (b) Profile of the buoyancy flux.

Others refer to the treatment of diffusion and viscosity beneath the surface boundary layer:

`ln_kpprimix` Default is `.true.`. Switches on KPP-style Ri #-dependent mixing below the surface boundary layer.

If this is set `.true.`, the following variable settings are honoured:

`rn_riinfy` Critical value of local Ri # below which shear instability increases vertical mixing from background value.

`rn_difri` Maximum value of Ri #-dependent mixing at $Ri = 0$.

`ln_convmix` If `.true.`, then, where water column is unstable, specify diffusivity equal to `rn_dif_conv` (default value is 1 m s^{-2}).

Diagnostic output is controlled by:

`ln_dia_osm` Default is `.false.`, allows XIOS output of OSMOSIS internal fields.

Obsolete namelist parameters include:

`ln_use_osm_la` `ln_use_osm_la` With `nn_osm_wave=0`, `rn_osm_dstokes` is always used to specify the Stokes penetration depth.

`nn_ave` Choice of averaging method for KPP-style Ri # mixing. Not taken account of.

`rn_osm_hbl0` Depth of initial boundary layer is now set by a density criterion similar to that used in calculating `hmlp` (output as `mldr10_1`) in `zdfmxl.F90`.

Summary

Much of the time the turbulent motions in the ocean surface boundary layer (OSBL) are not given by classical shear turbulence. Instead they are in a regime known as ‘Langmuir turbulence’, dominated by an interaction between the currents and the Stokes drift of the surface waves (e.g. McWilliams et al., 1997). This regime is characterised by strong vertical turbulent motion, and appears when the surface Stokes drift u_{s0} is much greater than the friction velocity u_* . More specifically Langmuir turbulence is thought to be crucial where the turbulent Langmuir number $La_t = (u_*/u_{s0}) > 0.4$.

The OSMOSIS model is fundamentally based on results of Large Eddy Simulations (LES) of Langmuir turbulence and aims to fully describe this Langmuir regime. The description in this section is of necessity incomplete and further details are available in Grant. A (2019); in prep.

The OSMOSIS turbulent closure scheme is a similarity-scale scheme in the same spirit as the K-profile parameterization (KPP) scheme of Large et al. (1994). A specified shape of diffusivity, scaled by the (OSBL) depth h_{BL} and a turbulent velocity scale, is imposed throughout the boundary layer $-h_{BL} < z < \eta$. The turbulent closure model also includes fluxes of tracers and momentum that are “non-local” (independent of the local property gradient).

Rather than the OSBL depth being diagnosed in terms of a bulk Richardson number criterion, as in KPP, it is set by a prognostic equation that is informed by energy budget considerations reminiscent of the classical mixed layer models of Kraus and Turner (1967). The model also includes an explicit parametrization of the structure of the pycnocline (the stratified region at the bottom of the OSBL).

Presently, mixing below the OSBL is handled by the Richardson number-dependent mixing scheme used in Large et al. (1994).

Convective parameterizations such as described in 9.2 below should not be used with the OSMOSIS-OBL model: instabilities within the OSBL are part of the model, while instabilities below the ML are handled by the Ri # dependent scheme.

Depth and velocity scales

The model supposes a boundary layer of thickness h_{bl} enclosing a well-mixed layer of thickness h_{ml} and a relatively thin pycnocline at the base of thickness Δh ; Fig. 9.2 shows typical (a) buoyancy structure and (b) turbulent buoyancy flux profile for the unstable boundary layer (losing buoyancy at the surface; e.g. cooling). The pycnocline in the OSMOSIS scheme is assumed to have a finite thickness, and may include a number of model levels. This means that the OSMOSIS scheme must parametrize both the thickness of the pycnocline, and the turbulent fluxes within the pycnocline.

Consideration of the power input by wind acting on the Stokes drift suggests that the Langmuir turbulence has velocity scale:

$$w_{*L} = (u_*^2 u_{s0})^{1/3}; \quad (9.8)$$

but at times the Stokes drift may be weak due to e.g. ice cover, short fetch, misalignment with the surface stress, etc. so a composite velocity scale is assumed for the stable (warming) boundary layer:

$$\nu_* = \{u_*^3 [1 - \exp(-.5La_t^2)] + w_{*L}^3\}^{1/3}. \quad (9.9)$$

For the unstable boundary layer this is merged with the standard convective velocity scale $w_{*C} = (\overline{w'b'}_0 h_{ml})^{1/3}$, where $\overline{w'b'}_0$ is the upwards surface buoyancy flux, to give:

$$\omega_* = (\nu_*^3 + 0.5w_{*C}^3)^{1/3}. \quad (9.10)$$

The flux gradient model

The flux-gradient relationships used in the OSMOSIS scheme take the form:

$$\overline{w'\chi'} = -K \frac{\partial \overline{\chi}}{\partial z} + N_{\chi,s} + N_{\chi,b} + N_{\chi,t}, \quad (9.11)$$

where χ is a general variable and $N_{\chi,s}$, $N_{\chi,b}$ and $N_{\chi,t}$ are the non-gradient terms, and represent the effects of the different terms in the turbulent flux-budget on the transport of χ . $N_{\chi,s}$ represents the effects that the Stokes shear has on the transport of χ , $N_{\chi,b}$ the effect of buoyancy, and $N_{\chi,t}$ the effect of the turbulent transport. The same general form for the flux-gradient relationship is used to parametrize the transports of momentum, heat and salinity.

In terms of the non-dimensionalized depth variables

$$\sigma_{ml} = -z/h_{ml}; \quad \sigma_{bl} = -z/h_{bl}, \quad (9.12)$$

in unstable conditions the eddy diffusivity (K_d) and eddy viscosity (K_ν) profiles are parametrized as:

$$K_d = 0.8 \omega_* h_{ml} \sigma_{ml} (1 - \beta_d \sigma_{ml})^{3/2} \quad (9.13)$$

$$K_\nu = 0.3 \omega_* h_{ml} \sigma_{ml} (1 - \beta_\nu \sigma_{ml}) (1 - \frac{1}{2} \sigma_{ml}^2) \quad (9.14)$$

where β_d and β_ν are parameters that are determined by matching Eqs 9.13 and 9.14 to the eddy diffusivity and viscosity at the base of the well-mixed layer, given by

$$K_{d,ml} = K_{\nu,ml} = 0.16 \omega_* \Delta h. \quad (9.15)$$

For stable conditions the eddy diffusivity/viscosity profiles are given by:

$$K_d = 0.75 \nu_* h_{ml} \exp[-2.8 (h_{bl}/L_L)^2] \sigma_{ml} (1 - \sigma_{ml})^{3/2} \quad (9.16)$$

$$K_\nu = 0.375 \nu_* h_{ml} \exp[-2.8 (h_{bl}/L_L)^2] \sigma_{ml} (1 - \sigma_{ml}) (1 - \frac{1}{2} \sigma_{ml}^2). \quad (9.17)$$

The shape of the eddy viscosity and diffusivity profiles is the same as the shape in the unstable OSBL. The eddy diffusivity/viscosity depends on the stability parameter h_{bl}/L_L where L_L is analogous to the Obukhov length, but for Langmuir turbulence:

$$L_L = -w_{*L}^3 / \langle \overline{w'b'} \rangle_L, \quad (9.18)$$

with the mean turbulent buoyancy flux averaged over the boundary layer given in terms of its surface value $\overline{w'b'}_0$ and (downwards) solar irradiance $I(z)$ by

$$\langle \overline{w'b'} \rangle_L = \frac{1}{2} \overline{w'b'}_0 - g\alpha_E [\frac{1}{2}(I(0) + I(-h)) - \langle I \rangle]. \quad (9.19)$$

In unstable conditions the eddy diffusivity and viscosity depend on stability through the velocity scale ω_* , which depends on the two velocity scales ν_* and w_{*C} .

Details of the non-gradient terms in (9.11) and of the fluxes within the pycnocline $-h_{bl} < z < h_{ml}$ can be found in Grant (2019).

Evolution of the boundary layer depth

The prognostic equation for the depth of the neutral/unstable boundary layer is given by (?),

$$\frac{\partial h_{bl}}{\partial t} = W_b - \frac{\overline{w'b'}_{ent}}{\Delta B_{bl}} \quad (9.20)$$

where h_{bl} is the horizontally-varying depth of the OSBL, U_b and W_b are the mean horizontal and vertical velocities at the base of the OSBL, $\overline{w'b'}_{ent}$ is the buoyancy flux due to entrainment and ΔB_{bl} is the difference between the buoyancy

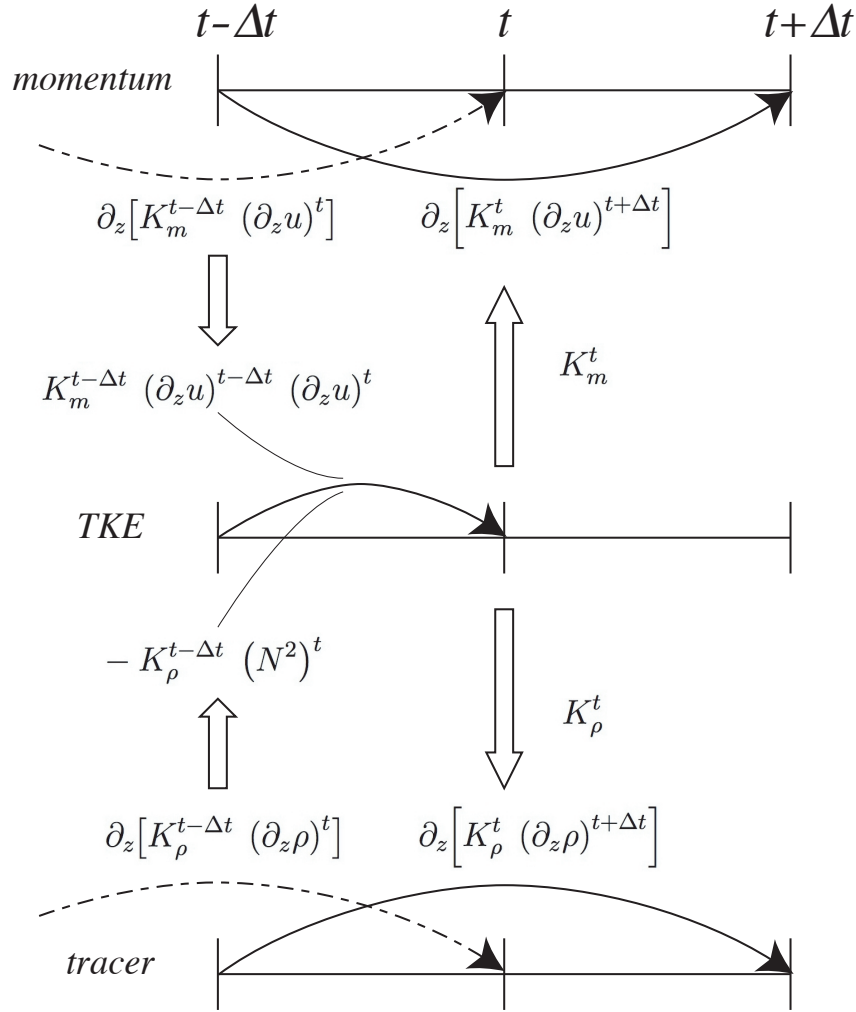


Figure 9.3.: Illustration of the subgrid kinetic energy integration in GLS and TKE schemes and its links to the momentum and tracer time integration.

averaged over the depth of the OSBL (i.e. including the ML and pycnocline) and the buoyancy just below the base of the OSBL. This equation for the case when the pycnocline has a finite thickness, based on the potential energy budget of the OSBL, is the leading term (?) of a generalization of that used in mixed-layer models e.g. Kraus and Turner (1967), in which the thickness of the pycnocline is taken to be zero.

The entrainment flux for the combination of convective and Langmuir turbulence is given by

$$\overline{w'b'}_{\text{ent}} = -\alpha_B \overline{w'b'}_0 - \alpha_S \frac{u_*^3}{h_{\text{ml}}} + G(\delta/h_{\text{ml}}) \left[\alpha_S e^{-1.5 \text{La}_t} - \alpha_L \frac{w_{*L}^3}{h_{\text{ml}}} \right] \quad (9.21)$$

where the factor $G \equiv 1 - e^{-25\delta/h_{\text{bl}}} (1 - 4\delta/h_{\text{bl}})$ models the lesser efficiency of Langmuir mixing when the boundary-layer depth is much greater than the Stokes depth, and α_B , α_S and α_L depend on the ratio of the appropriate eddy turnover time to the inertial timescale f^{-1} . Results from the LES suggest $\alpha_B = 0.18F(fh_{\text{bl}}/w_{*C})$, $\alpha_S = 0.15F(fh_{\text{bl}}/u_*)$ and $\alpha_L = 0.035F(fh_{\text{bl}}/w_{*L})$, where $F(x) \equiv \tanh(x^{-1})^{0.69}$.

For the stable boundary layer, the equation for the depth of the OSBL is:

$$\max \left(\Delta B_{\text{bl}}, \frac{w_{*L}^2}{h_{\text{bl}}} \right) \frac{\partial h_{\text{bl}}}{\partial t} = \left(0.06 + 0.52 \frac{h_{\text{bl}}}{L_L} \right) \frac{w_{*L}^3}{h_{\text{bl}}} + \langle \overline{w'b'} \rangle_L. \quad (9.22)$$

Equation. 9.20 always leads to the depth of the entraining OSBL increasing (ignoring the effect of the mean vertical motion), but the change in the thickness of the stable OSBL given by Eq. 9.22 can be positive or negative, depending on the magnitudes of $\langle \overline{w'b'} \rangle_L$ and h_{bl}/L_L . The rate at which the depth of the OSBL can decrease is limited by choosing an effective buoyancy w_{*L}^2/h_{bl} , in place of ΔB_{bl} which will be ≈ 0 for the collapsing OSBL.

9.1.6. Discrete energy conservation for TKE and GLS schemes

The production of turbulence by vertical shear (the first term of the right hand side of equation 9.1) and equation 9.7) should balance the loss of kinetic energy associated with the vertical momentum diffusion (first line in equation 1.17). To

do so a special care has to be taken for both the time and space discretization of the kinetic energy equation (Burchard, 2002; Marsaleix et al., 2008).

Let us first address the time stepping issue. [figure 9.3](#) shows how the two-level Leap-Frog time stepping of the momentum and tracer equations interplays with the one-level forward time stepping of the equation for \bar{e} . 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:

$$\begin{aligned} \int_{-H}^{\eta} u^t \partial_z (K_m^t (\partial_z u)^{t+\Delta t}) dz \\ = \left[u^t K_m^t (\partial_z u)^{t+\Delta t} \right]_{-H}^{\eta} - \int_{-H}^{\eta} K_m^t \partial_z u^t \partial_z u^{t+\Delta t} dz \end{aligned} \quad (9.23)$$

Here, the vertical diffusion of momentum is discretized backward in time with a coefficient, K_m , known at time t ([figure 9.3](#)), as it is required when using the TKE scheme (see [section 2.3](#)). The first term of the right hand side of [equation 9.23](#) 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} . [equation 9.23](#) 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 [equation 9.1](#) and [equation 9.7](#)). 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 to vertical mixing is obtained by multiplying the vertical density diffusion tendency by $g z$ and and summing the result vertically:

$$\begin{aligned} \int_{-H}^{\eta} g z \partial_z (K_\rho^t (\partial_k \rho)^{t+\Delta t}) dz \\ = \left[g z K_\rho^t (\partial_k \rho)^{t+\Delta t} \right]_{-H}^{\eta} - \int_{-H}^{\eta} g K_\rho^t (\partial_k \rho)^{t+\Delta t} dz \\ = - \left[z K_\rho^t (N^2)^{t+\Delta t} \right]_{-H}^{\eta} + \int_{-H}^{\eta} \rho^{t+\Delta t} K_\rho^t (N^2)^{t+\Delta t} dz \end{aligned} \quad (9.24)$$

where we use $N^2 = -g \partial_k \rho / (e_3 \rho)$. The first term of the right hand side of [equation 9.24](#) is always zero because there is no diffusive flux through the ocean surface and bottom). The second term is minus the destruction rate of \bar{e} due to stratification. Therefore [equation 9.23](#) implies that, to be energetically consistent, the product $K_\rho^{t-\Delta t} (N^2)^t$ should be used in [equation 9.1](#) and [equation 9.7](#).

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 ([figure 3.1](#)). A space averaging is thus required to obtain the shear TKE production term. By redoing the [equation 9.23](#) 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 time variation of e_3 has be taken into account.

The above energetic considerations leads to the following final discrete form for the TKE equation:

$$\begin{aligned} \frac{(\bar{e})^t - (\bar{e})^{t-\Delta t}}{\Delta t} \equiv & \left\{ \left(\left(\overline{K_m^{i+1/2}} \right)^{t-\Delta t} \frac{\delta_{k+1/2}[u^{t+\Delta t}]}{e_3 u^{t+\Delta t}} \frac{\delta_{k+1/2}[u^t]}{e_3 u^t} \right)^i \right. \\ & \left. + \left(\left(\overline{K_m^{j+1/2}} \right)^{t-\Delta t} \frac{\delta_{k+1/2}[v^{t+\Delta t}]}{e_3 v^{t+\Delta t}} \frac{\delta_{k+1/2}[v^t]}{e_3 v^t} \right)^j \right\} \\ & - K_\rho^{t-\Delta t} (N^2)^t \\ & + \frac{1}{e_3 w^{t+\Delta t}} \delta_{k+1/2} \left[K_m^{t-\Delta t} \frac{\delta_k[(\bar{e})^{t+\Delta t}]}{e_3 w^{t+\Delta t}} \right] \\ & - c_\epsilon \left(\frac{\sqrt{\bar{e}}}{l_\epsilon} \right)^{t-\Delta t} (\bar{e})^{t+\Delta t} \end{aligned} \quad (9.25)$$

where the last two terms in [equation 9.25](#) (vertical diffusion and Kolmogorov dissipation) are time stepped using a backward scheme (see [section 2.3](#)). Note that the Kolmogorov term has been linearized in time in order to render the implicit computation possible.

9.2. Convection

Static instabilities (*i.e.* light potential densities under heavy ones) may occur at particular ocean grid points. In nature, convective processes quickly re-establish the static stability of the water column. These processes have been removed

from the model via the hydrostatic assumption so they must be parameterized. Three parameterisations are available to deal with convective processes: a non-penetrative convective adjustment or an enhanced vertical diffusion, or/and the use of a turbulent closure scheme.

9.2.1. Non-penetrative convective adjustment (`ln_tranpc`)

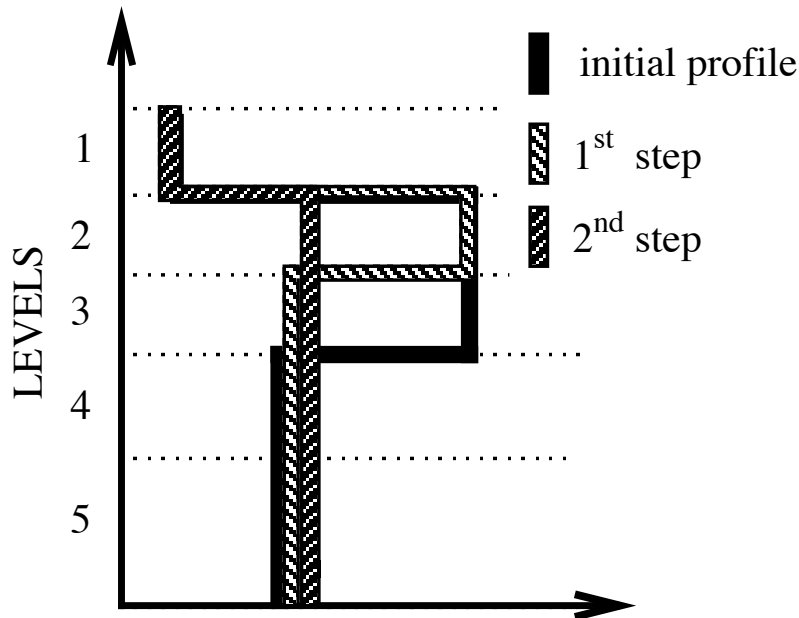


Figure 9.4.: Example of an unstable density profile treated by the non penetrative convective adjustment algorithm. 1^{st} step: the initial profile is checked from the surface to the bottom. It is found to be unstable between levels 3 and 4. They are mixed. The resulting ρ is still larger than $\rho(5)$: 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.

Options are defined through the `&namzdf` (namelist 9.1) namelist variables. 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) (Madec et al., 1991b). The associated algorithm is an iterative process used in the following way (figure 9.4): 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 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 Killworth (1989): it avoids the existence of permanent and unrealistic static instabilities at the sea surface. This non-penetrative convective algorithm has been proved successful in studies of the deep water formation in the north-western Mediterranean Sea (Madec et al., 1991b,a; Madec and Crépon, 1991).

The current implementation has been modified in order to deal with any non linear equation of seawater (L. Brodeau, personal communication). Two main differences have been introduced compared to the original algorithm: (i) the stability is now checked using the Brunt-Väisälä frequency (not the difference in potential density); (ii) when two levels are found unstable, their thermal and haline expansion coefficients are vertically mixed in the same way their temperature and salinity has been mixed. These two modifications allow the algorithm to perform properly and accurately with TEOS10 or EOS-80 without having to recompute the expansion coefficients at each mixing iteration.

9.2.2. Enhanced vertical diffusion (`ln_zdfevd`)

Options are defined through the `&namzdf` (namelist 9.1) namelist variables. The enhanced vertical diffusion parameterisation is used when `ln_zdfevd=.true.`. In this case, the vertical eddy mixing coefficients are assigned very large

values in regions where the stratification is unstable (*i.e.* when N^2 the Brunt-Vaisälä frequency is negative) (Lazar, 1997; Lazar et al., 1999). This is done either on tracers only (`nn_evdm=0`) or on both momentum and tracers (`ln_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 $100 \text{ m}^2 \cdot \text{s}^{-1}$. 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.

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 (Leclair, 2010) (see section 2.5).

9.2.3. Handling convection with turbulent closure schemes

(`ln_zdf{tke, gls, osm}`)

The turbulent closure schemes presented in subsection 9.1.3, subsection 9.1.4 and subsection 9.1.5 (*i.e.* `ln_zdf_tke` or `ln_zdf_gls` or `ln_zdf_osm` defined) deal, in theory, with statically unstable density profiles. In such a case, the term corresponding to the destruction of turbulent kinetic energy through stratification in equation 9.1 or equation 9.7 becomes a source term, since N^2 is negative. It results in large values of A_T^{vT} and A_T^{vS} , and also of the four neighboring values at velocity points A_u^{vm} and A_v^{vm} (up to $1 \text{ m}^2 \cdot \text{s}^{-1}$). These large values restore the static stability of the water column in a way similar to that of the enhanced vertical diffusion parameterisation (subsection 9.2.2). However, in the vicinity of the sea surface (first ocean layer), the eddy coefficients computed by the turbulent closure scheme do not usually exceed $10^{-2} \text{ m}^2 \cdot \text{s}^{-1}$, 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_zdf_npc` namelist parameter to true and defining the turbulent closure (`ln_zdf_tke` or `ln_zdf_gls = .true.`) all together.

The OSMOSIS turbulent closure scheme already includes enhanced vertical diffusion in the case of convection, therefore `ln_zdf_evdm=.false.` should be used with the OSMOSIS scheme.

9.3. Double diffusion mixing (`ln_zdfddm`)

This parameterisation has been introduced in `zdfddm.F90` module and is controlled by the namelist parameter `ln_zdfddm` in `&namzdf` (namelist 9.1). 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. Merryfield et al. (1999) include a parameterisation of such phenomena in a global ocean model and show that it leads to relatively minor changes in circulation but exerts significant regional influences on temperature and salinity.

Diapycnal mixing of S and T are described by diapycnal diffusion coefficients

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

where subscript *f* represents mixing by salt fingering, *d* by diffusive convection, and *o* by processes other than double diffusion. The rates of double-diffusive mixing depend on 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 subsection 4.8.1). To represent mixing of *S* and *T* by salt fingering, we adopt the diapycnal diffusivities suggested by Schmitt (1981):

$$A_f^{vS} = \begin{cases} \frac{A^{*v}}{1+(R_\rho/R_c)^n} & \text{if } R_\rho > 1 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \quad (9.26)$$

$$A_f^{vT} = 0.7 A_f^{vS} / R_\rho \quad (9.27)$$

The factor 0.7 in equation 9.27 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.*, McDougall and Taylor (1984)). Following Merryfield et al. (1999), we adopt $R_c = 1.6$, $n = 6$, and $A^{*v} = 10^{-4} \text{ m}^2 \cdot \text{s}^{-1}$.

To represent mixing of S and T by diffusive layering, the diapycnal diffusivities suggested by Federov (1988) is used:

$$\begin{aligned} A_d^{vT} &= \begin{cases} 1.3635 \exp(4.6 \exp[-0.54(R_\rho^{-1} - 1)]) & \text{if } 0 < R_\rho < 1 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \\ A_d^{vS} &= \begin{cases} A_d^{vT} (1.85 R_\rho - 0.85) & \text{if } 0.5 \leq R_\rho < 1 \text{ and } N^2 > 0 \\ A_d^{vT} 0.15 R_\rho & \text{if } 0 < R_\rho < 0.5 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (9.28)$$

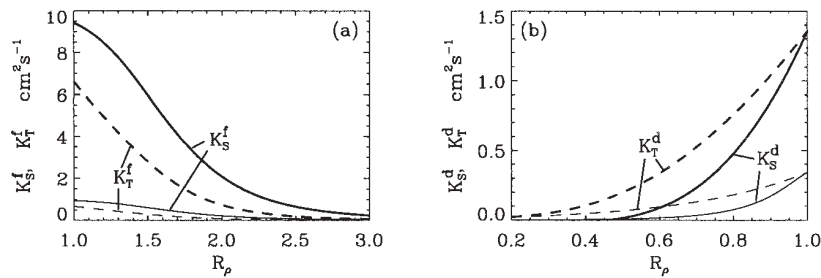


Figure 9.5.: From Merryfield et al. (1999): (a) Diapycnal diffusivities A_f^{vT} and A_f^{vS} for temperature and salt in regions of salt fingering. Heavy curves denote $A^{*v} = 10^{-3} \text{ m}^2.\text{s}^{-1}$ and thin curves $A^{*v} = 10^{-4} \text{ m}^2.\text{s}^{-1}$; (b) diapycnal diffusivities 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*.

```

!-----
&namdrg      !   top/bottom drag coefficient                               (default: NO selection)
!-----
ln_OFF       = .false.  !   free-slip      : Cd = 0                               (F => fill namdrg_bot
ln_lin       = .false.  !   linear drag: Cd = Cd0 Uc0                               & namdrg_top)
ln_non_lin   = .false.  !   non-linear drag: Cd = Cd0 |U|
ln_loglayer  = .false.  !   logarithmic drag: Cd = vkarmn/log(z/z0) |U|
!
ln_drgimp    = .true.   !   implicit top/bottom friction flag
/
    
```

namelist 9.6.: `&namdrg`

The dependencies of equation 9.26 to equation 9.28 on R_ρ are illustrated in figure 9.5. Implementing this requires computing R_ρ at each grid point on every time step. This is done in *eosbn2.F90* at the same time as N^2 is computed. This avoids duplication in the computation of α and β (which is usually quite expensive).

9.4. Bottom and top friction (*zdfdrg.F90*)

Options to define the top and bottom friction are defined through the `&namdrg` (namelist 9.6) namelist variables. The bottom friction represents the friction generated by the bathymetry. The top friction represents the friction generated by the ice shelf/ocean interface. As the friction processes at the top and the bottom are treated in an identical way, the description below considers mostly the bottom friction case, if not stated otherwise.

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:

$$A^{vm} (\partial U_h / \partial z) = \mathcal{F}_h^U$$

where \mathcal{F}_h^U 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^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 $d = \sqrt{2 A^{vm} / f} = 50 \text{ m}$, one needs a vertical diffusion coefficient $A^{vm} = 0.125 \text{ m}^2.\text{s}^{-1}$ (for a Coriolis frequency

```

!-----
&namdrg_top  !   TOP friction                                           (ln_OFF =F & ln_isfcav=T)
!-----
rn_Cd0       = 1.e-3  !   drag coefficient [-]
rn_Uc0       = 0.4    !   ref. velocity [m/s] (linear drag=Cd0*Uc0)
rn_Cdmax     = 0.1    !   drag value maximum [-] (logarithmic drag)
rn_ke0       = 2.5e-3 !   background kinetic energy [m2/s2] (non-linear cases)
rn_z0        = 3.0e-3 !   roughness [m] (ln_loglayer=T)
ln_boost     = .false. !   =T regional boost of Cd0 ; =F constant
rn_boost     = 50.    !   local boost factor [-]
/
    
```

namelist 9.7.: `&namdrg_top`

```

!-----
&namdrg_bot    !   BOTTOM friction                                (ln_OFF =F)
!-----
rn_Cd0         = 1.e-3    !   drag coefficient [-]
rn_Uc0         = 0.4      !   ref. velocity [m/s] (linear drag=Cd0*Uc0)
rn_Cdmax       = 0.1      !   drag value maximum [-] (logarithmic drag)
rn_ke0         = 2.5e-3   !   background kinetic energy [m2/s2] (non-linear cases)
rn_z0          = 3.e-3    !   roughness [m] (ln_loglayer=T)
ln_boost       = .false.  !   =T regional boost of Cd0 ; =F constant
                rn_boost = 50.    !   local boost factor [-]
/

```

namelist 9.8.: &namdrg_bot

$f = 10^{-4} \text{ m}^2\text{s}^{-1}$). With a background diffusion coefficient $A^{vm} = 10^{-4} \text{ m}^2\text{s}^{-1}$, 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:

$$\frac{\partial u_k}{\partial t} = \frac{1}{e_{3u}} \left[\frac{A^{vm}}{e_{3uw}} \delta_{k+1/2} [u] - \mathcal{F}_h^u \right] \approx -\frac{\mathcal{F}_h^u}{e_{3u}} \quad (9.29)$$

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}_h^u . 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 *dynzdf.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 *zdfdrg.F90* and generally take the form c_b^U where:

$$\frac{\partial \mathbf{U}_h}{\partial t} = -\frac{\mathcal{F}_h^U}{e_{3u}} = \frac{c_b^U}{e_{3u}} \mathbf{U}_h^b \quad (9.30)$$

where $\mathbf{U}_h^b = (u_b, v_b)$ is the near-bottom, horizontal, ocean velocity. Note that from *NEMO* 4.0, drag coefficients are only computed at cell centers (*i.e.* at T-points) and refer to as c_b^T in the following. These are then linearly interpolated in space to get c_b^U at velocity points.

9.4.1. Linear top/bottom friction (ln_lin)

The linear friction parameterisation (including the special case of a free-slip condition) assumes that the friction is proportional to the interior velocity (*i.e.* the velocity of the first/last model level):

$$\mathcal{F}_h^U = \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} = r \mathbf{U}_h^b$$

where r is a friction coefficient expressed in m s^{-1} . This coefficient is generally estimated by setting a typical decay time τ in the deep ocean, and setting $r = H/\tau$, where H is the ocean depth. Commonly accepted values of τ are of the order of 100 to 200 days (Weatherly, 1984). A value $\tau^{-1} = 10^{-7} \text{ s}^{-1}$ equivalent to 115 days, is usually used in quasi-geostrophic models. One may consider the linear friction as an approximation of quadratic friction, $r \approx 2 C_D U_{av}$ (Gill (1982), Eq. 9.6.6). For example, with a drag coefficient $C_D = 0.002$, a typical speed of tidal currents of $U_{av} = 0.1 \text{ m s}^{-1}$, and assuming an ocean depth $H = 4000 \text{ m}$, the resulting friction coefficient is $r = 4 \cdot 10^{-4} \text{ m s}^{-1}$. 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_Uc0` (namelist parameter).

For the linear friction case the drag coefficient used in the general expression equation 9.30 is:

$$c_b^T = -r$$

When `ln_lin=.true.`, the value of r used is `rn_Uc0 * rn_Cd0`. Setting `ln_OFF=.true.` (and `ln_lin=.true.`) is equivalent to setting $r = 0$ and leads to a free-slip boundary condition.

These values are assigned in `zdfdrg.F90`. Note that there is support for local enhancement of these values via an externally defined 2D mask array (`ln_boost=.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_boost * rn_Cd0`.

9.4.2. Non-linear top/bottom friction (`ln_non_lin`)

The non-linear bottom friction parameterisation assumes that the top/bottom friction is quadratic:

$$\mathcal{F}_h^{\mathbf{U}} = \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} = C_D \sqrt{u_b^2 + v_b^2 + e_b} \mathbf{U}_h^b$$

where C_D is a drag coefficient, and e_b a top/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 (Tréguier, 1992) uses $C_D = 10^{-3}$ and $e_b = 2.5 \cdot 10^{-3} \text{m}^2 \text{s}^{-2}$, while the FRAM experiment (Killworth, 1992) uses $C_D = 1.4 \cdot 10^{-3}$ and $e_b = 2.5 \cdot 10^{-3} \text{m}^2 \text{s}^{-2}$. The CME choices have been set as default values (`rn_Cd0` and `rn_ke0` namelist parameters).

As for the linear case, the friction is imposed in the code by adding the trend due to the friction to the general momentum trend in `dynzdf.F90`. For the non-linear friction case the term computed in `zdfdrg.F90` is:

$$c_b^T = -C_D \left[(\bar{u}_b^i)^2 + (\bar{v}_b^j)^2 + e_b \right]^{1/2}$$

The coefficients that control the strength of the non-linear friction are initialised as namelist parameters: $C_D = \text{rn_Cd0}$, and $e_b = \text{rn_bfeb2}$. Note that for applications which consider tides explicitly, a low or even zero value of `rn_bfeb2` is recommended. A local enhancement of C_D is again possible via an externally defined 2D mask array (`ln_boost=.true.`). This works in the same way as for the linear friction case with non-zero masked locations increased by `mask_value * rn_boost * rn_Cd0`.

9.4.3. Log-layer top/bottom friction (`ln_loglayer`)

In the non-linear friction case, the drag coefficient, C_D , can be optionally enhanced using a "law of the wall" scaling. This assumes that the model vertical resolution can capture the logarithmic layer which typically occur for layers thinner than 1 m or so. If `ln_loglayer=.true.`, C_D is no longer constant but is related to the distance to the wall (or equivalently to the half of the top/bottom layer thickness):

$$C_D = \left(\frac{\kappa}{\log(0.5 e_{3b}/rn_z0)} \right)^2$$

where κ is the von-Karman constant and `rn_z0` is a roughness length provided via the namelist.

The drag coefficient is bounded such that it is kept greater or equal to the base `rn_Cd0` value which occurs where layer thicknesses become large and presumably logarithmic layers are not resolved at all. For stability reason, it is also not allowed to exceed the value of an additional namelist parameter: `rn_Cdmax`, *i.e.*

$$rn_Cd0 \leq C_D \leq rn_Cdmax$$

The log-layer enhancement can also be applied to the top boundary friction if under ice-shelf cavities are activated (`ln_isfcav=.true.`).

9.4.4. Explicit top/bottom friction (`ln_drgimp=.false.`)

Setting `ln_drgimp=.false.` means that bottom friction is treated explicitly in time, which has the advantage of simplifying the interaction with the split-explicit free surface (see subsection 9.4.6). The latter does indeed require the knowledge of bottom stresses in the course of the barotropic sub-iteration, which becomes less straightforward in the implicit case. In the explicit case, top/bottom stresses can be computed using *before* velocities and inserted in the overall momentum tendency budget. This reads:

At the top (below an ice shelf cavity):

$$\left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_t = c_t^{\mathbf{U}} \mathbf{u}_t^{n-1}$$

At the bottom (above the sea floor):

$$\left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_b = c_b^{\mathbf{U}} \mathbf{u}_b^{n-1}$$

Since this is conditionally stable, some care needs to be exercised over the choice of parameters to ensure that the implementation of explicit top/bottom friction does not induce numerical instability. For the purposes of stability analysis, an approximation to [equation 9.29](#) is:

$$\begin{aligned}\Delta u &= -\frac{\mathcal{F}_h^u}{e_{3u}} 2\Delta t \\ &= -\frac{ru}{e_{3u}} 2\Delta t\end{aligned}\tag{9.31}$$

where linear friction and a leapfrog timestep have been assumed. To ensure that the friction cannot reverse the direction of flow it is necessary to have:

$$|\Delta u| < |u|$$

which, using [equation 9.31](#), gives:

$$r \frac{2\Delta t}{e_{3u}} < 1 \quad \Rightarrow \quad r < \frac{e_{3u}}{2\Delta t}$$

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:

$$e_{3u} > 2 r \Delta t$$

which it may be necessary to impose if partial steps are being used. For example, if $|u| = 1 \text{ m.s}^{-1}$, $r\Delta t = 1800 \text{ s}$, $r = 10^{-3}$ 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 top/bottom friction coefficients both during initialisation and at each time step. Checks at initialisation are made in *zdfdrg.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 *dynzdf.F90*. Values of the friction coefficient are reduced as necessary to ensure stability; these changes are not reported.

Limits on the top/bottom friction coefficient are not imposed if the user has elected to handle the friction implicitly (see [subsection 9.4.5](#)). The number of potential breaches of the explicit stability criterion are still reported for information purposes.

9.4.5. Implicit top/bottom friction (`ln_drgimp=.true.`)

An optional implicit form of bottom friction has been implemented to improve model stability. We recommend this option for shelf sea and coastal ocean applications. This option can be invoked by setting `ln_drgimp` to `.true.` in the `&namdrg` ([namelist 9.6](#)) namelist.

This implementation is performed in *dynzdf.F90* where the following boundary conditions are set while solving the fully implicit diffusion step:

At the top (below an ice shelf cavity):

$$\left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_t = c_t^U \mathbf{u}_t^{n+1}$$

At the bottom (above the sea floor):

$$\left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_b = c_b^U \mathbf{u}_b^{n+1}$$

where t and b refers to top and bottom layers respectively. Superscript $n + 1$ means the velocity used in the friction formula is to be calculated, so it is implicit.

9.4.6. Bottom friction with split-explicit free surface

With split-explicit free surface, the sub-stepping of barotropic equations needs the knowledge of top/bottom stresses. An obvious way to satisfy this is to take them as constant over the course of the barotropic integration and equal to the value used to update the baroclinic momentum trend. Provided `ln_drgimp=.false.` and a centred or *leap-frog* like integration of barotropic equations is used (*i.e.* `ln_bt_fw=.false.`, cf [subsection 5.5.2](#)), this does ensure that barotropic and baroclinic dynamics feel the same stresses during one leapfrog time step. However, if `ln_drgimp=.true.`, stresses depend on the *after* value of the velocities which themselves depend on the barotropic iteration result. This cyclic dependency makes difficult obtaining consistent stresses in 2d and 3d dynamics. Part of this mismatch is then removed when setting the final barotropic component of 3d velocities to the time splitting estimate. This last step can be seen as a necessary evil but should be minimized since it interferes with the adjustment to the boundary conditions.

The strategy to handle top/bottom stresses with split-explicit free surface in *NEMO* is as follows:

```

!-----
&namzdf_iwm      !      internal wave-driven mixing parameterization      (ln_zdfiwm =T)
!-----
nn_zpyc         = 1          !      pycnocline-intensified dissipation scales as N (=1) or N^2 (=2)
ln_mevar        = .true.    !      variable (T) or constant (F) mixing efficiency
ln_tsdiff       = .true.    !      account for differential T/S mixing (T) or not (F)
/
    
```

namelist 9.9.: &namzdf_iwm

1. To extend the stability of the barotropic sub-stepping, bottom stresses are refreshed at each sub-iteration. The baroclinic part of the flow entering the stresses is frozen at the initial time of the barotropic iteration. In case of non-linear friction, the drag coefficient is also constant.
2. In case of an implicit drag, specific computations are performed in `dynzdf.F90` which renders the overall scheme mixed explicit/implicit: the barotropic components of 3d velocities are removed before seeking for the implicit vertical diffusion result. Top/bottom stresses due to the barotropic components are explicitly accounted for thanks to the updated values of barotropic velocities. Then the implicit solution of 3d velocities is obtained. Lastly, the residual barotropic component is replaced by the time split estimate.

Note that other strategies are possible, like considering vertical diffusion step in advance, *i.e.* prior barotropic integration.

9.5. Internal wave-driven mixing (`ln_zdfiwm`)

The parameterization of mixing induced by breaking internal waves is a generalization of the approach originally proposed by [St Laurent et al. \(2002\)](#). A three-dimensional field of internal wave energy dissipation $\epsilon(x, y, z)$ is first constructed, and the resulting diffusivity is obtained as

$$A_{wave}^{vT} = R_f \frac{\epsilon}{\rho N^2}$$

where R_f is the mixing efficiency and ϵ is a specified three dimensional distribution of the energy available for mixing. If the `ln_mevar` namelist parameter is set to `.false.`, the mixing efficiency is taken as constant and equal to 1/6 ([Osborn, 1980](#)). In the opposite (recommended) case, R_f is instead a function of the turbulence intensity parameter $Re_b = \frac{\epsilon}{\nu N^2}$, with ν the molecular viscosity of seawater, following the model of [Bouffard and Boegman \(2013\)](#) and the implementation of [de Lavergne et al. \(2016\)](#). Note that A_{wave}^{vT} is bounded by $10^{-2} m^2/s$, a limit that is often reached when the mixing efficiency is constant.

In addition to the mixing efficiency, the ratio of salt to heat diffusivities can chosen to vary as a function of Re_b by setting the `ln_tsdiff` parameter to `.true.`, a recommended choice. This parameterization of differential mixing, due to [Jackson and Rehmann \(2014\)](#), is implemented as in [de Lavergne et al. \(2016\)](#).

The three-dimensional distribution of the energy available for mixing, $\epsilon(i, j, k)$, is constructed from three static maps of column-integrated internal wave energy dissipation, $E_{cri}(i, j)$, $E_{pyc}(i, j)$, and $E_{bot}(i, j)$, combined to three corresponding vertical structures:

$$\begin{aligned} F_{cri}(i, j, k) &\propto e^{-h_{ab}/h_{cri}} \\ F_{pyc}(i, j, k) &\propto N^{n_p} \\ F_{bot}(i, j, k) &\propto N^2 e^{-h_{wkb}/h_{bot}} \end{aligned}$$

In the above formula, h_{ab} denotes the height above bottom, h_{wkb} denotes the WKB-stretched height above bottom, defined by

$$h_{wkb} = H \frac{\int_{-H}^z N dz'}{\int_{-H}^{\eta} N dz'}$$

The n_p parameter (given by `nn_zpyc` in `&namzdf_iwm` (namelist 9.9) namelist) controls the stratification-dependence of the pycnocline-intensified dissipation. It can take values of 1 (recommended) or 2. Finally, the vertical structures F_{cri} and F_{bot} require the specification of the decay scales $h_{cri}(i, j)$ and $h_{bot}(i, j)$, which are defined by two additional input maps. h_{cri} is related to the large-scale topography of the ocean (etopo2) and h_{bot} is a function of the energy flux E_{bot} , the characteristic horizontal scale of the abyssal hill topography ([Goff, 2010](#)) and the latitude.

9.6. Surface wave-induced mixing (`ln_zdfswm`)

Surface waves produce an enhanced mixing through wave-turbulence interaction. In addition to breaking waves induced turbulence (subsection 9.1.3), the influence of non-breaking waves can be accounted introducing wave-induced viscosity and diffusivity as a function of the wave number spectrum. Following Qiao et al. (2010), a formulation of wave-induced mixing coefficient is provided as a function of wave amplitude, Stokes Drift and wave-number:

$$B_v = \alpha AU_{st} \exp(3kz) \quad (9.32)$$

Where B_v is the wave-induced mixing coefficient, A is the wave amplitude, U_{st} is the Stokes Drift velocity, k is the wave number and α is a constant which should be determined by observations or numerical experiments and is set to be 1.

The coefficient B_v is then directly added to the vertical viscosity and diffusivity coefficients.

In order to account for this contribution set: `ln_zdfswm=.true.`, then wave interaction has to be activated through `ln_wave=.true.`, the Stokes Drift can be evaluated by setting `ln_sdw=.true.` (see subsection 6.12.2) and the needed wave fields can be provided either in forcing or coupled mode (for more information on wave parameters and settings see section 6.12)

9.7. Adaptive-implicit vertical advection(`ln_zad_Aimp`)

The adaptive-implicit vertical advection option in NEMO is based on the work of (Shchepetkin, 2015). In common with most ocean models, the timestep used with NEMO needs to satisfy multiple criteria associated with different physical processes in order to maintain numerical stability. (Shchepetkin, 2015) pointed out that the vertical CFL criterion is commonly the most limiting. (Lemarié et al., 2015) examined the constraints for a range of time and space discretizations and provide the CFL stability criteria for a range of advection schemes. The values for the Leap-Frog with Robert Asselin filter time-stepping (as used in NEMO) are reproduced in table 9.2. Treating the vertical advection implicitly can avoid these restrictions but at the cost of large dispersive errors and, possibly, large numerical viscosity. The adaptive-implicit vertical advection option provides a targetted use of the implicit scheme only when and where potential breaches of the vertical CFL condition occur. In many practical applications these events may occur remote from the main area of interest or due to short-lived conditions such that the extra numerical diffusion or viscosity does not greatly affect the overall solution. With such applications, setting: `ln_zad_Aimp=.true.` should allow much longer model timesteps to be used whilst retaining the accuracy of the high order explicit schemes over most of the domain.

spatial discretization	$2^n d$ order centered	$3^r d$ order upwind	$4^t h$ order compact
advective CFL criterion	0.904	0.472	0.522

Table 9.2.: The advective CFL criteria for a range of spatial discretizations for the leapfrog with Robert Asselin filter time-stepping ($\nu = 0.1$) as given in (Lemarié et al., 2015).

In particular, the advection scheme remains explicit everywhere except where and when local vertical velocities exceed a threshold set just below the explicit stability limit. Once the threshold is reached a tapered transition towards an implicit scheme is used by partitioning the vertical velocity into a part that can be treated explicitly and any excess that must be treated implicitly. The partitioning is achieved via a Courant-number dependent weighting algorithm as described in (Shchepetkin, 2015).

The local cell Courant number (Cu) used for this partitioning is:

$$Cu = \frac{2\Delta t}{e_{3t_{ijk}}^n} \left([\text{Max}(w_{ijk}^n, 0.0) - \text{Min}(w_{ijk+1}^n, 0.0)] \right. \\ + [\text{Max}(e_{2u_{ij}} e_{3u_{ijk}}^n u_{ijk}^n, 0.0) - \text{Min}(e_{2u_{i-1j}} e_{3u_{i-1jk}}^n u_{i-1jk}^n, 0.0)] / e_{1t_{ij}} e_{2t_{ij}} \\ \left. + [\text{Max}(e_{1v_{ij}} e_{3v_{ijk}}^n v_{ijk}^n, 0.0) - \text{Min}(e_{1v_{ij-1}} e_{3v_{ij-1k}}^n v_{ij-1k}^n, 0.0)] / e_{1t_{ij}} e_{2t_{ij}} \right) \quad (9.33)$$

and the tapering algorithm follows (Shchepetkin, 2015) as:

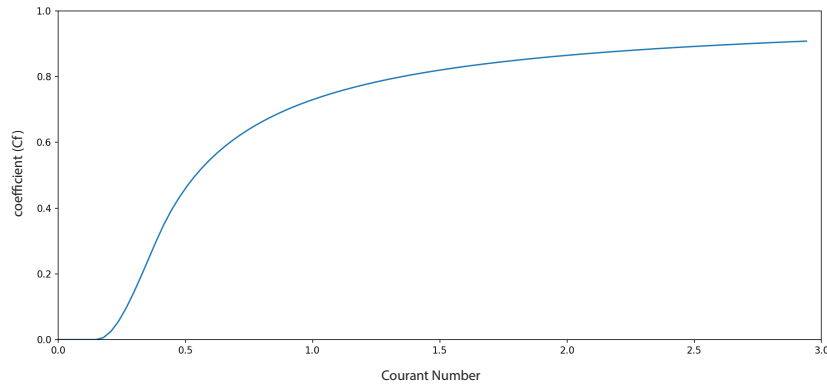


Figure 9.6.: The value of the partitioning coefficient (Cf) used to partition vertical velocities into parts to be treated implicitly and explicitly for a range of typical Courant numbers (`ln_zad_Aimp=.true.`).

$$\begin{aligned}
 Cu_{min} &= 0.15 \\
 Cu_{max} &= 0.3 \\
 Cu_{cut} &= 2Cu_{max} - Cu_{min} \\
 Fcu &= 4Cu_{max} * (Cu_{max} - Cu_{min}) \\
 Cf &= \begin{cases} 0.0 & \text{if } Cu \leq Cu_{min} \\ (Cu - Cu_{min})^2 / (Fcu + (Cu - Cu_{min})^2) & \text{else if } Cu < Cu_{cut} \\ (Cu - Cu_{max}) / Cu & \text{else} \end{cases} \quad (9.34)
 \end{aligned}$$

The partitioning coefficient is used to determine the part of the vertical velocity that must be handled implicitly (w_i) and to subtract this from the total vertical velocity (w_n) to leave that which can continue to be handled explicitly:

$$\begin{aligned}
 w_{i_{ijk}} &= Cf_{ijk} w_{n_{ijk}} \\
 w_{n_{ijk}} &= (1 - Cf_{ijk}) w_{n_{ijk}} \quad (9.35)
 \end{aligned}$$

Note that the coefficient is such that the treatment is never fully implicit; the three cases from [equation 9.34](#) can be considered as: fully-explicit; mixed explicit/implicit and mostly-implicit. With the settings shown the coefficient (Cf) varies as shown in [figure 9.6](#). Note with these values the Cu_{cut} boundary between the mixed implicit-explicit treatment and 'mostly implicit' is 0.45 which is just below the stability limited given in [table 9.2](#) for a 3rd order scheme.

The w_i component is added to the implicit solvers for the vertical mixing in `dynzdf.F90` and `trazdf.F90` in a similar way to ([Shchepetkin, 2015](#)). This is sufficient for the flux-limited advection scheme (`ln_traadv_mus`) but further intervention is required when using the flux-corrected scheme (`ln_traadv_fct`). For these schemes the implicit upstream fluxes must be added to both the monotonic guess and to the higher order solution when calculating the antidiffusive fluxes. The implicit vertical fluxes are then removed since they are added by the implicit solver later on.

The adaptive-implicit vertical advection option is new to NEMO at v4.0 and has yet to be used in a wide range of simulations. The following test simulation, however, does illustrate the potential benefits and will hopefully encourage further testing and feedback from users:

9.7.1. Adaptive-implicit vertical advection in the OVERFLOW test-case

The [OVERFLOW test case](#) provides a simple illustration of the adaptive-implicit advection in action. The example here differs from the basic test case by only a few extra physics choices namely:

```

ln_dynldf_OFF = .false.
ln_dynldf_lap = .true.
ln_dynldf_hor = .true.
ln_zdfnpc     = .true.
ln_traadv_fct = .true.
  nn_fct_h    = 2
  nn_fct_v    = 2
    
```

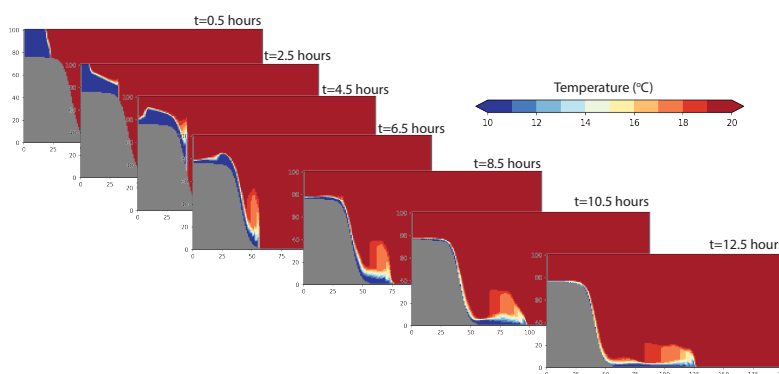


Figure 9.7.: A time-series of temperature vertical cross-sections for the OVERFLOW test case. These results are for the default settings with `nn_rdt=10.0` and without adaptive implicit vertical advection (`ln_zad_Aimp=.false.`).

which were chosen to provide a slightly more stable and less noisy solution. The result when using the default value of `nn_rdt=10.` without adaptive-implicit vertical velocity is illustrated in figure 9.7. The mass of cold water, initially sitting on the shelf, moves down the slope and forms a bottom-trapped, dense plume. Even with these extra physics choices the model is close to stability limits and attempts with `nn_rdt=30.` will fail after about 5.5 hours with excessively high horizontal velocities. This time-scale corresponds with the time the plume reaches the steepest part of the topography and, although detected as a horizontal CFL breach, the instability originates from a breach of the vertical CFL limit. This is a good candidate, therefore, for use of the adaptive-implicit vertical advection scheme.

The results with `ln_zad_Aimp=.true.` and a variety of model timesteps are shown in figure 9.8 (together with the equivalent frames from the base run). In this simple example the use of the adaptive-implicit vertical advection scheme has enabled a 12x increase in the model timestep without significantly altering the solution (although at this extreme the plume is more diffuse and has not travelled so far). Notably, the solution with and without the scheme is slightly different even with `nn_rdt=10.`; suggesting that the base run was close enough to instability to trigger the scheme despite completing successfully. To assist in diagnosing how active the scheme is, in both location and time, the 3D implicit and explicit components of the vertical velocity are available via XIOS as `wimp` and `wexp` respectively. Likewise, the partitioning coefficient (C_f) is also available as `wi_cff`. For a quick oversight of the schemes activity the global maximum values of the absolute implicit component of the vertical velocity and the partitioning coefficient are written to the netCDF version of the run statistics file (`run.stat.nc`) if this is active (see section 14.5 for activation details).

figure 9.9 shows examples of the maximum partitioning coefficient for the various overflow tests. Note that the adaptive-implicit vertical advection scheme is active even in the base run with `nn_rdt=10.0s` adding to the evidence that the test case is close to stability limits even with this value. At the larger timesteps, the vertical velocity is treated mostly implicitly at some location throughout the run. The oscillatory nature of this measure appears to be linked to the progress of the plume front as each cusp is associated with the location of the maximum shifting to the adjacent cell. This is illustrated in figure 9.10 where the i - and k - locations of the maximum have been overlaid for the base run case.

Only limited tests have been performed in more realistic configurations. In the ORCA2_ICE_PISCES reference configuration the scheme does activate and passes restartability and reproducibility tests but it is unable to improve the model's stability enough to allow an increase in the model time-step. A view of the time-series of maximum partitioning coefficient (not shown here) suggests that the default time-step of 5400s is already pushing at stability limits, especially in the initial start-up phase. The time-series does not, however, exhibit any of the 'cuspidity' found with the overflow tests.

A short test with an eORCA1 configuration promises more since a test using a time-step of 3600s remains stable with `ln_zad_Aimp=.true.` whereas the time-step is limited to 2700s without.

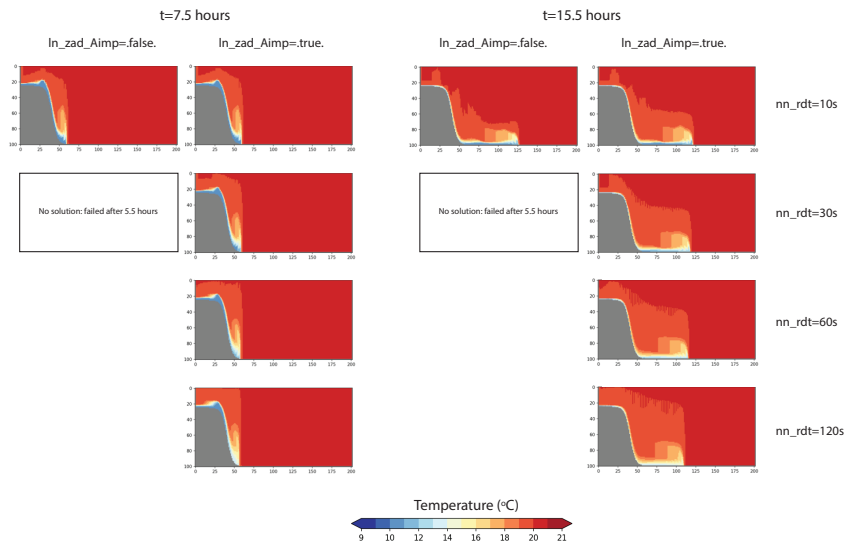


Figure 9.8.: Sample temperature vertical cross-sections from mid- and end-run using different values for `nn_rdt` and with or without adaptive implicit vertical advection. Without the adaptive implicit vertical advection only the run with the shortest timestep is able to run to completion. Note also that the colour-scale has been chosen to confirm that temperatures remain within the original range of 10° to 20° .

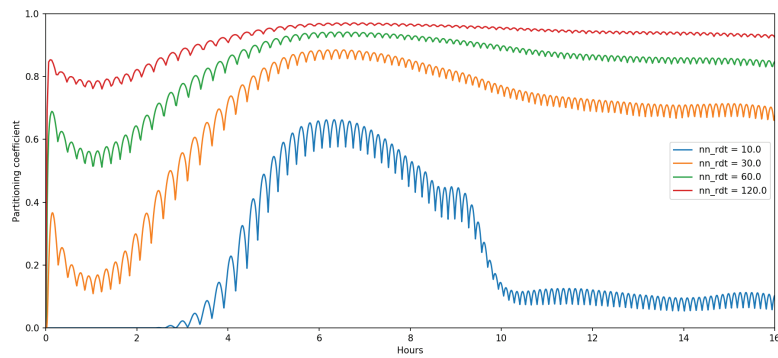


Figure 9.9.: The maximum partitioning coefficient during a series of test runs with increasing model timestep length. At the larger timesteps, the vertical velocity is treated mostly implicitly at some location throughout the run.

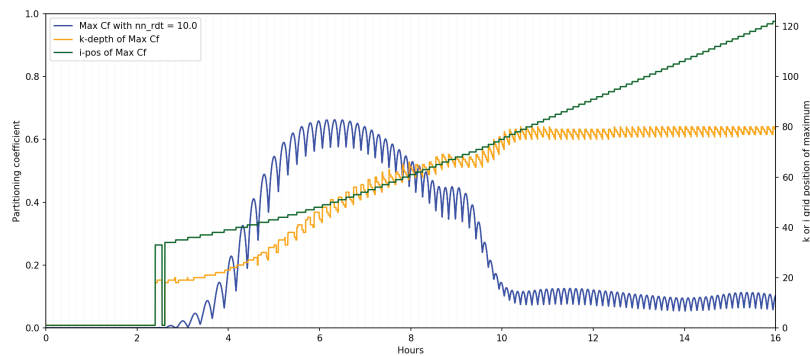


Figure 9.10.: The maximum partitioning coefficient for the `nn_rdt=10.0` case overlaid with information on the gridcell `i`- and `k`-locations of the maximum value.

Table of contents

10.1. Model output	137
10.2. Standard model output (IOM)	137
10.2.1. XIOS: Reading and writing restart file	138
10.2.2. XIOS: XML Inputs-Outputs Server	138
10.2.3. Practical issues	139
10.2.4. XML fundamentals	140
10.2.5. Detailed functionalities	142
10.2.6. XML reference tables	144
10.2.7. CF metadata standard compliance	145
10.3. NetCDF4 support (key_netcdf4)	149
10.4. Tracer/Dynamics trends (&namtrd)	150
10.5. FLO: On-Line Floats trajectories (key_floats)	151
10.6. Harmonic analysis of tidal constituents (key_diaharm)	153
10.7. Transports across sections (key_diadct)	153
10.8. Diagnosing the steric effect in sea surface height	155
10.9. Other diagnostics	157
10.9.1. Depth of various quantities (<i>diahth.F90</i>)	157
10.9.2. CMIP specific diagnostics (<i>diar5.F90, diapr.F90</i>)	157
10.9.3. 25 hour mean output for tidal models	157
10.9.4. Top middle and bed hourly output	158
10.9.5. Courant numbers	158

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

10.1. Model output

The model outputs are of three types: the restart file, the output listing, and the diagnostic 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 contains two consecutive time steps for all the prognostic variables.

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.

By default, diagnostic output files are written in NetCDF format. Since version 3.2, when defining `key_iomput`, an I/O server has been added which provides more flexibility in the choice of the fields to be written as well as how the writing work is distributed over the processors in massively parallel computing. A complete description of the use of this I/O server is presented in the next section.

10.2. Standard model output (IOM)

Since version 3.2, `iomput` is the *NEMO* output interface of choice. It has been designed to be simple to use, flexible and efficient. The two main purposes of `iomput` are:

1. The complete and flexible control of the output files through external XML files adapted by the user from standard templates.
2. To achieve high performance and scalable output through the optional distribution of all diagnostic output related tasks to dedicated processes.

The first functionality allows the user to specify, without code changes or recompilation, aspects of the diagnostic output stream, such as:

- The choice of output frequencies that can be different for each file (including real months and years).
- The choice of file contents; includes complete flexibility over which data are written in which files (the same data can be written in different files).
- The possibility to split output files at a chosen frequency.
- The possibility to extract a vertical or an horizontal subdomain.
- The choice of the temporal operation to perform, *e.g.*: average, accumulate, instantaneous, min, max and once.
- Control over metadata via a large XML "database" of possible output fields.

In addition, `iomput` allows the user to add in the code the output of any new variable (scalar, 2D or 3D) in a very easy way. All details of `iomput` functionalities are listed in the following subsections. Examples of the XML files that control the outputs can be found in: `cfgs/ORCA2_ICE_PISCES/EXPREF/iodef.xml`, `cfgs/SHARED/field_def_nemo-oce.xml`, `cfgs/SHARED/field_def_nemo-pisces.xml`, `cfgs/SHARED/field_def_nemo-ice.xml` and `cfgs/SHARED/domain_def_nemo.xml`.

The second functionality targets output performance when running in parallel (`key_mpp_mpi`). `Iomput` provides the possibility to specify `N` dedicated I/O processes (in addition to the *NEMO* processes) to collect and write the outputs. With an appropriate choice of `N` by the user, the bottleneck associated with the writing of the output files can be greatly reduced.

In version 3.6, the `iomput` interface depends on an external code called [XIOS-2.5](#). This new IO server can take advantage of the parallel I/O functionality of NetCDF4 to create a single output file and therefore to bypass the rebuilding phase. Note that writing in parallel into the same NetCDF files requires that your NetCDF4 library is linked to an HDF5 library that has been correctly compiled (*i.e.* with the configure option `--enable-parallel`). Note that the files created by `iomput` through XIOS are incompatible with NetCDF3. All post-processing and visualization tools must therefore be compatible with NetCDF4 and not only NetCDF3.

Even if not using the parallel I/O functionality of NetCDF4, using `N` dedicated I/O servers, where `N` is typically much less than the number of *NEMO* processors, will reduce the number of output files created. This can greatly reduce the post-processing burden usually associated with using large numbers of *NEMO* processors. Note that for smaller configurations, the rebuilding phase can be avoided, even without a parallel-enabled NetCDF4 library, simply by employing only one dedicated I/O server.

10.2.1. XIOS: Reading and writing restart file

XIOS may be used to read single file restart produced by *NEMO*. Currently only the variables written to file `numror` can be handled by XIOS. To activate restart reading using XIOS, set `ln_xios_read=.true.` in `namelist_cfg`. This setting will be ignored when multiple restart files are present, and default *NEMO* functionality will be used for reading. There is no need to change `iodef.xml` file to use XIOS to read restart, all definitions are done within the *NEMO* code. For high resolution configuration, however, there may be a need to add the following line in `iodef.xml` (xios context):

```
<variable id="recv_field_timeout" type="double">1800</variable>
```

This variable sets timeout for reading.

If XIOS is to be used to read restart from file generated with an earlier *NEMO* version (3.6 for instance), dimension `z` defined in restart file must be renamed to `nav_lev`.

XIOS can also be used to write *NEMO* restart. A namelist parameter `nn_wxios` is used to determine the type of restart *NEMO* will write. If it is set to 0, default *NEMO* functionality will be used - each processor writes its own restart file; if it is set to 1 XIOS will write restart into a single file; for `nn_wxios=2` the restart will be written by XIOS into multiple files, one for each XIOS server. Note, however, that ***NEMO will not read restart generated by XIOS when `nn_wxios=2`***. The restart will have to be rebuild before continuing the run. This option aims to reduce number of restart files generated by *NEMO* only, and may be useful when there is a need to change number of processors used to run simulation.

If an additional variable must be written to a restart file, the following steps are needed:

1. Add variable name to a list of restart variables (in subroutine `iom_set_rst_vars`, `iom.F90`) and define correct grid for the variable (`grid_N_3D` - 3D variable, `grid_N` - 2D variable, `grid_vector` - 1D variable, `grid_scalar` - scalar),
2. Add variable to the list of fields written by restart. This can be done either in subroutine `iom_set_rstw_core` (`iom.F90`) or by calling `iom_set_rstw_active` (`iom.F90`) with the name of a variable as an argument. This convention follows approach for writing restart using `iom`, where variables are written either by `rst_write` or by calling `iom_rstput` from individual routines.

An older versions of XIOS do not support reading functionality. It's recommended to use at least XIOS2@1451.

10.2.2. XIOS: XML Inputs-Outputs Server

Attached or detached mode?

`Iomput` is based on `XIOS`, the `io_server` developed by Yann Meurdesoif from IPSL. The behaviour of the I/O subsystem is controlled by settings in the external XML files listed above. Key settings in the `iodef.xml` file are the tags associated with each defined file.

```
<variable id="using_server" type="bool"></variable>
```

The `using_server` setting determines whether or not the server will be used in *attached mode* (as a library) [`> false <`] or in *detached mode* (as an external executable on `N` additional, dedicated cpus) [`> true <`]. The *attached mode* is simpler to use but much less efficient for massively parallel applications. The type of each file can be either "multiple_file" or "one_file".

In *attached mode* and if the type of file is "multiple_file", then each *NEMO* process will also act as an IO server and produce its own set of output files. Superficially, this emulates the standard behaviour in previous versions. However, the subdomain written out by each process does not correspond to the `jpi x jpj x jpk` domain actually computed by the process (although it may if `jpni=1`). Instead each process will have collected and written out a number of complete longitudinal strips. If the "one_file" option is chosen then all processes will collect their longitudinal strips and write (in parallel) to a single output file.

In *detached mode* and if the type of file is "multiple_file", then each stand-alone XIOS process will collect data for a range of complete longitudinal strips and write to its own set of output files. If the "one_file" option is chosen then all XIOS processes will collect their longitudinal strips and write (in parallel) to a single output file. Note running in detached mode requires launching a Multiple Process Multiple Data (MPMD) parallel job. The following subsection provides a typical example but the syntax will vary in different MPP environments.

variable name	description	example
buffer_size	buffer size used by XIOS to send data from NEMO to XIOS. Larger is more efficient. Note that needed/used buffer sizes are summarized at the end of the job	25000000
buffer_server_factor_size	ratio between NEMO and XIOS buffer size. Should be 2.	2
info_level	verbosity level (0 to 100)	0
using_server	activate attached(false) or detached(true) mode	true
using_oasis	XIOS is used with OASIS(true) or not (false)	false
oasis_codes_id	when using oasis, define the identifier of NEMO in the namcouple. Note that the identifier of XIOS is xios.x	oceanx

Number of cpu used by XIOS in detached mode

The number of cores used by the XIOS is specified when launching the model. The number of cores dedicated to XIOS should be from $\sim 1/10$ to $\sim 1/50$ of the number of cores dedicated to NEMO. Some manufacturers suggest using $O(\sqrt{N})$ dedicated IO processors for N processors but this is a general recommendation and not specific to NEMO. It is difficult to provide precise recommendations because the optimal choice will depend on the particular hardware properties of the target system (parallel filesystem performance, available memory, memory bandwidth etc.) and the volume and frequency of data to be created. Here is an example of 2 cpus for the io_server and 62 cpu for nemo using mpirun:

```
mpirun -np 62 ./nemo.exe : -np 2 ./xios_server.exe
```

Control of XIOS: the context in iodef.xml

As well as the using_server flag, other controls on the use of XIOS are set in the XIOS context in iodef.xml. See the XML basics section below for more details on XML syntax and rules.

10.2.3. Practical issues

Installation

As mentioned, XIOS is supported separately and must be downloaded and compiled before it can be used with NEMO. See the installation guide on the XIOS wiki for help and guidance. NEMO will need to link to the compiled XIOS library. The Extract and install XIOS guide provides an example illustration of how this can be achieved.

Add your own outputs

It is very easy to add your own outputs with iomput. Many standard fields and diagnostics are already prepared (i.e., steps 1 to 3 below have been done) and simply need to be activated by including the required output in a file definition in iodef.xml (step 4). To add new output variables, all 4 of the following steps must be taken.

1. in NEMO code, add a `CALL iom_put ('identifier', array)` where you want to output a 2D or 3D array.
2. If necessary, add `USE iom ! I/O manager library` to the list of used modules in the upper part of your module.
3. in the field_def.xml file, add the definition of your variable using the same identifier you used in the f90 code (see subsequent sections for a details of the XML syntax and rules). For example:

```
<field_definition>
^I<field_group id="grid_T" grid_ref="grid_T_3D"> <!-- T grid -->
^I...
^I^I<field id="identifier" long_name="blabla" ... />
^I^I^I...
</field_definition>
```

Note your definition must be added to the field_group whose reference grid is consistent with the size of the array passed to iomput. The grid_ref attribute refers to definitions set in iodef.xml which, in turn, reference grids and axes either defined in the code (iom_set_domain_attr and iom_set_axis_attr in iom.F90) or defined in the domain_def.xml file. e.g.:

flavor	description	example
root	declaration of the root element that can contain element groups or elements	<code><file_definition ... ></code>
group	declaration of a group element that can contain element groups or elements	<code><file_group ... ></code>
element	declaration of an element that can contain elements	<code><file ... ></code>

context	description	example
context xios	context containing information for XIOS	<code><context id="xios" ... ></code>
context nemo	context containing IO information for <i>NEMO</i> (mother grid when using AGRIF)	<code><context id="nemo" ... ></code>
context l_nemo	context containing IO information for <i>NEMO</i> child grid l (when using AGRIF)	<code><context id="l_nemo" ... ></code>
context n_nemo	context containing IO information for <i>NEMO</i> child grid n (when using AGRIF)	<code><context id="n_nemo" ... ></code>

```
<grid id="grid_T_3D" domain_ref="grid_T" axis_ref="deptht"/>
```

Note, if your array is computed within the surface module each `nn_fsbc` `time_step`, add the field definition within the `field_group` defined with the id "SBC": `<field_group id="SBC" ... >` which has been defined with the correct frequency of operations (`iom_set_field_attr` in *iom.F90*)

- add your field in one of the output files defined in `iodef.xml` (again see subsequent sections for syntax and rules)

```
<file id="file1" ... />
...
^^I<field field_ref="identifier" />
^^I...
</file>
```

10.2.4. XML fundamentals

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 used in XIOS is structured by 7 families of tags: context, axis, domain, grid, field, file and variable. Each tag family has hierarchy of three flavors (except for context):

Each element may have several attributes. Some attributes are mandatory, other are optional but have a default value and other are completely optional. Id is a special attribute used to identify an element or a group of elements. It must be unique for a kind of element. It is optional, but no reference to the corresponding element can be done if it is not defined.

The XML file is split into context tags that are used to isolate IO definition from different codes or different parts of a code. No interference is possible between 2 different contexts. Each context has its own calendar and an associated timestep. In *NEMO*, we used the following contexts (that can be defined in any order):

The xios context contains only 1 tag:

Each context tag related to *NEMO* (mother or child grids) is divided into 5 parts (that can be defined in any order):

context tag	description	example
variable_definition	define variables needed by XIOS. This can be seen as a kind of namelist for XIOS.	<code><variable_definition ... ></code>

context tag	description	example
field_definition	define all variables that can potentially be out- puted	<code><field_definition ... ></code>
file_definition	define the netcdf files to be created and the vari- ables they will contain	<code><file_definition ... ></code>
axis_definition	define vertical axis	<code><axis_definition ... ></code>
domain_definition	define the horizontal grids	<code><domain_definition ... ></code>
grid_definition	define the 2D and 3D grids (association of an axis and a domain)	<code><grid_definition ... ></code>

Nesting XML files

The XML file can be split in different parts to improve its readability and facilitate its use. The inclusion of XML files into the main XML file can be done through the attribute src:

```
<context src="./nemo_def.xml" />
```

In *NEMO*, by default, the field definition is done in 3 separate files (`cfgs/SHARED/field_def_nemo-oce.xml`, `cfgs/SHARED/field_def_nemo-pisces.xml` and `cfgs/SHARED/field_def_nemo-ice.xml`) and the domain definition is done in another file (`cfgs/SHARED/domain_def_nemo.xml`) that are included in the main `iodef.xml` file through the following commands:

```
<context id="nemo" src="./context_nemo.xml"/>
```

Use of inheritance

XML extensively uses the concept of inheritance. XML has a tree based structure with a parent-child oriented relation: all children inherit attributes from parent, but an attribute defined in a child replace the inherited attribute value. Note that the special attribute "id" is never inherited.

example 1: Direct inheritance.

```
<field_definition operation="average" >
^^I<field id="sst" /> <!-- averaged sst -->
^^I<field id="sss" operation="instant"/> <!-- instantaneous sss -->
</field_definition>
```

The field "sst" which is part (or a child) of the field_definition will inherit the value "average" of the attribute "operation" from its parent. Note that a child can overwrite the attribute definition inherited from its parents. In the example above, the field "sss" will for example output instantaneous values instead of average values.

example 2: Inheritance by reference.

```
<field_definition>
^^I<field id="sst" long_name="sea surface temperature" />
^^I<field id="sss" long_name="sea surface salinity" />
</field_definition>
<file_definition>
^^I<file id="myfile" output_freq="1d" />
^^I^^I<field field_ref="sst" /> <!-- default def -->
^^I^^I<field field_ref="sss" long_name="my description" /> <!-- overwrite -->
^^I</file>
</file_definition>
```

Inherit (and overwrite, if needed) the attributes of a tag you are referring to.

Use of groups

Groups can be used for 2 purposes. Firstly, the group can be used to define common attributes to be shared by the elements of the group through inheritance. In the following example, we define a group of field that will share a common grid "grid_T_2D". Note that for the field "toce", we overwrite the grid definition inherited from the group by "grid_T_3D".

```
<field_group id="grid_T" grid_ref="grid_T_2D">
^^I<field id="toce" long_name="temperature" unit="degC" grid_ref="grid_T_3D"/>
^^I<field id="sst" long_name="sea surface temperature" unit="degC" />
^^I<field id="sss" long_name="sea surface salinity" unit="psu" />
^^I<field id="ssh" long_name="sea surface height" unit="m" />
^^I...
```

Secondly, the group can be used to replace a list of elements. Several examples of groups of fields are proposed at the end of the XML field files (`cfgs/SHARED/field_def_nemo-oce.xml`, `cfgs/SHARED/field_def_nemo-pisces.xml` and `cfgs/SHARED/field_def_nemo-ice.xml`). For example, a short list of the usual variables related to the U grid:

```
<field_group id="groupU" >
^^I<field field_ref="uoc" />
^^I<field field_ref="ssu" />
^^I<field field_ref="utau" />
</field_group>
```

that can be directly included in a file through the following syntax:

```
<file id="myfile_U" output_freq="1d" />
^^I<field_group group_ref="groupU" />
^^I<field field_ref="uocetr_eff" /> <!-- add another field -->
</file>
```

10.2.5. Detailed functionalities

The file `NEMOGCM/CONFIG/ORCA2_LIM/iodef_demo.xml` provides several examples of the use of the new functionalities offered by the XML interface of XIOS.

Define horizontal subdomains

Horizontal subdomains are defined through the attributes `zoom_ibegin`, `zoom_jbegin`, `zoom_ni`, `zoom_nj` of the tag family `domain`. It must therefore be done in the domain part of the XML file. For example, in `cfgs/SHARED/domain_def.xml`, we provide the following example of a definition of a 5 by 5 box with the bottom left corner at point (10,10).

```
<domain id="myzoomT" domain_ref="grid_T">
^^I<zoom_domain ibegin="10" jbegin="10" ni="5" nj="5" />
```

The use of this subdomain is done through the redefinition of the attribute `domain_ref` of the tag family `field`. For example:

```
<file id="myfile_vzoom" output_freq="1d" >
^^I<field field_ref="toce" domain_ref="myzoomT"/>
</file>
```

Moorings are seen as an extrem case corresponding to a 1 by 1 subdomain. The Equatorial section, the TAO, RAMA and PIRATA moorings are already registered in the code and can therefore be outputted without taking care of their (i,j) position in the grid. These predefined domains can be activated by the use of specific `domain_ref`: "EqT", "EqU" or "EqW" for the equatorial sections and the mooring position for TAO, RAMA and PIRATA followed by "T" (for example: "8s137eT", "1.5s80.5eT" ...)

```
<file id="myfile_vzoom" output_freq="1d" >
^^I<field field_ref="toce" domain_ref="0n180wT"/>
</file>
```

Note that if the domain decomposition used in XIOS cuts the subdomain in several parts and if you use the "multiple_file" type for your output files, you will end up with several files you will need to rebuild using unprovided tools (like `ncpdq` and `ncrcat`, see [nco manual](#)). We are therefore advising to use the "one_file" type in this case.

Define vertical zooms

Vertical zooms are defined through the attributes `zoom_begin` and `zoom_n` of the tag family `axis`. It must therefore be done in the axis part of the XML file. For example, in `cfgs/ORCA2_ICE_PISCES/EXPREF/iodef_demo.xml`, we provide the following example:

placeholder string	automatically replaced by
@expname@	the experiment name (from cn_exp in the namelist)
@freq@	output frequency (from attribute output_freq)
@startdate@	starting date of the simulation (from nn_date0 in the restart or the namelist). yyyymmdd format
@startdatefull@	starting date of the simulation (from nn_date0 in the restart or the namelist). yyyymmdd_hh:mm:ss format
@enddate@	ending date of the simulation (from nn_date0 and nn_itend in the namelist). yyyymmdd format
@enddatefull@	ending date of the simulation (from nn_date0 and nn_itend in the namelist). yyyymmdd_hh:mm:ss format

```
<axis_definition>
^^I<axis id="depth" long_name="Vertical T levels" unit="m" positive="down" />
^^I<axis id="depth_zoom" azix_ref="depth" >
^^I^^I<zoom_axis zoom_begin="1" zoom_n="10" />
^^I</axis>
```

The use of this vertical zoom is done through the redefinition of the attribute axis_ref of the tag family field. For example:

```
<file id="myfile_hzoom" output_freq="1d" >
^^I<field field_ref="toce" axis_ref="depth_myzoom"/>
</file>
```

Control of the output file names

The output file names are defined by the attributes "name" and "name_suffix" of the tag family file. For example:

```
<file_group id="1d" output_freq="1d" name="myfile_1d" >
^^I<file id="myfileA" name_suffix="AAA" > <!-- will create file "myfile_1d_AAA" -->
^^I^^I...
^^I</file>
^^I<file id="myfileB" name_suffix="BBB" > <!-- will create file "myfile_1d_BBB" -->
^^I^^I...
^^I</file>
</file_group>
```

However it is often very convenient to define the file name with the name of the experiment, the output file frequency and the date of the beginning and the end of the simulation (which are informations stored either in the namelist or in the XML file). To do so, we added the following rule: if the id of the tag file is "fileN" (where N = 1 to 999 on 1 to 3 digits) or one of the predefined sections or moorings (see next subsection), the following part of the name and the name_suffix (that can be inherited) will be automatically replaced by:

For example,

```
<file id="myfile_hzoom" name="myfile_@expname@_@startdate@_@freq@freq@" output_freq="1d" >
```

with the namelist:

```
cn_exp      = "ORCA2"
nn_date0    = 19891231
ln_rstart   = .false.
```

will give the following file name radical: *myfile_ORCA2_19891231_freq1d.nc*

Other controls of the XML attributes from NEMO

The values of some attributes are defined by subroutine calls within *NEMO* (calls to *iom_set_domain_attr*, *iom_set_axis_attr* and *iom_set_field_attr* in *iom.F90*). Any definition given in the XML file will be 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
ptrc_T	freq_op	rn_rdt × nn_dttrc
diad_T	freq_op	rn_rdt × nn_dttrc
EqT, EqU, EqW	jbegin, ni, name_suffix	according to the grid
TAO, RAMA and PIRATA moorings	zoom_ibegin, zoom_jbegin, name_suffix	according to the grid

Advanced use of XIOS functionalities

10.2.6. XML reference tables

1. Simple computation: directly define the computation when referring to the variable in the file definition.

```
<field field_ref="sst" name="tosK" unit="degK" > sst + 273.15 </field>
<field field_ref="taum" name="taum2" unit="N2/m4" long_name="square of wind stress module" > ^Itaum * taum
↪ </field>
<field field_ref="qt" name="stupid_check" > qt - qsr - qns </field>
```

2. Simple computation: define a new variable and use it in the file definition.

in field_definition:

```
<field id="sst2" long_name="square of sea surface temperature" unit="degC2" > sst * sst </field >
```

in file_definition:

```
<field field_ref="sst2" > sst2 </field>
```

Note that in this case, the following syntaxe `<field field_ref="sst2" />` is not working as sst2 won't be evaluated.

3. Change of variable precision:

```
<!-- force to keep real 8 -->
<field field_ref="sst" name="tos_r8" prec="8" />
<!-- integer 2 with add_offset and scale_factor attributes -->
<field field_ref="sss" name="sos_i2" prec="2" add_offset="20." scale_factor="1.e-3" />
```

Note that, then the code is crashing, writting real4 variables forces a numerical conversion from real8 to real4 which will create an internal error in NetCDF and will avoid the creation of the output files. Forcing double precision outputs with prec="8" (for example in the field_definition) will avoid this problem.

4. add user defined attributes:

```
<file_group id="ld" output_freq="ld" output_level="10" enabled=".true." > <!-- ld files -->
^^I<file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
^^I^^I<field field_ref="sst" name="tos" >
^^I^^I^^I<variable id="my_attribute1" type="string" > blabla </variable>
^^I^^I^^I<variable id="my_attribute2" type="integer" > 3 </variable>
^^I^^I^^I<variable id="my_attribute3" type="float" > 5.0 </variable>
^^I^^I</field>
^^I^^I<variable id="my_global_attribute" type="string" > blabla_global </variable>
^^I</file>
</file_group>
```

5. use of the "@" function: example 1, weighted temporal average

- define a new variable in field_definition

```
<field id="toce_e3t" long_name="temperature * e3t" unit="degC*m" grid_ref="grid_T_3D" >toce * e3t</field>
```

- use it when defining your file.


```
<file_group id="5d" output_freq="5d" output_level="10" enabled=".true." > <!-- 5d files -->
^^I<file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
^^I^^I<field field_ref="toce" operation="instant" freq_op="5d" > @toce_e3t / @e3t </field>
^^I</file>
</file_group>
```

The `freq_op="5d"` attribute is used to define the operation frequency of the “@” function: here 5 day. The temporal operation done by the “@” is the one defined in the field definition: here we use the default, average. So, in the above case, `@toce_e3t` will do the 5-day mean of `toce*e3t`. `Operation="instant"` refers to the temporal operation to be performed on the field “`@toce_e3t / @e3t`”: here the temporal average is already done by the “@” function so we just use `instant` to do the ratio of the 2 mean values. `field_ref="toce"` means that attributes not explicitly defined, are inherited from `toce` field. Note that in this case, `freq_op` must be equal to the file `output_freq`.

6. use of the “@” function: example 2, monthly SSH standard deviation

- define a new variable in `field_definition`

```
<field id="ssh2" long_name="square of sea surface temperature" unit="degC2" > ssh * ssh </field >
```

- use it when defining your file.

```
<file_group id="1m" output_freq="1m" output_level="10" enabled=".true." > <!-- 1m files -->
^^I<file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
^^I^^I<field field_ref="ssh" name="sshstd" long_name="sea_surface_temperature_standard_deviation"
^^I^^I operation="instant" freq_op="1m" >
^^I^^I^^I sqrt ( @ssh2 - @ssh * @ssh )
^^I^^I</field>
^^I</file>
</file_group>
```

The `freq_op="1m"` attribute is used to define the operation frequency of the “@” function: here 1 month. The temporal operation done by the “@” is the one defined in the field definition: here we use the default, average. So, in the above case, `@ssh2` will do the monthly mean of `ssh*ssh`. `Operation="instant"` refers to the temporal operation to be performed on the field “`sqrt(@ssh2 - @ssh * @ssh)`”: here the temporal average is already done by the “@” function so we just use `instant`. `field_ref="ssh"` means that attributes not explicitly defined, are inherited from `ssh` field. Note that in this case, `freq_op` must be equal to the file `output_freq`.

7. use of the “@” function: example 3, monthly average of SST diurnal cycle

- define 2 new variables in `field_definition`

```
<field id="sstmax" field_ref="sst" long_name="max of sea surface temperature" operation="maximum" />
<field id="sstmin" field_ref="sst" long_name="min of sea surface temperature" operation="minimum" />
```

- use these 2 new variables when defining your file.

```
<file_group id="1m" output_freq="1m" output_level="10" enabled=".true." > <!-- 1m files -->
^^I<file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
^^I^^I<field field_ref="sst" name="sstdcy" long_name="amplitude of sst diurnal cycle" operation="average"
^^I^^I freq_op="1d" >
^^I^^I^^I @sstmax - @sstmin
^^I^^I</field>
^^I</file>
</file_group>
```

The `freq_op="1d"` attribute is used to define the operation frequency of the “@” function: here 1 day. The temporal operation done by the “@” is the one defined in the field definition: here maximum for `sstmax` and minimum for `sstmin`. So, in the above case, `@sstmax` will do the daily max and `@sstmin` the daily min. `Operation="average"` refers to the temporal operation to be performed on the field “`@sstmax - @sstmin`”: here monthly mean (of daily max - daily min of the sst). `field_ref="sst"` means that attributes not explicitly defined, are inherited from `sst` field.

Tag list per family

Attributes list per family

10.2.7. CF metadata standard compliance

Output from the XIOS IO server is compliant with [version 1.5](#) of the CF metadata standard. Therefore while a user may wish to add their own metadata to the output files (as demonstrated in example 4 of section [subsection 10.2.6](#)) the metadata should, for the most part, comply with the CF-1.5 standard.

tag name	description	accepted attribute	child of	parent of
simulation	this tag is the root tag which encapsulates all the content of the XML file	none	none	context
context	encapsulates parts of the XML file dedicated to different codes or different parts of a code	id ("xios", "nemo" or "n_nemo" for the nth AGRIF zoom), src, time_origin	simulation	all root tags: ... _definition

Table 10.1.: XIOS: context tags

tag name	description	accepted attribute	child of	parent of
field_definition	encapsulates the definition of all the fields that can potentially be outputted	axis_ref, default_value, domain_ref, enabled, grid_ref, level, operation, prec, src	context	field or field_group
field_group	encapsulates a group of fields	axis_ref, default_value, domain_ref, enabled, group_ref, grid_ref, id, level, operation, prec, src	field_definition, field_group, file	field or field_group
field	define a specific field	axis_ref, default_value, domain_ref, enabled, field_ref, grid_ref, id, level, long_name, name, operation, prec, standard_name, unit	field_definition, field_group, file	none

Table 10.2.: XIOS: field tags ("field_*")

tag name	description	accepted attribute	child of	parent of
file_definition	encapsulates the definition of all the files that will be outputted	enabled, min_digits, name, name_suffix, output_level, split_freq_format, split_freq, sync_freq, type, src	context	file or file_group
file_group	encapsulates a group of files that will be outputted	enabled, description, id, min_digits, name, name_suffix, output_freq, output_level, split_freq_format, split_freq, sync_freq, type, src	file_definition, file_group	file or file_group
file	define the contents of a file to be outputted	enabled, description, id, min_digits, name, name_suffix, output_freq, output_level, split_freq_format, split_freq, sync_freq, type, src	file_definition, file_group	file

Table 10.3.: XIOS: file tags ("file_*")

tag name	description	accepted attribute	child of	parent of
axis_definition	define all the vertical axis potentially used by the variables	src	context	axis_group, axis
axis_group	encapsulates a group of vertical axis	id, lon_name, positive, src, standard_name, unit, zoom_begin, zoom_end, zoom_size	axis_definition, axis_group	axis_group, axis
axis	define a vertical axis	id, lon_name, positive, src, standard_name, unit, zoom_begin, zoom_end, zoom_size	axis_definition, axis_group	none

Table 10.4.: XIOS: axis tags ("axis_*")

tag name	description	accepted attribute	child of	parent of
domain_definition	define all the horizontal domains potentially used by the variables	src	context	domain_group, domain
domain_group	encapsulates a group of horizontal domains	id, lon_name, src, zoom_ibegin, zoom_jbegin, zoom_ni, zoom_nj	domain_definition, domain_group	domain_group, domain
domain	define an horizontal domain	id, lon_name, src, zoom_ibegin, zoom_jbegin, zoom_ni, zoom_nj	domain_definition, domain_group	none

Table 10.5.: XIOS: domain tags ("domain_*")

tag name	description	accepted attribute	child of	parent of
grid_definition	define all the grid (association of a domain and/or an axis) potentially used by the variables	src	context	grid_group, grid
grid_group	encapsulates a group of grids	id, domain_ref, axis_ref	grid_definition, grid_group	grid_group, grid
grid	define a grid	id, domain_ref, axis_ref	grid_definition, grid_group	none

Table 10.6.: XIOS: grid tags ("grid_*")

attribute name	description	example	accepted by
axis_ref	refers to the id of a vertical axis	axis_ref="depth"	field, grid families
domain_ref	refers to the id of a domain	domain_ref="grid_T"	field or grid families
field_ref	id of the field we want to add in a file	field_ref="toce"	field
grid_ref	refers to the id of a grid	grid_ref="grid_T_2D"	field family
group_ref	refer to a group of variables	group_ref="mooring"	field_group

Table 10.7.: XIOS: reference attributes ("*_ref")

attribute name	description	example	accepted by
zoom_ibegin	starting point along x direction of the zoom. Automatically defined for TAO/RAMA/PIRATA moorings	zoom_ibegin="1"	domain family
zoom_jbegin	starting point along y direction of the zoom. Automatically defined for TAO/RAMA/PIRATA moorings	zoom_jbegin="1"	domain family
zoom_ni	zoom extent along x direction	zoom_ni="1"	domain family
zoom_nj	zoom extent along y direction	zoom_nj="1"	domain family

Table 10.8.: XIOS: domain attributes ("zoom_*")

attribute name	description	example	accepted by
min_digits	specify the minimum of digits used in the core number in the name of the NetCDF file	min_digits="4"	file family
name_suffix	suffix to be inserted after the name and before the cpu number and the ".nc" termination of a file	name_suffix="_myzoom"	file family
output_level	output priority of variables in a file: 0 (high) to 10 (low). All variables listed in the file with a level smaller or equal to output_level will be output. Other variables won't be output even if they are listed in the file.	output_level="10"	file family
split_freq	frequency at which to temporally split output files. Units can be ts (timestep), y, mo, d, h, mi, s. Useful for long runs to prevent over-sized output files.	split_freq="1mo"	file family
split_freq_format	date format used in the name of temporally split output files. Can be specified using the following syntaxes: %y, %mo, %d, %h %mi and %s	split_freq_format= "%y%mo%d"	file family
sync_freq	NetCDF file synchronization frequency (update of the time_counter). Units can be ts (timestep), y, mo, d, h, mi, s.	sync_freq="10d"	file family
type (1)	specify if the output files are to be split spatially (multiple_file) or not (one_file)	type="multiple_file"	file family

Table 10.9.: XIOS: file attributes

attribute name	description	example	accepted by
default_value	missing_value definition	default_value="1.e20"	field family
level	output priority of a field: 0 (high) to 10 (low)	level="1"	field family
operation	type of temporal operation: average, accumulate, instantaneous, min, max and once	operation="average"	field family
output_freq	operation frequency. units can be ts (timestep), y, mo, d, h, mi, s.	output_freq="1d12h"	field family
prec	output precision: real 4 or real 8	prec="4"	field family
long_name	define the long_name attribute in the NetCDF file	long_name="Vertical T levels"	field
standard_name	define the standard_name attribute in the NetCDF file	standard_name= "Eastward_Sea_Ice_Transport"	field

Table 10.10.: XIOS: field attributes

attribute name	description	example	accepted by
enabled	switch on/off the output of a field or a file	enabled=".true."	field, file families
description	just for information, not used	description="ocean T grid variables"	all tags
id	allow to identify a tag	id="nemo"	accepted by all tags except simulation
name	name of a variable or a file. If the name of a file is undefined, its id is used as a name	name="tos"	field or file families
positive	convention used for the orientation of vertical axis (positive downward in <i>NEMO</i>).	positive="down"	axis family
src	allow to include a file	src="./field_def.xml"	accepted by all tags except simulation
time_origin	specify the origin of the time counter	time_origin="1900-01-01 00:00:00"	context
type (2)	define the type of a variable tag	type="boolean"	variable
unit	unit of a variable or the vertical axis	unit="m"	field and axis families

Table 10.11.: XIOS: miscellaneous attributes

```

!-----
&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
/

```

namelist 10.1.: &namnc4

Some metadata that may significantly increase the file size (horizontal cell areas and vertices) are controlled by the namelist parameter `ln_cfmeta` in the `&namrun` (namelist 2.1) namelist. This must be set to true if these metadata are to be included in the output files.

10.3. 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 10.1) namelist:

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

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%

Table 10.12.: Filesize comparison between NetCDF3 and NetCDF4 with chunking and compression

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 chunk sizes 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 10.12 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.

When **key_ioinput** is activated with **key_netcdf4** chunking and compression parameters for fields produced via `io_put` calls are set via an equivalent and identically named namelist to `&namnc4` (namelist 10.1) in `xmllo_server.def`. Typically this namelist serves the mean files whilst the `&namnc4` (namelist 10.1) 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 individual processing regions and different chunking choices may be desired.

10.4. Tracer/Dynamics trends (&namtrd (namelist 10.2))

```

!-----
&namtrd      !   trend diagnostics                               (default: OFF)
!-----
ln_glo_trd   = .false.    ! (T) global domain averaged diag for T, T^2, KE, and PE
ln_dyn_trd   = .false.    ! (T) 3D momentum trend output
ln_dyn_mxl   = .false.    ! (T) 2D momentum trends averaged over the mixed layer (not coded yet)
ln_vor_trd   = .false.    ! (T) 2D barotropic vorticity trends (not coded yet)
ln_KE_trd    = .false.    ! (T) 3D Kinetic Energy trends
ln_PE_trd    = .false.    ! (T) 3D Potential Energy trends
ln_tra_trd   = .false.    ! (T) 3D tracer trend output
ln_tra_mxl   = .false.    ! (T) 2D tracer trends averaged over the mixed layer (not coded yet)
nn_trd       = 365        ! print frequency (ln_glo_trd=T) (unit=time step)
/

```

namelist 10.2.: &namtrd

```

!-----
&namflo      !   float parameters                               (default: OFF)
!-----
ln_floats    = .false.    ! activate floats or not
jpnfl        = 1          ! total number of floats during the run
jpnnewflo    = 0          ! number of floats for the restart
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)
ln_ariane    = .true.     ! Input with Ariane tool convention(T)
ln_flo_ascii = .true.     ! Output with Ariane tool netcdf convention(F) or ascii file (T)
/

```

namelist 10.3.: &namflo

Each trend of the dynamics and/or temperature and salinity time evolution equations can be sent to *trddyn.F90* and/or *trdtra.F90* modules (see TRD directory) just after their computation (*i.e.* at the end of each *dyn....F90* and/or *tra....F90* routines). This capability is controlled by options offered in `&namtrd` (namelist 10.2) namelist. Note that the output are done with XIOS, and therefore the `key_iomput` is required.

What is done depends on the `&namtrd` (namelist 10.2) logical set to `.true.`:

ln_glo_trd : at each `nn_trd` time-step a check of the basin averaged properties of the momentum and tracer equations is performed. This also includes a check of T^2 , S^2 , $\frac{1}{2}(u^2 + v^2)$, and potential energy time evolution equations properties;

ln_dyn_trd : each 3D trend of the evolution of the two momentum components is output;

ln_dyn_mxl : each 3D trend of the evolution of the two momentum components averaged over the mixed layer is output;

ln_vor_trd : a vertical summation of the moment tendencies is performed, then the curl is computed to obtain the barotropic vorticity tendencies which are output;

ln_KE_trd : each 3D trend of the Kinetic Energy equation is output;

ln_tra_trd : each 3D trend of the evolution of temperature and salinity is output;

ln_tra_mxl : each 2D trend of the evolution of temperature and salinity averaged over the mixed layer is output;

Note that the mixed layer tendency diagnostic can also be used on biogeochemical models via the `key_trdtrc` and `key_trdmxl_trc` CPP keys.

Note that in the current version (v3.6), many changes has been introduced but not fully tested. In particular, options associated with `ln_dyn_mxl`, `ln_vor_trd`, and `ln_tra_mxl` are not working, and none of the options have been tested with variable volume (*i.e.* `ln_linssh=.true.`).

10.5. FLO: On-Line Floats trajectories (`key_floats`)

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. Options are defined by `&namflo` (namelist 10.3) namelist variables. The algorithm used is based either on the work of [Blanke and Raynaud \(1997\)](#) (default option), or on a 4th Runge-Hutta algorithm (`ln_florck4=.true.`). Note that the [Blanke and Raynaud \(1997\)](#) algorithm have the advantage of providing trajectories which are consistent with the numeric of the code, so that the trajectories never intercept the bathymetry.

Input data: initial coordinates

Initial coordinates can be given with Ariane Tools convention (IJK coordinates, (`ln_ariane=.true.`)) or with longitude and latitude.

In case of Ariane convention, input filename is `init_float_ariane`. Its format is:

```
I J K nisobfl itrash
```

with:

- I,J,K : indexes of initial position
- nisobfl: 0 for an isobar float, 1 for a float following the w velocity
- itrash : set to zero; it is a dummy variable to respect Ariane Tools convention

Example:

```
100.00000 90.00000 -1.50000 1.00000 0.00000
102.00000 90.00000 -1.50000 1.00000 0.00000
104.00000 90.00000 -1.50000 1.00000 0.00000
106.00000 90.00000 -1.50000 1.00000 0.00000
108.00000 90.00000 -1.50000 1.00000 0.00000
```

In the other case (longitude and latitude), input filename is `init_float`. Its format is:

```
Long Lat depth nisobfl ngrpfl itrash
```

with:

- Long, Lat, depth : Longitude, latitude, depth
- nisobfl: 0 for an isobar float, 1 for a float following the w velocity
- ngrpfl : number to identify searcher group
- itrash :set to 1; it is a dummy variable.

Example:

```
20.0 0.0 0.0 0 1 1
-21.0 0.0 0.0 0 1 1
-22.0 0.0 0.0 0 1 1
-23.0 0.0 0.0 0 1 1
-24.0 0.0 0.0 0 1 1
```

`jpnfl` is the total number of floats during the run. When initial positions are read in a restart file (`ln_rstflo=.true.`), `jpnflnewflo` can be added in the initialization file.

Output data

`nn_writefl` is the frequency of writing in float output file and `nn_stockfl` is the frequency of creation of the float restart file.

Output data can be written in ascii files (`ln_flo_ascii=.true.`). In that case, output filename is `trajec_float`.

Another possibility of writing format is Netcdf (`ln_flo_ascii=.false.`) with `key_iomput` and outputs selected in `iodef.xml`. Here it is an example of specification to put in files description section:

```
<group id="ld_grid_T" name="auto" description="ocean T grid variables" > }
^^I<file id="floats" description="floats variables"> }
^^I^^I<field ref="traj_lon" name="floats_longitude" freq_op="86400" />}
^^I^^I<field ref="traj_lat" name="floats_latitude" freq_op="86400" />}
^^I^^I<field ref="traj_dep" name="floats_depth" freq_op="86400" />}
^^I^^I<field ref="traj_temp" name="floats_temperature" freq_op="86400" />}
^^I^^I<field ref="traj_salt" name="floats_salinity" freq_op="86400" />}
^^I^^I<field ref="traj_dens" name="floats_density" freq_op="86400" />}
^^I^^I<field ref="traj_group" name="floats_group" freq_op="86400" />}
^^I</file>}
</group>}
```



```

!-----
&nam_diaharm ! Harmonic analysis of tidal constituents (default: OFF)
!-----
ln_diaharm = .false. ! Choose tidal harmonic output or not
nit000_han = 1 ! First time step used for harmonic analysis
nitend_han = 75 ! Last time step used for harmonic analysis
nstep_han = 15 ! Time step frequency for harmonic analysis
tname(1) = 'M2' ! Name of tidal constituents
tname(2) = 'K1' !
/

```

namelist 10.4.: &nam_diaharm

```

!-----
&nam_diadct ! transports through some sections (default: OFF)
!-----
ln_diadct = .false. ! Calculate transport thru sections or not
nn_dct = 15 ! time step frequency for transports computing
nn_dctwri = 15 ! time step frequency for transports writing
nn_secdebug = 112 ! 0 : no section to debug
! ! -1 : debug all section
! ! 0 < n : debug section number n
/

```

namelist 10.5.: &nam_diadct

10.6. Harmonic analysis of tidal constituents (key_diaharm)

A module is available to compute the amplitude and phase of tidal waves. This on-line Harmonic analysis is actived with **key_diaharm**.

Some parameters are available in namelist `&nam_diaharm` (namelist 10.4):

- `nit000_han` is the first time step used for harmonic analysis
- `nitend_han` is the last time step used for harmonic analysis
- `nstep_han` is the time step frequency for harmonic analysis
- `tname` is an array with names of tidal constituents to analyse

`nit000_han` and `nitend_han` must be between `nit000` and `nitend` of the simulation. The restart capability is not implemented.

The Harmonic analysis solve the following equation:

$$h_i - A_0 + \sum_{j=1}^{nb_ana} [A_j \cos(\nu_j t_j - \phi_j)] = e_i$$

With A_j , ν_j , ϕ_j , the amplitude, frequency and phase for each wave and e_i the error. h_i is the sea level for the time t_i and A_0 is the mean sea level.

We can rewrite this equation:

$$h_i - A_0 + \sum_{j=1}^{nb_ana} [C_j \cos(\nu_j t_j) + S_j \sin(\nu_j t_j)] = e_i$$

with $A_j = \sqrt{C_j^2 + S_j^2}$ and $\phi_j = \arctan(S_j/C_j)$.

We obtain in output C_j and S_j for each tidal wave.

10.7. Transports across sections (key_diadct)

A module is available to compute the transport of volume, heat and salt through sections. This diagnostic is actived with **key_diadct**.

Each section is defined by the coordinates of its 2 extremities. The pathways between them are constructed using tools which can be found in `tools/SECTIONS_DIADCT` and are written in a binary file `section_ijglobal.diadct` which is later read in by *NEMO* to compute on-line transports.

The on-line transports module creates three output ascii files:

- `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: 10^9Kgs^{-1})

Namelist variables in `&nam_diadct` ([namelist 10.5](#)) control how frequently the flows are summed and the time scales over which they are averaged, as well as the level of output for debugging: `nn_dct` : frequency of instantaneous transports computing `nn_dctwri` : frequency of writing (mean of instantaneous transports) `nn_debug` : debugging of the section

Creating a binary file containing the pathway of each section

In `tools/SECTIONS_DIADCT/run`, the file `list_sections.ascii_global` contains a list of all the sections that are to be computed (this list of sections is based on MERSEA project metrics).

Another file is available for the 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 (*e.g.* bounds for 2 classes);
- `okstrpond` to compute heat and salt transports, `nostrpond` if no;
- `ice` to compute surface and volume ice transports, `noice` if no.

The results of the computing of transports, and the directions of positive and negative flow do not depend on the order of the 2 extremities in this file.

If `nclass` \neq 0, the next lines contain 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

The script `job.ksh` computes the pathway for each section and creates a binary file `section_ijglobal.diadct` which is read by *NEMO*.

It is possible to use this tools for new configurations: `job.ksh` has to be updated with the coordinates file name and path.

Examples of two sections, the `ACC_Drake_Passage` with no classes, and the `ATL_Cuba_Florida` with 4 temperature classes (5 class bounds), are shown:

```
-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
```

To read the output files

The output format is:

date, time-step number, section number,

section slope coefficient	section type	direction 1	direction 2	total transport
0.	horizontal	northward	southward	postive: northward
1000.	vertical	eastward	westward	postive: eastward
$\neq 0, \neq 1000.$	diagonal	eastward	westward	postive: eastward

section name, section slope coefficient, class number,
class name, class bound 1 , classe bound2,
transport_direction1, transport_direction2,
transport_total

For sections with classes, the first `nclass-1` lines correspond to the transport for each class and the last line corresponds to the total transport summed over all classes. For sections with no classes, class number 1 corresponds to `total class` and this class is called N, meaning none.

- `transport_direction1` is the positive part of the transport (≥ 0).
- `transport_direction2` is the negative part of the transport (≤ 0).

The `section slope coefficient` gives information about the significance of transports signs and direction:

10.8. 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 explicitly represented. This approximation does not represent a serious error with respect to the flow field calculated by the model (Greatbatch, 1994), 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 (Greatbatch, 1994). 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:

$$\begin{aligned} \mathcal{M} &= \mathcal{V} \bar{\rho} \\ \mathcal{V} &= \mathcal{A} \bar{\eta} \end{aligned} \quad (10.1)$$

Temporal changes in total mass is obtained from the density conservation equation:

$$\frac{1}{e_3} \partial_t (e_3 \rho) + \nabla(\rho \mathbf{U}) = \frac{emp}{e_3} \Big|_{surface} \quad (10.2)$$

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

$$\partial_t \mathcal{M} = \mathcal{A} \overline{emp} \quad (10.3)$$

where $\overline{emp} = \int_S emp ds$ is the net mass flux through the ocean surface. Bringing equation 10.3 and the time derivative of equation 10.1 together leads to the evolution equation of the mean sea level

$$\partial_t \bar{\eta} = \frac{\overline{emp}}{\bar{\rho}} - \frac{\mathcal{V}}{\mathcal{A}} \frac{\partial_t \bar{\rho}}{\bar{\rho}} \quad (10.4)$$

The first term in equation 10.4 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, [equation 10.2](#), degenerates into the incompressibility equation:

$$\frac{1}{e_3} \partial_t(e_3) + \nabla(\mathbf{U}) = \frac{emp}{\rho_o e_3} \Big|_{surface}$$

and the global average of this equation now gives the temporal change of the total volume,

$$\partial_t \mathcal{V} = \mathcal{A} \frac{\overline{emp}}{\rho_o}$$

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 ([Greatbatch, 1994](#)), 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 ([Greatbatch, 1994](#)). 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:

$$\mathcal{M}_o = \mathcal{M} + \rho_o \eta_s \mathcal{A} \quad (10.5)$$

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. [subsection 4.8.1](#)) in [equation 10.5](#) leads to a very simple form for the steric height:

$$\eta_s = -\frac{1}{\mathcal{A}} \mathcal{D} \quad (10.6)$$

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 ([Gill \(1982\)](#), 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 [equation 10.6](#) depends on the type of free surface which is considered. In the non linear free surface case, *i.e.* `ln_linssh=.true.`, it is given by

$$\eta_s = -\frac{\sum_{i,j,k} d_a e_{1t} e_{2t} e_{3t}}{\sum_{i,j,k} e_{1t} e_{2t} e_{3t}}$$

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:

$$\eta_s = -\frac{\sum_{i,j,k} d_a e_{1t} e_{2t} e_{3t} + \sum_{i,j} d_a e_{1t} e_{2t} \eta}{\sum_{i,j,k} e_{1t} e_{2t} e_{3t} + \sum_{i,j} e_{1t} e_{2t} \eta}$$

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 ([Campin et al., 2008](#)). 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:

$$\eta_s = -\frac{1}{\mathcal{A}} \int_D d_a(T, S_o, p_o) dv$$

where S_o and p_o are the initial salinity and pressure, respectively.

Both steric and thermosteric sea level are computed in `diar5.F90`.

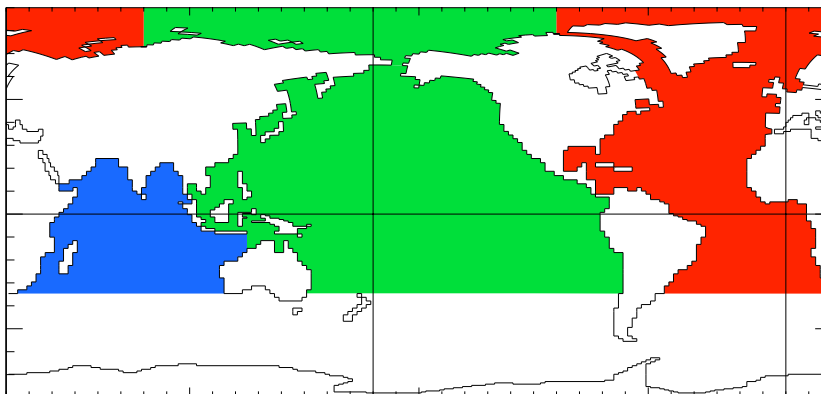


Figure 10.1.: Decomposition of the World Ocean (here ORCA2) into sub-basin used in to compute the heat and salt transports as well as the meridional stream-function: Atlantic basin (red), Pacific basin (green), Indian basin (blue), Indo-Pacific basin (blue+green). Note that semi-enclosed seas (Red, Med 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.

```

!-----
&namptr      ! Poleward Transport Diagnostic (default: OFF)
!-----
ln_diaptr   = .false. ! Poleward heat and salt transport (T) or not (F)
ln_subbas   = .false. ! Atlantic/Pacific/Indian basins computation (T) or not
/

```

namelist 10.6.: `&namptr`

10.9. Other diagnostics

Aside from the standard model variables, other diagnostics can be computed on-line. The available ready-to-add diagnostics modules can be found in directory DIA.

10.9.1. Depth of various quantities (*diahth.F90*)

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 (de Boyer Montégut et al., 2004)) (*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*)

10.9.2. CMIP specific diagnostics (*diar5.F90*)

A series of diagnostics has been added in the *diar5.F90* and *diaptr.F90*. In *diar5.F90* they correspond to outputs that are required for AR5 simulations (CMIP5) (see also section 10.8 for one of them). The module *diar5.F90* is active when one of the following outputs is required : global total volume (votot), global mean ssh (sshtot), global total mass (masstot), global mean temperature (temptot), global mean ssh steric (sshsteric), global mean ssh thermosteric (sshtster), global mean salinity (saltot), sea water pressure at sea floor (botpres), dynamic sea surface height (sshdyn).

In *diaptr.F90* when `ln_diaptr=.true.` (see the `&namptr` (namelist 10.6) namelist below) can be computed on-line the poleward heat and salt transports, their advective and diffusive component, and the meridional stream function. 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 (figure 10.1).

10.9.3. 25 hour mean output for tidal models

A module is available to compute a crudely detided M2 signal by obtaining a 25 hour mean. The 25 hour mean is available for daily runs by summing up the 25 hourly instantaneous hourly values from midnight at the start of the day to midnight at the day end. This diagnostic is activated with the logical `ln_dia25h`.

```

!-----
&nam_dia25h ! 25h Mean Output (default: OFF)
!-----
ln_dia25h = .false. ! Choose 25h mean output or not
/

```

namelist 10.7.: &nam_dia25h

```

!-----
&nam_diatmb ! Top Middle Bottom Output (default: OFF)
!-----
ln_diatmb = .false. ! Choose Top Middle and Bottom output or not
/

```

namelist 10.8.: &nam_diatmb

10.9.4. Top middle and bed hourly output

A module is available to output the surface (top), mid water and bed diagnostics of a set of standard variables. This can be a useful diagnostic when hourly or sub-hourly output is required in high resolution tidal outputs. The tidal signal is retained but the overall data usage is cut to just three vertical levels. Also the bottom level is calculated for each cell. This diagnostic is activated with the logical *ln_diatmb*.

10.9.5. Courant numbers

Courant numbers provide a theoretical indication of the model's numerical stability. The advective Courant numbers can be calculated according to

$$C_u = |u| \frac{\Delta t}{e_{1u}}, \quad C_v = |v| \frac{\Delta t}{e_{2v}}, \quad C_w = |w| \frac{\Delta t}{e_{3w}}$$

in the zonal, meridional and vertical directions respectively. The vertical component is included although it is not strictly valid as the vertical velocity is calculated from the continuity equation rather than as a prognostic variable. Physically this represents the rate at which information is propagated across a grid cell. Values greater than 1 indicate that information is propagated across more than one grid cell in a single time step.

The variables can be activated by setting the *nn_diacfl* namelist parameter to 1 in the *&namctl* (namelist 14.1) namelist. The diagnostics will be written out to an ascii file named *cfl_diagnostics.ascii*. In this file the maximum value of C_u , C_v , and C_w are printed at each timestep along with the coordinates of where the maximum value occurs. At the end of the model run the maximum value of C_u , C_v , and C_w for the whole model run is printed along with the coordinates of each. The maximum values from the run are also copied to the *ocean.output* file.

Observation and Model Comparison (OBS)

Table of contents

11.1. Running the observation operator code example	160
11.2. Technical details (feedback type observation file headers)	161
11.2.1. Profile feedback file	161
11.2.2. Sea level anomaly feedback file	163
11.2.3. Sea surface temperature feedback file	165
11.3. Theoretical details	166
11.3.1. Horizontal interpolation and averaging methods	166
11.3.2. Grid search	168
11.3.3. Parallel aspects of horizontal interpolation	169
11.3.4. Vertical interpolation operator	169
11.4. Standalone observation operator	172
11.4.1. Concept	172
11.4.2. Using the standalone observation operator	172
11.4.3. Configuring the standalone observation operator	172
11.5. Observation utilities	173
11.5.1. Obstoos	173
11.5.2. Building the obstoos	174
11.5.3. Dataplot	175

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

The observation and model comparison code, the observation operator (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 time step. The resulting data are saved in a “feedback” file (or files). The code was originally developed for use with the NEMOVAR data assimilation code, but can be used for validation or verification of the model or with any other data assimilation system.

The OBS code is called from *nemogcm.F90* for model initialisation and to calculate the model equivalent values for observations on the 0th time step. The code is then called again after each time step from *step.F90*. The code is only activated if the `&namobs` (namelist 11.1) namelist logical `ln_diaobs` is set to true.

For all data types a 2D horizontal interpolator or averager is needed to interpolate/average 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. This now works in a generalised vertical coordinate system.

Some profile observation types (*e.g.* tropical moored buoys) are made available as daily averaged quantities. The observation operator code can be set-up to calculate the equivalent daily average model temperature fields using the `nn_profdavtypes` namelist array. Some SST observations are equivalent to a night-time average value and the observation operator code can calculate equivalent night-time average model SST fields by setting the namelist value `ln_sstnight` to true. Otherwise (by default) the model value from the nearest time step to the observation time is used.

The code is controlled by the namelist `&namobs` (namelist 11.1). See the following sections for more details on setting up the namelist.

In section 11.1 a test example of the observation operator code is introduced, including where to obtain data and how to setup the namelist. In section 11.2 some more technical details of the different observation types used are introduced, and we also show a more complete namelist. In section 11.3 some of the theoretical aspects of the observation operator are described including interpolation methods and running on multiple processors. In section 11.4 the standalone observation operator code is described. In section 11.5 we describe some utilities to help work with the files produced by the OBS code.

11.1. Running the observation operator code example

In this section an example of running the observation operator code is described using profile observation data which can be freely downloaded. It shows how to adapt an existing run and build of *NEMO* to run the observation operator. Note also the observation operator and the assimilation increments code are run in the ORCA2_ICE_OBS SETTE test.

1. Compile *NEMO*.
2. Download some EN4 data from www.metoffice.gov.uk/hadobs. 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. Compile the OBSTOOLS code in the `tools` directory using:

```
./maketools -n OBSTOOLS -m [ARCH]
```

replacing `[ARCH]` with the build architecture file for your machine. Note the tools are checked out from a separate location of the repository (under `/utils/tools`).

4. Convert the EN4 data into feedback format:

```
enact2fb.exe profiles_01.nc EN.4.1.1.f.profiles.g10.YYYYMM.nc
```

5. Include the following in the *NEMO* namelist to run the observation operator on this data:

Options are defined through the `&namobs` (namelist 11.1) namelist variables. The options `ln_t3d` and `ln_s3d` switch on the temperature and salinity profile observation operator code. The filename or array of filenames are specified using the `cn_profbbfiles` 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 `cn_gridsearch` 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 (see subsection 11.3.3).

A number of utilities are now provided to plot the feedback files, convert and recombine the files. These are explained in more detail in section 11.5. Utilities to convert other input data formats into the feedback format are also described in section 11.5.


```

!-----
&namobs      ! observation usage switch                               (default: OFF)
!-----
ln_diaobs   = .false.      ! Logical switch for the observation operator
!
ln_t3d      = .false.      ! Logical switch for T profile observations
ln_s3d      = .false.      ! Logical switch for S profile observations
ln_sla      = .false.      ! Logical switch for SLA observations
ln_sst      = .false.      ! Logical switch for SST observations
ln_sss      = .false.      ! Logical switch for SSS observations
ln_sic      = .false.      ! Logical switch for Sea Ice observations
ln_vel3d    = .false.      ! Logical switch for velocity observations
ln_altbias  = .false.      ! Logical switch for altimeter bias correction
ln_sstbias  = .false.      ! Logical switch for SST bias correction
ln_nea      = .false.      ! Logical switch for rejection of observations near land
ln_grid_global = .true.    ! Logical switch for global distribution of observations
ln_grid_search_lookup = .false. ! Logical switch for obs grid search w/lookup table
ln_ignmis   = .true.      ! Logical switch for ignoring missing files
ln_s_at_t   = .false.      ! Logical switch for computing model S at T obs if not there
ln_sstnight = .false.      ! Logical switch for calculating night-time average for SST obs
ln_bound_reject = .false.  ! Logical to remove obs near boundaries in LAMs.
ln_sla_fp_indegs = .true.  ! Logical for SLA: T=> averaging footprint is in degrees, F=> in metres
ln_sst_fp_indegs = .true.  ! Logical for SST: T=> averaging footprint is in degrees, F=> in metres
ln_sss_fp_indegs = .true.  ! Logical for SSS: T=> averaging footprint is in degrees, F=> in metres
ln_sic_fp_indegs = .true.  ! Logical for SIC: T=> averaging footprint is in degrees, F=> in metres
! All of the *files* variables below are arrays. Use namelist_cfg to add more files
cn_profbf_files = 'profiles_01.nc' ! Profile feedback input observation file names
cn_slafbf_files = 'sla_01.nc'      ! SLA feedback input observation file names
cn_sstfb_files = 'sst_01.nc'      ! SST feedback input observation file names
cn_sssfb_files = 'sss_01.nc'      ! SSS feedback input observation file names
cn_sicfb_files = 'sic_01.nc'      ! SIC feedback input observation file names
cn_velfb_files = 'vel_01.nc'      ! Velocity feedback input observation file names
cn_altbias_file = 'altbias.nc'     ! Altimeter bias input file name
cn_sstbias_files = 'sstbias.nc'    ! SST bias input file name
cn_gridsearch_file = 'gridsearch.nc' ! Grid search file name
rn_gridsearch_res = 0.5           ! Grid search resolution
rn_mdtcorr       = 1.61           ! MDT correction
rn_mdtcutoff     = 65.0           ! MDT cutoff for computed correction
rn_dobsini       = 00010101.000000 ! Initial date in window YYYYMMDD.HHMMSS
rn_dobsend       = 00010102.000000 ! Final date in window YYYYMMDD.HHMMSS
rn_sla_avglam_scl = 0.           ! E/W diameter of SLA observation footprint (metres/degrees)
rn_sla_avgphi_scl = 0.           ! N/S diameter of SLA observation footprint (metres/degrees)
rn_sst_avglam_scl = 0.           ! E/W diameter of SST observation footprint (metres/degrees)
rn_sst_avgphi_scl = 0.           ! N/S diameter of SST observation footprint (metres/degrees)
rn_sss_avglam_scl = 0.           ! E/W diameter of SSS observation footprint (metres/degrees)
rn_sss_avgphi_scl = 0.           ! N/S diameter of SSS observation footprint (metres/degrees)
rn_sic_avglam_scl = 0.           ! E/W diameter of SIC observation footprint (metres/degrees)
rn_sic_avgphi_scl = 0.           ! N/S diameter of SIC observation footprint (metres/degrees)
nn_ldint        = 0              ! Type of vertical interpolation method
nn_2dint        = 0              ! Default horizontal interpolation method
nn_2dint_sla    = 0              ! Horizontal interpolation method for SLA
nn_2dint_sst    = 0              ! Horizontal interpolation method for SST
nn_2dint_sss    = 0              ! Horizontal interpolation method for SSS
nn_2dint_sic    = 0              ! Horizontal interpolation method for SIC
nn_msshc        = 0              ! MSSH correction scheme
nn_profdavtypes = -1            ! Profile daily average types - array
/

```

namelist 11.1.: &namobs

11.2. Technical details (feedback type observation file headers)

Here we show a more complete example namelist &namobs (namelist 11.1) and also show the NetCDF headers of the observation files that may be used with the observation operator.

The observation operator code uses the feedback observation file format for all data types. 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.

11.2.1. Profile feedback file

```

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" ;
}

```

11.2.2. Sea level anomaly feedback file

```

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" ;
}

```

To use Sea Level Anomaly (SLA) data 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" ;

```

11.2.3. Sea surface temperature feedback file

```

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" ;
}

```

11.3. Theoretical details

11.3.1. Horizontal interpolation and averaging methods

For most observation types, the horizontal extent of the observation is small compared to the model grid size and so the model equivalent of the observation is calculated by interpolating from the four surrounding grid points to the observation location. Some satellite observations (*e.g.* microwave satellite SST data, or satellite SSS data) have a footprint which is similar in size or larger than the model grid size (particularly when the grid size is small). In those cases the model counterpart should be calculated by averaging the model grid points over the same size as the footprint. *NEMO* therefore has the capability to specify either an interpolation or an averaging (for surface observation types only).

The main namelist option associated with the interpolation/averaging is `nn_2dint`. This default option can be set to values from 0 to 6. Values between 0 to 4 are associated with interpolation while values 5 or 6 are associated with averaging.

- `nn_2dint=0` : Distance-weighted interpolation
- `nn_2dint=1` : Distance-weighted interpolation (small angle)
- `nn_2dint=2` : Bilinear interpolation (geographical grid)
- `nn_2dint=3` : Bilinear remapping interpolation (general grid)
- `nn_2dint=4` : Polynomial interpolation
- `nn_2dint=5` : Radial footprint averaging with diameter specified in the namelist as `rn_[var]_avglamscl` in degrees or metres (set using `ln_[var]_fp_indegs`)
- `nn_2dint=6` : Rectangular footprint averaging with E/W and N/S size specified in the namelist as `rn_[var]_avglamscl` and `rn_[var]_avgphiscl` in degrees or metres (set using `ln_[var]_fp_indegs`)

Replace [var] in the last two options with the observation type (sla, sst, sss or sic) for which the averaging is to be performed (see namelist example above). The nn_2dint default option can be overridden for surface observation types using namelist values nn_2dint_[var] where [var] is the observation type.

Below is some more detail on the various options for interpolation and averaging available in *NEMO*.

Horizontal interpolation

Consider an observation point P with longitude and latitude (λ_P, ϕ_P) and the four nearest neighbouring model grid points A, B, C and D with longitude and latitude $(\lambda_A, \phi_A), (\lambda_B, \phi_B)$ 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.:

$$x_P = \frac{1}{w} (w_A x_A + w_B x_B + w_C x_C + w_D x_D)$$

where w_A, w_B etc. are the respective weights for the model field at points A, B etc., and $w = w_A + w_B + w_C + w_D$. 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:

$$w_A = s(P, B) s(P, C) s(P, D)$$

where

$$s(P, M) = \cos^{-1} \{ \sin \phi_P \sin \phi_M + \cos \phi_P \cos \phi_M \cos(\lambda_M - \lambda_P) \}$$

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 Daley and Barker (2001):

$$s(P, M) = \sin^{-1} \left\{ \sqrt{1 - x^2} \right\}$$

where

$$x = a_M a_P + b_M b_P + c_M c_P$$

and

$$\begin{aligned} a_M &= \sin \phi_M, \\ a_P &= \sin \phi_P, \\ b_M &= \cos \phi_M \cos \phi_M, \\ b_P &= \cos \phi_P \cos \phi_P, \\ c_M &= \cos \phi_M \sin \phi_M, \\ c_P &= \cos \phi_P \sin \phi_P. \end{aligned}$$

2. **Great-Circle distance-weighted interpolation with small angle approximation.** Similar to the previous interpolation but with the distance s computed as

$$s(P, M) = \sqrt{(\phi_M - \phi_P)^2 + (\lambda_M - \lambda_P)^2 \cos^2 \phi_M}$$

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](#).

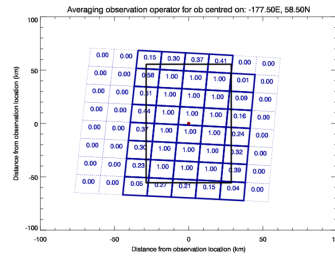


Figure 11.1.: Weights associated with each model grid box (blue lines and numbers) for an observation at -170.5°E , 56.0°N with a rectangular footprint of $1^\circ \times 1^\circ$.

Horizontal averaging

For each surface observation type:

- The standard grid-searching code is used to find the nearest model grid point to the observation location (see next subsection).
- The maximum number of grid points required for that observation in each local grid domain is calculated. Some of these points may later turn out to have zero weight depending on the shape of the footprint.
- The longitudes and latitudes of the grid points surrounding the nearest model grid box are extracted using existing MPI routines.
- The weights for each grid point associated with each observation are calculated, either for radial or rectangular footprints. For grid points completely within the footprint, the weight is one; for grid points completely outside the footprint, the weight is zero. For grid points which are partly within the footprint the ratio between the area of the footprint within the grid box and the total area of the grid box is used as the weight.
- The weighted average of the model grid points associated with each observation is calculated, and this is then given as the model counterpart of the observation.

Examples of the weights calculated for an observation with rectangular and radial footprints are shown in [figure 11.1](#) and [figure 11.2](#).

11.3.2. 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 $P(\lambda_P, \phi_P)$ denote the observation point, and let $A(\lambda_A, \phi_A)$, $B(\lambda_B, \phi_B)$, $C(\lambda_C, \phi_C)$ and $D(\lambda_D, \phi_D)$ 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

$$\begin{aligned} \mathbf{r}_{PA} \times \mathbf{r}_{PC} &= [(\lambda_A - \lambda_P)(\phi_C - \phi_P) - (\lambda_C - \lambda_P)(\phi_A - \phi_P)] \hat{\mathbf{k}} \\ \mathbf{r}_{PB} \times \mathbf{r}_{PA} &= [(\lambda_B - \lambda_P)(\phi_A - \phi_P) - (\lambda_A - \lambda_P)(\phi_B - \phi_P)] \hat{\mathbf{k}} \\ \mathbf{r}_{PC} \times \mathbf{r}_{PD} &= [(\lambda_C - \lambda_P)(\phi_D - \phi_P) - (\lambda_D - \lambda_P)(\phi_C - \phi_P)] \hat{\mathbf{k}} \\ \mathbf{r}_{PD} \times \mathbf{r}_{PB} &= [(\lambda_D - \lambda_P)(\phi_B - \phi_P) - (\lambda_B - \lambda_P)(\phi_D - \phi_P)] \hat{\mathbf{k}} \end{aligned}$$

point in the opposite direction to the unit normal $\hat{\mathbf{k}}$ (*i.e.* that the coefficients of $\hat{\mathbf{k}}$ are negative), where \mathbf{r}_{PA} , \mathbf{r}_{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](#).

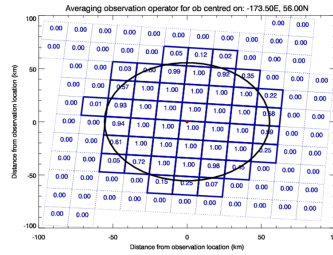


Figure 11.2.: Weights associated with each model grid box (blue lines and numbers) for an observation at -170.5°E , 56.0°N with a radial footprint with diameter 1° .

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.

11.3.3. Parallel aspects of horizontal interpolation

For horizontal interpolation, there is the basic problem that the observations are unevenly distributed on the globe. In *NEMO* the model grid is divided 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.

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 11.3](#) 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. This is under the assumption that we are dealing with point observations and 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 11.4](#) 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 decomposition with ORCA2 for the same input data as in [figure 11.3](#). 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.

11.3.4. Vertical interpolation operator

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

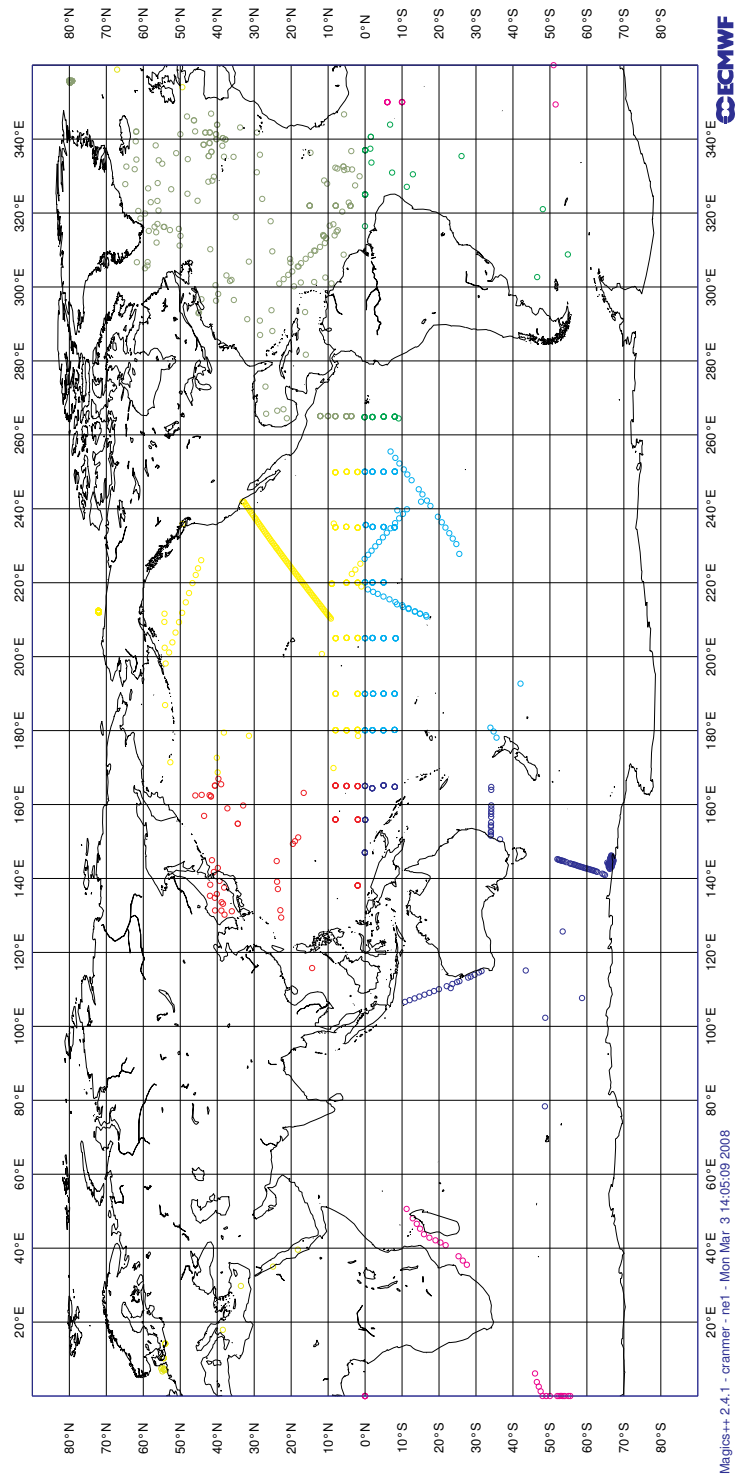


Figure 11.3.: Example of the distribution of observations with the geographical distribution of observational data

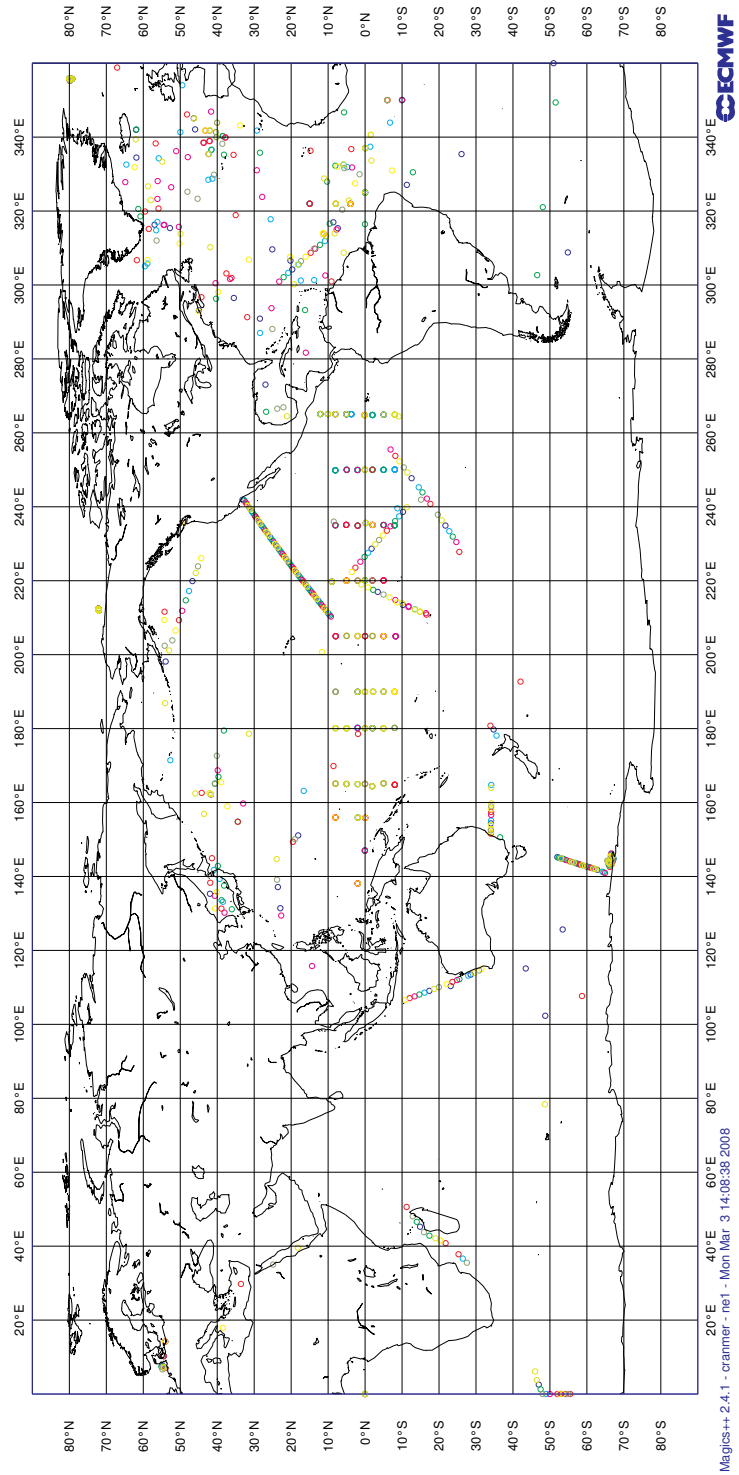


Figure 11.4.: Example of the distribution of observations with the round-robin distribution of observational data.

bottom boundary, this is done using the land-ocean mask.

For profile observation types we do both vertical and horizontal interpolation. *NEMO* has a generalised vertical coordinate system this means the vertical level depths can vary with location. Therefore, it is necessary first to perform vertical interpolation of the model value to the observation depths for each of the four surrounding grid points. After this the model values, at these points, at the observation depth, are horizontally interpolated to the observation location.

11.4. Standalone observation operator

11.4.1. Concept

The observation operator maps model variables to observation space. This is normally done while the model is running, i.e. online, it is possible to apply this mapping offline without running the model with the **standalone observation operator** (SAO). The process is divided into an initialisation phase, an interpolation phase and an output phase. During the interpolation phase the SAO populates the model arrays by reading saved model fields from disk. The interpolation and the output phases use the same OBS code described in the preceding sections.

There are two ways of exploiting the standalone capacity. The first is to mimic the behaviour of the online system by supplying model fields at regular intervals between the start and the end of the run. This approach results in a single model counterpart per observation. This kind of usage produces feedback files the same file format as the online observation operator. The second is to take advantage of the ability to run offline by calculating multiple model counterparts for each observation. In this case it is possible to consider all forecasts verifying at the same time. By forecast, we mean any method which produces an estimate of physical reality which is not an observed value.

11.4.2. Using the standalone observation operator

Building

In addition to *OPA_SRC* the SAO requires the inclusion of the *SAO_SRC* directory. *SAO_SRC* contains a replacement *nemo.F90* and *nemogcm.F90* which overwrites the resultant **nemo.exe**. Note this a similar approach to that taken by the standalone surface scheme *SAS_SRC* and the offline TOP model *OFF_SRC*.

Running

The simplest way to use the executable is to edit and append the **sao.nml** namelist to a full *NEMO* namelist and then to run the executable as if it were *nemo.exe*.

11.4.3. Configuring the standalone observation operator

The observation files and settings understood by `&namobs` (namelist 11.1) have been outlined in the online observation operator section. In addition is a further namelist `&namsao` (??) which used to set the input model fields for the SAO

Single field

In the SAO the model arrays are populated at appropriate time steps via input files. At present, **tsn** and **sshn** are populated by the default read routines. These routines will be expanded upon in future versions to allow the specification of any model variable. As such, input files must be global versions of the model domain with **votemper**, **vosaline** and optionally **sshn** present.

For each field read there must be an entry in the `&namsao` (??) namelist specifying the name of the file to read and the index along the *time_counter*. For example, to read the second time counter from a single file the namelist would be.

```
!-----
!      namsao Standalone obs_oper namelist
!-----
!   sao_files   specifies the files containing the model counterpart
!   nn_sao_idx  specifies the time_counter index within the model file
&namsao
  sao_files = "foo.nc"
  nn_sao_idx = 2
/
```

Multiple fields per run

Model field iteration is controlled via **nn_sao_freq** which specifies the number of model steps at which the next field gets read. For example, if 12 hourly fields are to be interpolated in a setup where 288 steps equals 24 hours.

```

!-----
!      namsao Standalone obs_oper namelist
!-----
!      sao_files    specifies the files containing the model counterpart
!      nn_sao_idx   specifies the time_counter index within the model file
!      nn_sao_freq  specifies number of time steps between read operations
&namsao
  sao_files = "foo.nc" "foo.nc"
  nn_sao_idx = 1 2
  nn_sao_freq = 144
/

```

The above namelist will result in feedback files whose first 12 hours contain the first field of foo.nc and the second 12 hours contain the second field.

Note Missing files can be denoted as "nofile".

A collection of fields taken from a number of files at different indices can be combined at a particular frequency in time to generate a pseudo model evolution. If all that is needed is a single model counterpart at a regular interval then the standard SAO is all that is required. However, just to note, it is possible to extend this approach by comparing multiple forecasts, analyses, persisted analyses and climatologies with the same set of observations. This approach is referred to as *Class 4* since it is the fourth metric defined by the GODAE intercomparison project. This requires multiple runs of the SAO and running an additional utility (not currently in the *NEMO* repository) to combine the feedback files into one class 4 file.

11.5. Observation utilities

For convenience some tools for viewing and processing of observation and feedback files are provided in the *NEMO* repository. These tools include OBSTOOLS which are a collection of FORTRAN programs which are helpful to deal with feedback files. They do such tasks as observation file conversion, printing of file contents, some basic statistical analysis of feedback files. The other main tool is an IDL program called dataplot which uses a graphical interface to visualise observations and feedback files. OBSTOOLS and dataplot are described in more detail below.

11.5.1. Obstools

A series of FORTRAN utilities is provided with *NEMO* called OBSTOOLS. These are helpful in handling observation files and the feedback file output from the observation operator. A brief description of some of the utilities follows

corio2fb

The program corio2fb converts profile observation files from the Coriolis format to the standard feedback format. It is called in the following way:

```
corio2fb.exe outputfile inputfile1 inputfile2 ...
```

enact2fb

The program enact2fb converts profile observation files from the ENACT format to the standard feedback format. It is called in the following way:

```
enact2fb.exe outputfile inputfile1 inputfile2 ...
```

fbcomb

The program fbcomb combines multiple feedback files produced by individual processors in an MPI run of *NEMO* into a single feedback file. It is called in the following way:

```
fbcomb.exe outputfile inputfile1 inputfile2 ...
```

fbmatchup

The program fbmatchup will match observations from two feedback files. It is called in the following way:

```
fbmatchup.exe outputfile inputfile1 varname1 inputfile2 varname2 ...
```

fbprint

The program fbprint will print the contents of a feedback file or files to standard output. Selected information can be output using optional arguments. It is called in the following way:

```
fbprint.exe [options] inputfile

options:
  -b          shorter output
  -q          Select observations based on QC flags
  -Q          Select observations based on QC flags
  -B          Select observations based on QC flags
  -u          unsorted
  -s ID       select station ID
  -t TYPE     select observation type
  -v NUM1-NUM2 select variable range to print by number
              (default all)
  -a NUM1-NUM2 select additional variable range to print by number
              (default all)
  -e NUM1-NUM2 select extra variable range to print by number
              (default all)
  -d          output date range
  -D          print depths
  -z          use zipped files
```

fbssel

The program fbssel will select or subsample observations. It is called in the following way:

```
fbssel.exe <input filename> <output filename>
```

fbstat

The program fbstat will output summary statistics in different global areas into a number of files. It is called in the following way:

```
fbstat.exe [-nmlev] <filenames>
```

fbthin

The program fbthin will thin the data to 1 degree resolution. The code could easily be modified to thin to a different resolution. It is called in the following way:

```
fbthin.exe inputfile outputfile
```

sla2fb

The program sla2fb will convert an AVISO SLA format file to feedback format. It is called in the following way:

```
sla2fb.exe [-s type] outputfile inputfile1 inputfile2 ...

Option:
  -s          Select altimeter data_source
```

vel2fb

The program vel2fb will convert TAO/PIRATA/RAMA currents files to feedback format. It is called in the following way:

```
vel2fb.exe outputfile inputfile1 inputfile2 ...
```

11.5.2. Building the obstools

To build the obstools use in the tools directory use `./maketools -n OBSTOOLS -m [ARCH]`.

11.5.3. Dataplot

An IDL program called dataplot is included which uses a graphical interface to visualise observations and feedback files. Note a similar package has recently developed in python (also called dataplot) which does some of the same things that the IDL dataplot does. Please contact the authors of the this chapter if you are interested in this.

It is possible to zoom in, plot individual profiles and calculate some basic statistics. To plot some data run IDL and then:

To read multiple files into dataplot, for example multiple feedback files from different processors or from different days, the easiest method is to use the spawn command to generate a list of files which can then be passed to dataplot.

figure 11.5 shows the main window which is launched when dataplot starts. This is split into three parts. At the top there is a menu bar which contains a variety of drop down menus. Areas - zooms into prespecified regions; plot - plots the data as a timeseries or a T-S diagram if appropriate; Find - allows data to be searched; Config - sets various configuration options.

The middle part is a plot of the geographical location of the observations. This will plot the observation value, the model background value or observation minus background value depending on the option selected in the radio button at the bottom of the window. The plotting colour range can be changed by clicking on the colour bar. The title of the plot gives some basic information about the date range and depth range shown, the extreme values, and the mean and RMS values. It is possible to zoom in using a drag-box. You may also zoom in or out using the mouse wheel.

The bottom part of the window controls what is visible in the plot above. There are two bars which select the level range plotted (for profile data). The other bars below select the date range shown. The bottom of the figure allows the option to plot the mean, root mean square, standard deviation or mean square values. As mentioned above you can choose to plot the observation value, the model background value or observation minus background value. The next group of radio buttons selects the map projection. This can either be regular longitude latitude grid, or north or south polar stereographic. The next group of radio buttons will plot bad observations, switch to salinity and plot density for profile observations. The rightmost group of buttons will print the plot window as a postscript, save it as png, or exit from dataplot.

If a profile point is clicked with the mouse button a plot of the observation and background values as a function of depth (figure 11.6).

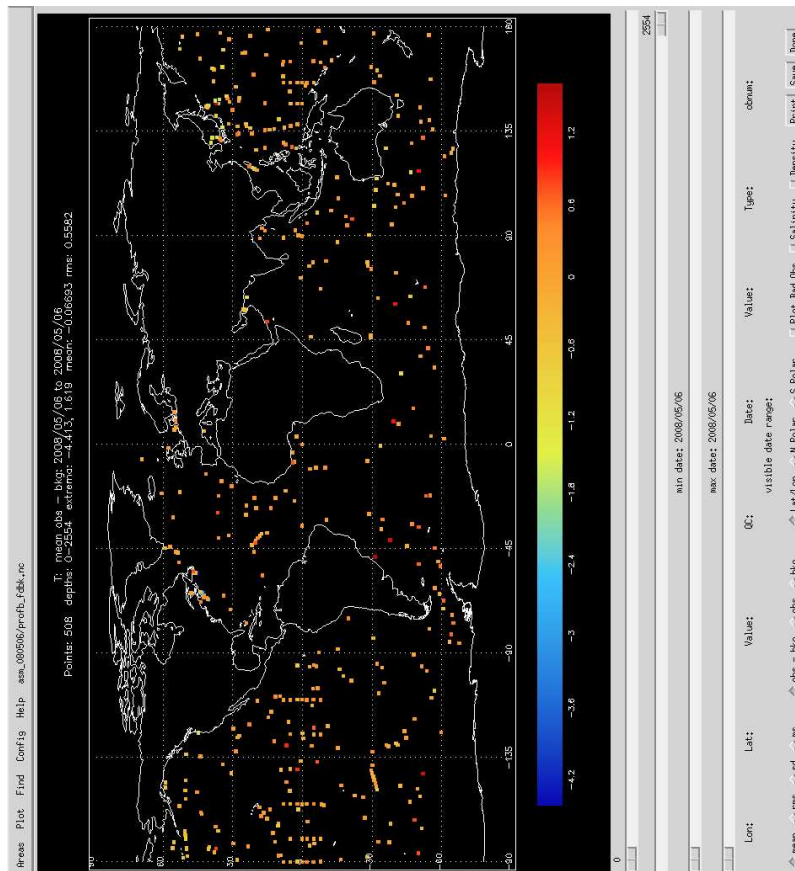


Figure 11.5.: Main window of dataplot

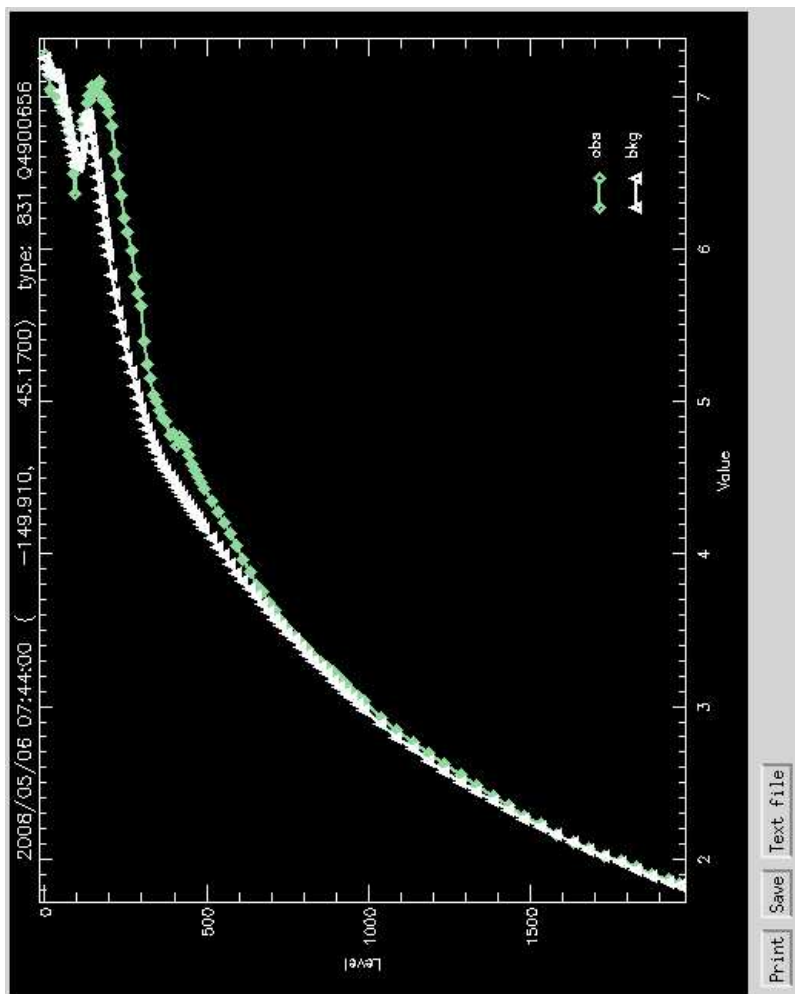


Figure 11.6.: Profile plot from dataplot produced by right clicking on a point in the main window

Apply Assimilation Increments (ASM)

Table of contents

12.1. Direct initialization	179
12.2. Incremental analysis updates	179
12.3. Divergence damping initialisation	179
12.4. Implementation details	180

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

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 separate data assimilation code. 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` (namelist 12.1). There is a brief description of all the namelist options provided. To build the ASM code `key_asminc` must be set.

12.1. 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.

12.2. 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) (Bloom et al., 1996). IAU is a common technique used with 3D assimilation methods such as 3D-Var or OL. IAU is used when `ln_asmiau` is set to true.

With IAU, the model state trajectory 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:

$$x^a(t_i) = M(t_i, t_0)[x^b(t_0)] + F_i \delta \tilde{x}^a$$

where F_i is a weighting function for applying the increments $\delta \tilde{x}^a$ defined such that $\sum_{i=1}^N F_i = 1$. x^b denotes the model initial state and 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 (namelist option `niaufn = 0`) employs constant weights,

$$F_i^{(1)} = \begin{cases} 0 & \text{if } t_i < t_m \\ 1/M & \text{if } t_m < t_i \leq t_n \\ 0 & \text{if } t_i > t_n \end{cases} \quad (12.1)$$

where $M = m - n$. The second function (namelist option `niaufn = 1`) 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:

$$F_i^{(2)} = \begin{cases} 0 & \text{if } t_i < t_m \\ \alpha i & \text{if } t_m \leq t_i \leq t_{M/2} \\ \alpha (M - i + 1) & \text{if } t_{M/2} < t_i \leq t_n \\ 0 & \text{if } t_i > t_n \end{cases} \quad (12.2)$$

where $\alpha^{-1} = \sum_{i=1}^{M/2} 2i$ and M is assumed to be even. The weights described by equation 12.2 provide a smoother transition of the analysis trajectory from one assimilation cycle to the next than that described by equation 12.1.

12.3. Divergence damping initialisation

It is quite challenging for data assimilation systems to provide non-divergent velocity increments. Applying divergent velocity increments will likely cause spurious vertical velocities in the model. This section describes a method to take velocity increments provided to *NEMO* (u_I^0 and v_I^0) and adjust them by the iterative application of a divergence damping operator. The method is also described in Dobricic et al. (2007).

In iteration step n (starting at $n = 1$) new estimates of velocity increments u_I^n and v_I^n are updated by:

$$\begin{cases} u_I^n = u_I^{n-1} + \frac{1}{e_{1u}} \delta_{i+1/2} (A_D \chi_I^{n-1}) \\ v_I^n = v_I^{n-1} + \frac{1}{e_{2v}} \delta_{j+1/2} (A_D \chi_I^{n-1}) \end{cases}, \quad (12.3)$$

where the divergence is defined as

```

!-----
&nam_asminc ! assimilation increments ('key_asminc')
!-----
ln_bkgwri = .false. ! Logical switch for writing out background state
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
ln_salfix = .false. ! Logical switch for ensuring that the sa > salfixmin
salfixmin = -9999 ! Minimum salinity after applying the increments
nn_divdmp = 0 ! Number of iterations of divergence damping operator
/

```

namelist 12.1.: &nam_asminc

$$\chi_I^{n-1} = \frac{1}{e_{1t} e_{2t} e_{3t}} (\delta_i [e_{2u} e_{3u} u_I^{n-1}] + \delta_j [e_{1v} e_{3v} v_I^{n-1}]).$$

By the application of [equation 12.3](#) the divergence is filtered in each iteration, and the vorticity is left unchanged. In the presence of coastal boundaries with zero velocity increments perpendicular to the coast the divergence is strongly damped. This type of the initialisation reduces the vertical velocity magnitude and alleviates the problem of the excessive unphysical vertical mixing in the first steps of the model integration (Talagrand, 1972; Dobricic et al., 2007). Diffusion coefficients are defined as $A_D = \alpha e_{1t} e_{2t}$, where $\alpha = 0.2$. The divergence damping is activated by assigning to `nn_divdmp` in the `&nam_asminc` (namelist 12.1) namelist a value greater than zero. This specifies the number of iterations of the divergence damping. Setting a value of the order of 100 will result in a significant reduction in the vertical velocity induced by the increments.

12.4. Implementation details

Here we show an example `&nam_asminc` (namelist 12.1) namelist and the header of an example assimilation increments file on the ORCA2 grid.

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_ttype = "BOX" ;
}

```

Stochastic Parametrization of EOS (STO)

Table of contents

13.1. Stochastic processes	182
13.2. Implementation details	183

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

As a result of the nonlinearity of the seawater equation of state, unresolved scales represent a major source of uncertainties in the computation of the large-scale horizontal density gradient from the large-scale temperature and salinity fields. Following Brankart (2013), the impact of these uncertainties can be simulated by random processes representing unresolved T/S fluctuations. The Stochastic Parametrization of EOS (STO) module implements this parametrization.

As detailed in Brankart (2013), the stochastic formulation of the equation of state can be written as:

$$\rho = \frac{1}{2} \sum_{i=1}^m \{ \rho[T + \Delta T_i, S + \Delta S_i, p_o(z)] + \rho[T - \Delta T_i, S - \Delta S_i, p_o(z)] \} \quad (13.1)$$

where $p_o(z)$ is the reference pressure depending on the depth and, ΔT_i and ΔS_i ($i=1,m$) is a set of T/S perturbations defined as the scalar product of the respective local T/S gradients with random walks ξ :

$$\Delta T_i = \xi_i \cdot \nabla T \quad \text{and} \quad \Delta S_i = \xi_i \cdot \nabla S \quad (13.2)$$

ξ_i are produced by a first-order autoregressive process (AR-1) with a parametrized decorrelation time scale, and horizontal and vertical standard deviations σ_s . ξ are uncorrelated over the horizontal and fully correlated along the vertical.

13.1. Stochastic processes

There are many existing parameterizations based on autoregressive processes, which are used as a basic source of randomness to transform a deterministic model into a probabilistic model. The generic approach here is to a new STO module, generating processes features with appropriate statistics to simulate these uncertainties in the model (see Brankart et al. (2015) for more details).

In practice, at each model grid point, independent Gaussian autoregressive processes $\xi^{(i)}$, $i = 1, \dots, m$ are first generated using the same basic equation:

$$\xi_{k+1}^{(i)} = a^{(i)} \xi_k^{(i)} + b^{(i)} w^{(i)} + c^{(i)} \quad (13.3)$$

where k is the index of the model timestep and $a^{(i)}$, $b^{(i)}$, $c^{(i)}$ are parameters defining the mean ($\mu^{(i)}$) standard deviation ($\sigma^{(i)}$) and correlation timescale ($\tau^{(i)}$) of each process:

- for order 1 processes, $w^{(i)}$ is a Gaussian white noise, with zero mean and standard deviation equal to 1, and the parameters $a^{(i)}$, $b^{(i)}$, $c^{(i)}$ are given by:

$$\begin{cases} a^{(i)} = \varphi \\ b^{(i)} = \sigma^{(i)} \sqrt{1 - \varphi^2} \\ c^{(i)} = \mu^{(i)} (1 - \varphi) \end{cases} \quad \text{with} \quad \varphi = \exp(-1/\tau^{(i)})$$

- for order $n > 1$ processes, $w^{(i)}$ is an order $n - 1$ autoregressive process, with zero mean, standard deviation equal to $\sigma^{(i)}$; correlation timescale equal to $\tau^{(i)}$; and the parameters $a^{(i)}$, $b^{(i)}$, $c^{(i)}$ are given by:

$$\begin{cases} a^{(i)} = \varphi \\ b^{(i)} = \frac{n-1}{2(4n-3)} \sqrt{1 - \varphi^2} \\ c^{(i)} = \mu^{(i)} (1 - \varphi) \end{cases} \quad \text{with} \quad \varphi = \exp(-1/\tau^{(i)}) \quad (13.4)$$

In this way, higher order processes can be easily generated recursively using the same piece of code implementing equation 13.3, and using successive processes from order 0 to $n - 1$ as $w^{(i)}$. The parameters in equation 13.4 are computed so that this recursive application of equation 13.3 leads to processes with the required standard deviation and correlation timescale, with the additional condition that the $n - 1$ first derivatives of the autocorrelation function are equal to zero at $t = 0$, so that the resulting processes become smoother and smoother as n increases.

Overall, this method provides quite a simple and generic way of generating a wide class of stochastic processes. However, this also means that new model parameters are needed to specify each of these stochastic processes. As in any parameterization, the main issue is to tune the parameters using either first principles, model simulations, or real-world observations. The parameters are set by default as described in Brankart (2013), which has been shown in the paper to give good results for a global low resolution (2ř) NEMO configuration. where this parametrization produces a major effect on the average large-scale circulation, especially in regions of intense mesoscale activity. The set of parameters will need further investigation to find appropriate values for any other configuration or resolution of the model.

```

!-----
&namsto      ! Stochastic parametrization of EOS                (default: OFF)
!-----
ln_sto_eos   = .false.    ! stochastic equation of state
nn_sto_eos   = 1          ! number of independent random walks
rn_eos_stdxy = 1.4        ! random walk horz. standard deviation (in grid points)
rn_eos_stdz  = 0.7        ! random walk vert. standard deviation (in grid points)
rn_eos_tcor  = 1440.      ! random walk time correlation (in timesteps)
nn_eos_ord   = 1          ! order of autoregressive processes
nn_eos_flt   = 0          ! passes of Laplacian filter
rn_eos_lim   = 2.0        ! limitation factor (default = 3.0)
ln_rststo    = .false.    ! start from mean parameter (F) or from restart file (T)
ln_rstseed   = .true.     ! read seed of RNG from restart file
cn_storst_in = "restart_sto" ! suffix of stochastic parameter restart file (input)
cn_storst_out = "restart_sto" ! suffix of stochastic parameter restart file (output)
/

```

namelist 13.1.: &namsto

13.2. Implementation details

The code implementing stochastic parametrization is located in the `src/OCE/STO` directory. It contains three modules :

`stopar.F90` : define the Stochastic parameters and their time evolution

`stornrg.F90` : random number generator based on and including the 64-bit KISS (Keep It Simple Stupid) random number generator distributed by George Marsaglia

`stopts.F90` : stochastic parametrization associated with the non-linearity of the equation of seawater, implementing equation 13.2 so as specifics in the equation of state implementing equation 13.1.

The `stopar.F90` module includes three public routines called in the model:

(`sto_par`) is a direct implementation of equation 13.3, applied at each model grid point (in 2D or 3D), and called at each model time step (k) to update every autoregressive process ($i = 1, \dots, m$). This routine also includes a filtering operator, applied to $w^{(i)}$, to introduce a spatial correlation between the stochastic processes.

(`sto_par_init`) is the initialization routine computing the values $a^{(i)}, b^{(i)}, c^{(i)}$ for each autoregressive process, as a function of the statistical properties required by the model user (mean, standard deviation, time correlation, order of the process, ...). This routine also includes the initialization (seeding) of the random number generator.

(`sto_rst_write`) writes a restart file (which suffix name is given by `cn_storst_out` namelist parameter) containing the current value of all autoregressive processes to allow creating the file needed for a restart. This restart file also contains the current state of the random number generator. When `ln_rststo` is set to `.true.`, the restart file (which suffix name is given by `cn_storst_in` namelist parameter) is read by the initialization routine (`sto_par_init`). The simulation will continue exactly as if it was not interrupted only when `ln_rstseed` is set to `.true.`, *i.e.* when the state of the random number generator is read in the restart file.

The implementation includes the basics for a few possible stochastic parametrizations including equation of state, lateral diffusion, horizontal pressure gradient, ice strength, trend, tracers dynamics. As for this release, only the stochastic parametrization of equation of state is fully available and tested.

Options and parameters

The `ln_sto_eos` namelist variable activates stochastic parametrization of equation of state. By default it set to `.false.` and not active. The set of parameters is available in `&namsto` (namelist 13.1) namelist (only the subset for equation of state stochastic parametrization is listed below):

The variables of stochastic parametrization itself (based on the global 2ř experiments as in Brankart (2013)) are:

nn_sto_eos : number of independent random walks

rn_eos_stdxy : random walk horizontal standard deviation (in grid points)

rn_eos_stdz : random walk vertical standard deviation (in grid points)

rn_eos_tcor : random walk time correlation (in timesteps)

nn_eos_ord : order of autoregressive processes

nn_eos_flt : passes of Laplacian filter

rn_eos_lim : limitation factor (default = 3.0)

The first four parameters define the stochastic part of equation of state.

Table of contents

14.1. Representation of unresolved straits	185
14.1.1. Hand made geometry changes	185
14.2. Closed seas (<i>closea.F90</i>)	185
14.3. Sub-domain functionality	187
14.3.1. Simple subsetting of input files via NetCDF attributes	187
14.4. Accuracy and reproducibility (<i>lib_fortran.F90</i>)	188
14.4.1. Issues with intrinsic SIGN function (key_nosignedzero)	188
14.4.2. MPP reproducibility	188
14.4.3. MPP scalability	188
14.5. Model optimisation, control print and benchmark	189
14.5.1. Vector optimisation	189
14.5.2. Control print	189

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

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 two methods that can be used in *NEMO* to handle such improperly resolved straits. The methods consist of opening the strait while ensuring that the mass exchanges through the strait are not too large by either artificially reducing the cross-sectional area of the strait grid-cells or, locally increasing the lateral friction.

14.1.1. Hand made geometry changes

The first method involves reducing the scale factor in the cross-strait direction to a value in better agreement with the true mean width of the strait (figure 14.1). 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 which would require a reduction of the model time step. Thus to instigate a local change in the width of a Strait requires two steps:

- Add `e1e2u` and `e1e2v` arrays to the `cn_domcfg` file. These 2D arrays should contain the products of the unaltered values of: `e1u * e2u` and `e1u * e2v` respectively. That is the original surface areas of *u*- and *v*- cells respectively. These areas are usually defined by the corresponding product within the *NEMO* code but the presence of `e1e2u` and `e1e2v` in the `cn_domcfg` file will suppress this calculation and use the supplied fields instead. If the model domain is provided by user-supplied code in `usrdef_hgr.F90`, then this routine should also return `e1e2u` and `e1e2v` and set the integer return argument `ie1e2u_v` to a non-zero value. Values other than 0 for this argument will suppress the calculation of the areas.
- Change values of `e2u` or `e1v` (either in the `cn_domcfg` file or via code in `usrdef_hgr.F90`), wherever a Strait reduction is required. The choice of whether to alter `e2u` or `e1v` depends, respectively, on whether the Strait in question is North-South orientated (*e.g.* Gibraltar) or East-West orientated (*e.g.* Lombok).

The second method is to increase the viscous boundary layer thickness by a local increase of the `fmask` value at the coast. This method can also be effective in wider passages. The concept is illustrated in the second part of figure 14.1 and changes to specific locations can be coded in `usrdef_fmash.F90`. The `usr_def_fmash` routine is always called after `fmask` has been defined according to the choice of lateral boundary condition as discussed in section 7.1. The default version of `usrdef_fmash.F90` contains settings specific to ORCA2 and ORCA1 configurations. These are meant as examples only; it is up to the user to verify settings and provide alternatives for their own configurations. The default `usr_def_fmash` makes no changes to `fmask` for any other configuration.

14.2. Closed seas (`closea.F90`)

Some configurations include inland seas and lakes as ocean points. This is particularly the case for configurations that are coupled to an atmosphere model where one might want to include inland seas and lakes as ocean model points in order to provide a better bottom boundary condition for the atmosphere. However there is no route for freshwater to run off from the lakes to the ocean and this can lead to large drifts in the sea surface height over the lakes. The `closea` module provides options to either fill in closed seas and lakes at run time, or to set the net surface freshwater flux for each lake to zero and put the residual flux into the ocean.

Prior to *NEMO* 4 the locations of inland seas and lakes was set via hardcoded indices for various ORCA configurations. From *NEMO* 4 onwards the inland seas and lakes are defined using mask fields in the domain configuration file. The options are as follows.

1. **No “`closea_mask`” field is included in domain configuration file.** In this case the `closea` module does nothing.
2. **A field called `closea_mask` is included in the domain configuration file and `ln_closea=.false.` in namelist `namcfg`.** In this case the inland seas defined by the `closea_mask` field are filled in (turned to land points) at run time. That is every point in `closea_mask` that is nonzero is set to be a land point.
3. **A field called `closea_mask` is included in the domain configuration file and `ln_closea=.true.` in namelist `namcfg`.** Each inland sea or group of inland seas is set to a positive integer value in the `closea_mask` field (see

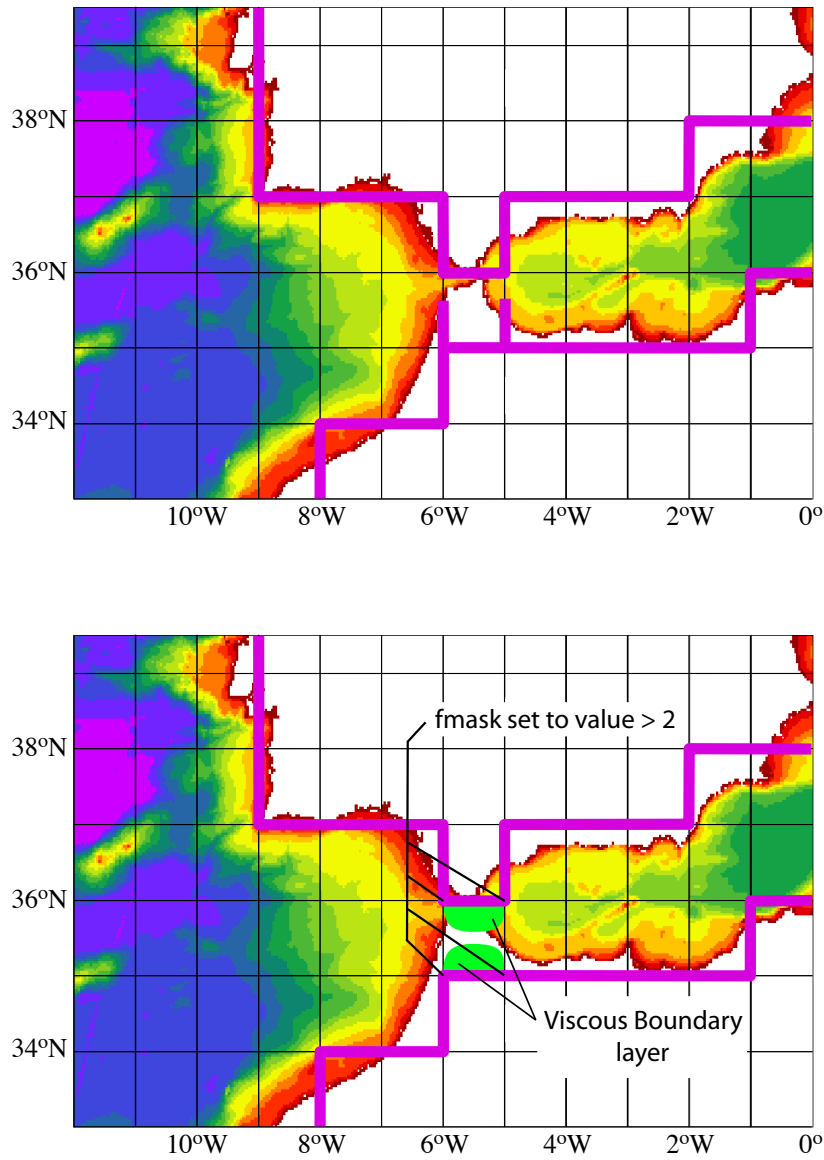


Figure 14.1.: Example of the Gibraltar strait defined in a $1^\circ \times 1^\circ$ mesh. *Top*: using partially open cells. The meridional scale factor at v -point is reduced on both sides of the strait to account for the real width of the strait (about 20 km). Note that the scale factors of the strait T -point remains unchanged. *Bottom*: using viscous boundary layers. The four f_{mask} parameters along the strait coastlines are set to a value larger than 4, *i.e.* "strong" no-slip case (see figure 7.2) creating a large viscous boundary layer that allows a reduced transport through the strait.

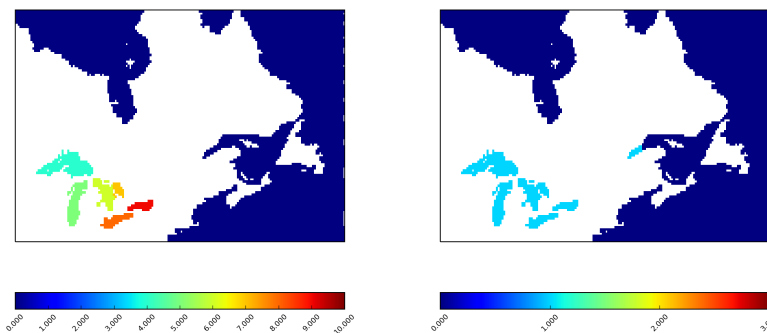


Figure 14.2.: Example of mask fields for the `closea.F90` module. *Left*: a `closea_mask` field; *Right*: a `closea_mask_rnf` field. In this example, if `ln_closea` is set to `.true.`, the mean freshwater flux over each of the American Great Lakes will be set to zero, and the total residual for all the lakes, if negative, will be put into the St Lawrence Seaway in the area shown.

figure 14.2 for an example). The net surface flux over each inland sea or group of inland seas is set to zero each timestep and the residual flux is distributed over the global ocean (ie. all ocean points where `closea_mask` is zero).

4. **Fields called `closea_mask` and `closea_mask_rnf` are included in the domain configuration file and `ln_closea=.true.` in namelist `namcfg`.** This option works as for option 3, except that if the net surface flux over an inland sea is negative (net precipitation) it is put into the ocean at specified runoff points. A net positive surface flux (net evaporation) is still spread over the global ocean. The mapping from inland seas to runoff points is defined by the `closea_mask_rnf` field. Each mapping is defined by a positive integer value for the inland sea(s) and the corresponding runoff points. An example is given in figure 14.2. If no mapping is provided for a particular inland sea then the residual is spread over the global ocean.
5. **Fields called `closea_mask` and `closea_mask_emp` are included in the domain configuration file and `ln_closea=.true.` in namelist `namcfg`.** This option works the same as option 4 except that the nonzero net surface flux is sent to the ocean at the specified runoff points regardless of whether it is positive or negative. The mapping from inland seas to runoff points in this case is defined by the `closea_mask_emp` field.

There is a python routine to create the `closea_mask` fields and append them to the domain configuration file in the `utils/tools/DOMAINcfg` directory.

14.3. Sub-domain functionality

14.3.1. Simple subsetting of input files via NetCDF attributes

The extended grids for use with the under-shelf ice cavities will result in redundant rows around Antarctica if the ice cavities are not active. A simple mechanism for subsetting input files associated with the extended domains has been implemented to avoid the need to maintain different sets of input fields for use with or without active ice cavities. This subsetting operates for the j-direction only and works by optionally looking for and using a global file attribute (named: `open_ocean_jstart`) to determine the starting j-row for input. The use of this option is best explained with an example:

Consider an ORCA1 configuration using the extended grid domain configuration file: `eORCA1_domcfg.nc.nc`. This file defines a horizontal domain of 362x332. The first row with open ocean wet points in the non-isf bathymetry for this set is row 42 (FORTRAN indexing) then the formally correct setting for `open_ocean_jstart` is 41. Using this value as the first row to be read will result in a 362x292 domain which is the same size as the original ORCA1 domain. Thus the extended domain configuration file can be used with all the original input files for ORCA1 if the ice cavities are not active (`ln_isfcav = .false.`). Full instructions for achieving this are:

- Add the new attribute to any input files requiring a j-row offset, i.e:

```
ncatted -a open_ocean_jstart,global,a,d,41 eORCA1_domcfg.nc
```

- Add the logical switch `ln_use_jattr` to `&namcfg` (namelist 15.1) in the configuration namelist (if it is not already there) and set `.true.`

Note that with this option, the j-size of the global domain is (extended j-size minus `open_ocean_jstart` + 1) and this must match the `jpjglo` value for the configuration. This means an alternative version of `eORCA1_domcfg.nc.nc` must be created for when `ln_use_jattr` is active. The `ncap2` tool provides a convenient way of achieving this:

```
ncap2 -s 'jpjglo=292' eORCA1_domcfg.nc nORCA1_domcfg.nc
```

The domain configuration file is unique in this respect since it also contains the value of `jpjglo` that is read and used by the model. Any other global, 2D and 3D, netcdf, input field can be prepared for use in a reduced domain by adding the `open_ocean_jstart` attribute to the file's global attributes. In particular this is true for any field that is read by *NEMO* using the following optional argument to the appropriate call to `iom_get`.

```
lrowattr=ln_use_jattr
```

Currently, only the domain configuration variables make use of this optional argument so this facility is of little practical use except for tests where no other external input files are needed or you wish to use an extended domain configuration with inputs from earlier, non-extended configurations. Alternatively, it should be possible to exclude empty rows for extended domain, forced ocean runs using interpolation on the fly, by adding the optional argument to `iom_get` calls for the weights and initial conditions. Experimenting with this remains an exercise for the user.

14.4. Accuracy and reproducibility (*lib_fortran.F90*)

14.4.1. Issues with intrinsic 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 90, 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 intrinsic 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 overwriting of a intrinsic function can present performance issues with some computers/compilers.

14.4.2. 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 [He and Ding \(2001\)](#) 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.

$$\begin{aligned} sum_1 &= a_1 + a_2 \\ error_1 &= a_2 + (a_1 - sum_1) \\ sum_2 &= sum_1 + a_3 + error_1 \\ error_2 &= a_3 + error_1 + (sum_1 - sum_2) \\ b &= sum_2 \end{aligned}$$

An example of this feature can be found in *lib_fortran.F90* module. It is systematicallt used in *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). The self-compensated summation method should be used in all summation in i- and/or j-direction. See *closea.F90* module for an example. Note also that this implementation may be sensitive to the optimization level.

14.4.3. MPP scalability

The default method of communicating values across the north-fold in distributed memory applications (**key_mpp_mpi**) uses a MPI_ALLGATHER function to exchange values from each processing region in the northern row with every other processing region in the northern row. This enables a global width array containing the top 4 rows to be collated on every northern row processor and then folded with a simple algorithm. Although conceptually simple, this "All to All" communication will hamper performance scalability for large numbers of northern row processors. From version 3.4 onwards an alternative method is available which only performs direct "Peer to Peer" communications between each processor and its immediate "neighbours" across the fold line. This is achieved by using the default MPI_ALLGATHER method during initialisation to help identify the "active" neighbours. Stored lists of these neighbours are then used in all subsequent north-fold exchanges to restrict exchanges to those between associated regions. The collated global width array for each region is thus only partially filled but is guaranteed to be set at all the locations actually required by each individual for the fold operation. This alternative method should give identical results to the default ALLGATHER method and is recommended for large values of *j_pni*. The new method is activated by setting *ln_nnogather* to be true (*&nammpp* (namelist 7.2)). The reproducibility of results using the two methods should be confirmed for each new, non-reference configuration.

```

!-----
&namctl          ! Control prints                                     (default: OFF)
!-----
ln_ctl = .FALSE.      ! Toggle all report printing on/off (T/F); Ignored if sn_cfctl%l_config is T
sn_cfctl%l_config = .TRUE. ! IF .true. then control which reports are written with the following
sn_cfctl%l_runstat = .FALSE. ! switches and which areas produce reports with the proc integer settings.
sn_cfctl%l_trcstat = .FALSE. ! The default settings for the proc integers should ensure
sn_cfctl%l_oeout = .FALSE. ! that all areas report.
sn_cfctl%l_layout = .FALSE. !
sn_cfctl%l_mppout = .FALSE. !
sn_cfctl%l_mpptop = .FALSE. !
sn_cfctl%l_procsmin = 0 ! Minimum area number for reporting [default:0]
sn_cfctl%l_procsmax = 1000000 ! Maximum area number for reporting [default:1000000]
sn_cfctl%l_procsincr = 1 ! Increment for optional subsetting of areas [default:1]
sn_cfctl%l_ptimincr = 1 ! Timestep increment for writing time step progress info
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
ln_timing = .false. ! timing by routine write out in timing.output file
ln_diacfl = .false. ! CFL diagnostics write out in cfl_diagnostics.ascii
/

```

namelist 14.1.: &namctl

14.5. Model optimisation, control print and benchmark

Options are defined through the &namctl (namelist 14.1) namelist variables.

14.5.1. 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.

14.5.2. Control print

The `ln_ctl` switch was originally used as a debugging option in two modes:

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.

However, in recent versions it has also been used to force all processors to assume the reporting role. Thus when `ln_ctl` is true all processors produce their own versions of files such as: `ocean.output`, `layout.dat`, etc. All such files, beyond the the normal reporting processor (`narea == 1`), are named with a `_XXXX` extension to their name, where `XXXX` is a 4-digit area number (with leading zeros, if required). Other reporting files such as `run.stat` (and its netCDF counterpart: `run.stat.nc`) and `tracer.stat` contain global information and are only ever produced by the reporting master (`narea == 1`). For version 4.0 a start has been made to return `ln_ctl` to its original function by introducing a new control structure which allows finer control over which files are produced. This feature is still evolving but it does already allow the user to: select individually the production of `run.stat` and `tracer.stat` files and to toggle the production of other files on processors other than the reporting master. These other reporters can be a simple subset of processors as defined by a minimum, maximum and incremental processor number.

Note, that production of the `run.stat` and `tracer.stat` files require global communications. For `run.stat`, these are global min and max operations to find metrics such as the gloabl maximum velocity. For `tracer.stat` these are global sums of tracer fields. To improve model performance these operations are disabled by default and, where necessary, any use of the global values have been replaced with local calculations. For example, checks on the CFL criterion are now done on the local domain and only reported if a breach is detected.

Experienced users may wish to still monitor this information as a check on model progress. If so, the best compromise will be to activate the files with:

```

sn_cfctl%l_config = .TRUE.
sn_cfctl%l_runstat = .TRUE.
sn_cfctl%l_trcstat = .TRUE.

```

and to use the new time increment setting to ensure the values are collected and reported at a suitably long interval. For example:

```
sn_cfctl%ptimincr = 25
```

will carry out the global communications and write the information every 25 timesteps. This increment also applies to the `time.step` file which is otherwise updated every timestep.

Table of contents

15.1. Introduction	192
15.2. CID: 1D Water column model (key_c1d)	192
15.3. ORCA family: global ocean with tripolar grid	192
15.3.1. ORCA tripolar grid	193
15.3.2. ORCA pre-defined resolution	193
15.4. GYRE family: double gyre basin	195
15.5. AMM: atlantic margin configuration	196

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

```

!-----
&namcfg      !  parameters of the configuration                                (default: use namusr_def in namelist_cfg)
!-----
ln_read_cfg = .false.  !  (=T) read the domain configuration file
!                   !  (=F) user defined configuration                    (F => create/check namusr_def)
cn_domcfg = "domain_cfg" ! domain configuration filename
!
ln_closea   = .false.  !  T => keep closed seas (defined by closea_mask field) in the
!                   !  domain and apply special treatment of freshwater fluxes.
!                   !  F => suppress closed seas (defined by closea_mask field)
!                   !  from the bathymetry at runtime.
!                   !  If closea_mask field doesn't exist in the domain_cfg file
!                   !  then this logical does nothing.
ln_write_cfg = .false. !  (=T) create the domain configuration file
cn_domcfg_out = "domain_cfg_out" ! newly created domain configuration filename
!
ln_use_jattr = .false. !  use (T) the file attribute: open_ocean_jstart, if present
!                   !  in netcdf input files, as the start j-row for reading
/

```

namelist 15.1.: &namcfg

15.1. Introduction

The purpose of this part of the manual is to introduce the *NEMO* reference configurations. 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 reference configurations also provide a sense for some of the options available in the code, though by no means are all options exercised in the reference configurations. Configuration is defined manually through the &namcfg (namelist 15.1) namelist variables.

15.2. C1D: 1D Water column model (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 position of the 1D water column in the grid (see `./cfgs/SHARED/namelist_ref`). 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 configuration of the smallest possible domain: a 3x3 domain with 75 vertical levels.

The 1D model has some specifics. First, all the horizontal derivatives are assumed to be zero, and second, the two components of the velocity are moved on a *T*-point. Therefore, defining **key_c1d** changes some things in the code behaviour:

1. a simplified `stp` routine is used (`stp_c1d`, see `step_c1d.F90` module) in which both lateral tendency terms and lateral physics are not called;
2. the vertical velocity is zero (so far, no attempt at introducing a Ekman pumping velocity has been made);
3. a simplified treatment of the Coriolis term is performed as *U*- and *V*-points are the same (see `dyncor_c1d.F90`).

All the relevant `_c1d` modules can be found in the `src/OCE/C1D` directory of the *NEMO* distribution.

15.3. ORCA family: global ocean with tripolar grid

The ORCA family is a series of global ocean configurations that are run together with the SI3 model (ORCA-ICE) and possibly with PISCES biogeochemical model (ORCA-ICE-PISCES). An appropriate namelist is available in `./cfgs/ORCA2_ICE_PISCES/EXPREF/namelist_cfg` for ORCA2. The domain of ORCA2 configuration is defined in `ORCA_R2_zps_domcfg.nc` file, this file is available in tar file on the *NEMO* community zenodo platform: <https://doi.org/10.5281/zenodo.2640723>

In this `namelist_cfg` the name of domain input file is set in &namcfg (namelist 15.1) block of namelist.

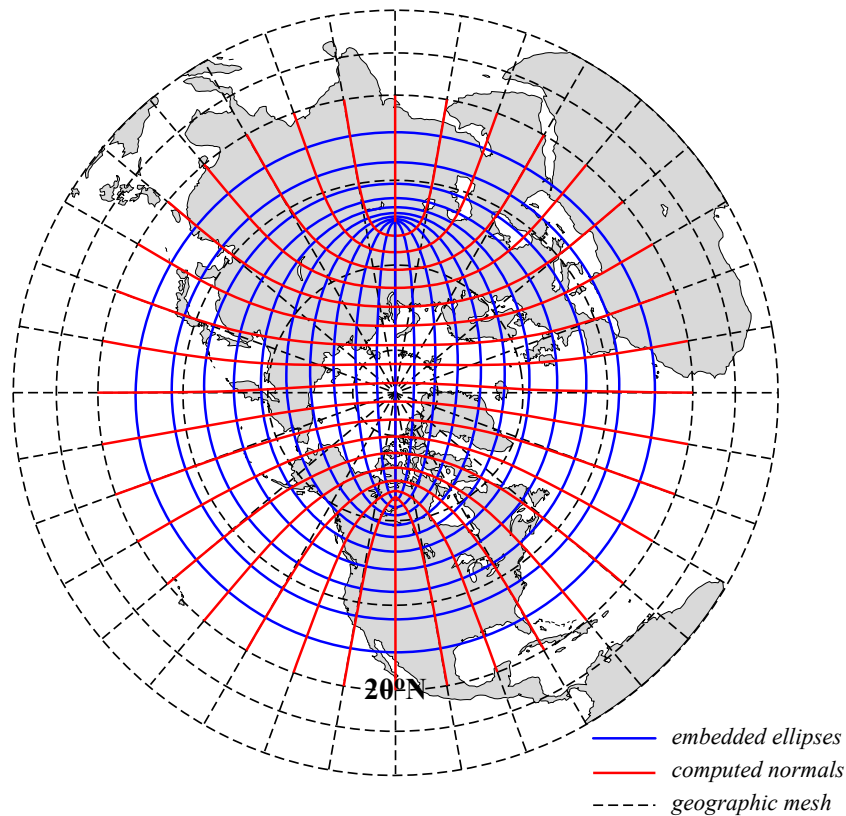


Figure 15.1.: ORCA mesh conception. The departure from an isotropic Mercator grid start poleward of 20°N. The two "north pole" are the foci of a series of embedded ellipses (blue curves) which are determined analytically and form the i-lines of the ORCA mesh (pseudo latitudes). Then, following [Madec and Imbard \(1996\)](#), the normal to the series of ellipses (red curves) is computed which provides the j-lines of the mesh (pseudo longitudes).

15.3.1. ORCA tripolar grid

The ORCA grid is a tripolar grid based on the semi-analytical method of [Madec and Imbard \(1996\)](#). 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 ([figure 15.1](#)). 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 [figure 15.1](#) and [figure 15.2](#) 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 the *domain_cfg* domain configuration file, which sets the grid size and configuration name parameters. The *NEMO* System Team provides only ORCA2 domain input file "ORCA_R2_zps_domcfg.nc" file ([table 15.1](#)).

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 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.

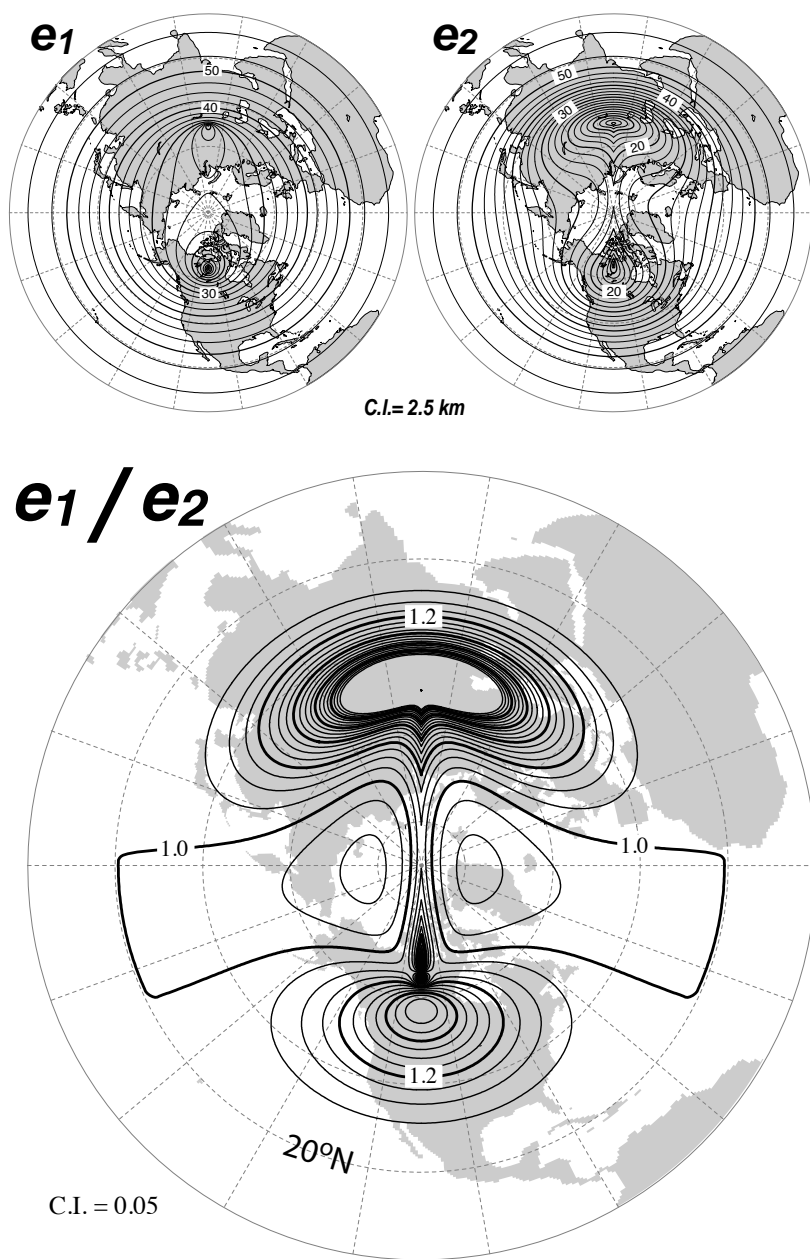


Figure 15.2.: *Top*: Horizontal scale factors (e_1 , e_2) and *Bottom*: ratio of anisotropy (e_1/e_2) for ORCA 0.5° mesh. South of 20°N a Mercator grid is used ($e_1 = e_2$) so that the anisotropy ratio is 1. Poleward of 20°N , the two "north pole" introduce a weak anisotropy over the ocean areas (< 1.2) except in vicinity of Victoria Island (Canadian Arctic Archipelago).

Horizontal Grid	ORCA_index	jpiglo	jpjglo
2 °	2	182	149
1 °	1	362	292
0.5 °	05	722	511
0.25°	025	1442	1021

Table 15.1.: Domain size of ORCA family configurations. The flag for configurations of ORCA family need to be set in *domain_cfg* file.

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 [table 15.1](#)).

Only the ORCA_R2 is provided with all its input files in the *NEMO* distribution.

This version of ORCA_R2 has 31 levels in the vertical, with the highest resolution (10m) in the upper 150m (see [table 15.1](#) and [figure 3.2](#)). The bottom topography and the coastlines are derived from the global atlas of Smith and Sandwell (1997). The default forcing uses the boundary forcing from [Large and Yeager \(2004\)](#) (see [subsection 6.4.1](#)), which was developed for the purpose of running global coupled ocean-ice simulations without an interactive atmosphere. This [Large and Yeager \(2004\)](#) dataset is available through the [GFDL web site](#). The "normal year" of [Large and Yeager \(2004\)](#) has been chosen of the *NEMO* distribution since release v3.3.

ORCA_R2 pre-defined configuration can also be run with multiply online nested zooms (*i.e.* with AGRIF, `key_agrif` defined). This is available as the AGRIF_DEMO configuration that can be found in the `./cfgs/AGRIF_DEMO/` directory.

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

The GYRE configuration ([Lévy et al., 2010](#)) has been built to simulate the seasonal cycle of a double-gyre box model. It consists in an idealized domain similar to that used in the studies of [Drijfhout \(1994\)](#) and [Hazeleger and Drijfhout \(1998, 1999, 2000b,a\)](#), 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 GYRE configuration run together with the PISCES biogeochemical model (GYRE-PISCES). The domain geometry is a closed rectangular basin on the β -plane centred at $\sim 30^\circ\text{N}$ and rotated by 45° , 3180 km long, 2120 km wide and 4 km deep ([figure 14.1](#)). The domain is bounded by vertical walls and by a flat 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 fluxes. The applied forcings vary seasonally in a sinusoidal manner between winter and summer extrema ([Lévy et al., 2010](#)). 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 flux takes the form of a restoring toward a zonal apparent air temperature profile. A portion of the net heat flux which comes from the solar radiation is allowed to penetrate within the water column. The fresh water flux is also prescribed and varies zonally. It is determined such as, at each time step, the basin-integrated flux 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 like an analytical configuration. Through `ln_read_cfg=.false.` in `&namcfg` ([namelist 15.1](#)) namelist defined in the reference configuration `./cfgs/GYRE_PISCES/EXPREF/namelist_cfg` analytical definition of grid in GYRE is done in `usrdef_hrg`, `usrdef_zgr` routines. Its horizontal resolution (and thus the size of the domain) is determined by setting `nn_GYRE` in `&namusr_def` (??):

```
jpiglo = 30× nn_GYRE + 2
```

```
jpjglo = 20× nn_GYRE + 2
```

Obviously, the namelist parameters have to be adjusted to the chosen resolution, see the Configurations pages on the *NEMO* web site (*NEMO* Configurations). In the vertical, GYRE uses the default 30 ocean levels (`jpk = 31`) ([figure 3.2](#)).

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. Benchmark is activate via `ln_bench=.true.` in `&namusr_def` (??) in namelist `./cfgs/GYRE_PISCES/EXPREF/namelist_cfg`.

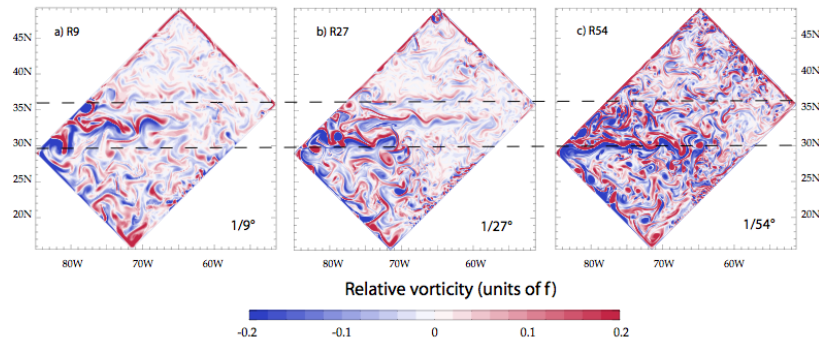


Figure 15.3.: Snapshot of relative vorticity at the surface of the model domain in GYRE R9, R27 and R54. From Lévy et al. (2010).

15.5. AMM: atlantic margin configuration

The AMM, Atlantic Margins Model, is a regional model covering the Northwest European Shelf domain on a regular lat-lon grid at approximately 12km horizontal resolution. The appropriate `&namecfg` namelist is available in `./cfgs/AMM12/EXPREF/namelist_cfg`. It is used to build the correct dimensions of the AMM domain.

This configuration tests several features of *NEMO* functionality specific to the shelf seas. In particular, the AMM uses *s*-coordinates in the vertical rather than *z*-coordinates and is forced with tidal lateral boundary conditions using a Flather boundary condition from the `BDY` module. Also specific to the AMM configuration is the use of the GLS turbulence scheme (`ln_zdfgls=.true.`).

In addition to the tidal boundary condition the model may also take open boundary conditions from a North Atlantic model. Boundaries may be completely omitted by setting `ln_bdy` to false. Sample surface fluxes, river forcing and a sample initial restart file are included to test a realistic model run. The Baltic boundary is included within the river input file and is specified as a river source. Unlike ordinary river points the Baltic inputs also include salinity and temperature data.



Curvilinear s -Coordinate Equations

Table of contents

A.1. Chain rule for s -coordinates	198
A.2. Continuity equation in s -coordinates	198
A.3. Momentum equation in s -coordinate	199
A.4. Tracer equation	202

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

A.1. Chain rule for s -coordinates

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 [subsection 1.3.2](#) 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:

$$\sigma_1 = \frac{1}{e_1} \left. \frac{\partial z}{\partial i} \right|_s \quad \text{and} \quad \sigma_2 = \frac{1}{e_2} \left. \frac{\partial z}{\partial j} \right|_s. \quad (\text{A.1})$$

The model fields (e.g. pressure p) can be viewed as functions of (i, j, z, t) (e.g. $p(i, j, z, t)$) or as functions of (i, j, s, t) (e.g. $p(i, j, s, t)$). The symbol \bullet will be used to represent any one of these fields. Any ‘‘infinitesimal’’ change in \bullet can be written in two forms:

$$\begin{aligned} \delta \bullet &= \delta i \left. \frac{\partial \bullet}{\partial i} \right|_{j,s,t} + \delta j \left. \frac{\partial \bullet}{\partial j} \right|_{i,s,t} + \delta s \left. \frac{\partial \bullet}{\partial s} \right|_{i,j,t} + \delta t \left. \frac{\partial \bullet}{\partial t} \right|_{i,j,s}, \\ \delta \bullet &= \delta i \left. \frac{\partial \bullet}{\partial i} \right|_{j,z,t} + \delta j \left. \frac{\partial \bullet}{\partial j} \right|_{i,z,t} + \delta z \left. \frac{\partial \bullet}{\partial z} \right|_{i,j,t} + \delta t \left. \frac{\partial \bullet}{\partial t} \right|_{i,j,z}. \end{aligned} \quad (\text{A.2})$$

Using the first form and considering a change δi with j, z and t held constant, shows that

$$\left. \frac{\partial \bullet}{\partial i} \right|_{j,z,t} = \left. \frac{\partial \bullet}{\partial i} \right|_{j,s,t} + \left. \frac{\partial s}{\partial i} \right|_{j,z,t} \left. \frac{\partial \bullet}{\partial s} \right|_{i,j,t}. \quad (\text{A.3})$$

The term $\partial s / \partial i|_{j,z,t}$ can be related to the slope of constant s surfaces, ([equation A.1](#)), by applying the second of ([equation A.2](#)) with \bullet set to s and j, t held constant

$$\delta s|_{j,t} = \delta i \left. \frac{\partial s}{\partial i} \right|_{j,z,t} + \delta z \left. \frac{\partial s}{\partial z} \right|_{i,j,t}. \quad (\text{A.4})$$

Choosing to look at a direction in the (i, z) plane in which $\delta s = 0$ and using ([equation A.1](#)) we obtain

$$\left. \frac{\partial s}{\partial i} \right|_{j,z,t} = - \left. \frac{\partial z}{\partial i} \right|_{j,s,t} \left. \frac{\partial s}{\partial z} \right|_{i,j,t} = - \frac{e_1}{e_3} \sigma_1. \quad (\text{A.5})$$

Another identity, similar in form to ([equation A.5](#)), can be derived by choosing \bullet to be s and using the second form of ([equation A.2](#)) to consider changes in which i, j and s are constant. This shows that

$$w_s = \left. \frac{\partial z}{\partial t} \right|_{i,j,s} = - \left. \frac{\partial z}{\partial s} \right|_{i,j,t} \left. \frac{\partial s}{\partial t} \right|_{i,j,z} = -e_3 \left. \frac{\partial s}{\partial t} \right|_{i,j,z}. \quad (\text{A.6})$$

In what follows, for brevity, indication of the constancy of the i, j and t indices is usually omitted. Using the arguments outlined above one can show that the chain rules needed to establish the model equations in the curvilinear s -coordinate system are:

$$\begin{aligned} \left. \frac{\partial \bullet}{\partial t} \right|_z &= \left. \frac{\partial \bullet}{\partial t} \right|_s + \frac{\partial \bullet}{\partial s} \frac{\partial s}{\partial t}, \\ \left. \frac{\partial \bullet}{\partial i} \right|_z &= \left. \frac{\partial \bullet}{\partial i} \right|_s + \frac{\partial \bullet}{\partial s} \frac{\partial s}{\partial i} = \left. \frac{\partial \bullet}{\partial i} \right|_s - \frac{e_1}{e_3} \sigma_1 \frac{\partial \bullet}{\partial s}, \\ \left. \frac{\partial \bullet}{\partial j} \right|_z &= \left. \frac{\partial \bullet}{\partial j} \right|_s + \frac{\partial \bullet}{\partial s} \frac{\partial s}{\partial j} = \left. \frac{\partial \bullet}{\partial j} \right|_s - \frac{e_2}{e_3} \sigma_2 \frac{\partial \bullet}{\partial s}, \\ \frac{\partial \bullet}{\partial z} &= \frac{1}{e_3} \frac{\partial \bullet}{\partial s}. \end{aligned} \quad (\text{A.7})$$

A.2. Continuity equation in s -coordinates

Using ([equation A.3](#)) 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:

$$\begin{aligned}
 \nabla \cdot \mathbf{U} &= \frac{1}{e_1 e_2} \left[\frac{\partial(e_2 u)}{\partial i} \Big|_z + \frac{\partial(e_1 v)}{\partial j} \Big|_z \right] + \frac{\partial w}{\partial z} \\
 &= \frac{1}{e_1 e_2} \left[\frac{\partial(e_2 u)}{\partial i} \Big|_s - \frac{e_1}{e_3} \sigma_1 \frac{\partial(e_2 u)}{\partial s} + \frac{\partial(e_1 v)}{\partial j} \Big|_s - \frac{e_2}{e_3} \sigma_2 \frac{\partial(e_1 v)}{\partial s} \right] + \frac{\partial w}{\partial s} \frac{\partial s}{\partial z} \\
 &= \frac{1}{e_1 e_2} \left[\frac{\partial(e_2 u)}{\partial i} \Big|_s + \frac{\partial(e_1 v)}{\partial j} \Big|_s \right] + \frac{1}{e_3} \left[\frac{\partial w}{\partial s} - \sigma_1 \frac{\partial u}{\partial s} - \sigma_2 \frac{\partial v}{\partial s} \right] \\
 &= \frac{1}{e_1 e_2 e_3} \left[\frac{\partial(e_2 e_3 u)}{\partial i} \Big|_s - e_2 u \frac{\partial e_3}{\partial i} \Big|_s + \frac{\partial(e_1 e_3 v)}{\partial j} \Big|_s - e_1 v \frac{\partial e_3}{\partial j} \Big|_s \right] \\
 &\quad + \frac{1}{e_3} \left[\frac{\partial w}{\partial s} - \sigma_1 \frac{\partial u}{\partial s} - \sigma_2 \frac{\partial v}{\partial s} \right]
 \end{aligned}$$

Noting that $\frac{1}{e_1} \frac{\partial e_3}{\partial i} \Big|_s = \frac{1}{e_1} \frac{\partial^2 z}{\partial i \partial s} \Big|_s = \frac{\partial}{\partial s} \left(\frac{1}{e_1} \frac{\partial z}{\partial i} \Big|_s \right) = \frac{\partial \sigma_1}{\partial s}$ and $\frac{1}{e_2} \frac{\partial e_3}{\partial j} \Big|_s = \frac{\partial \sigma_2}{\partial s}$, it becomes:

$$\begin{aligned}
 \nabla \cdot \mathbf{U} &= \frac{1}{e_1 e_2 e_3} \left[\frac{\partial(e_2 e_3 u)}{\partial i} \Big|_s + \frac{\partial(e_1 e_3 v)}{\partial j} \Big|_s \right] \\
 &\quad + \frac{1}{e_3} \left[\frac{\partial w}{\partial s} - u \frac{\partial \sigma_1}{\partial s} - v \frac{\partial \sigma_2}{\partial s} - \sigma_1 \frac{\partial u}{\partial s} - \sigma_2 \frac{\partial v}{\partial s} \right] \\
 &= \frac{1}{e_1 e_2 e_3} \left[\frac{\partial(e_2 e_3 u)}{\partial i} \Big|_s + \frac{\partial(e_1 e_3 v)}{\partial j} \Big|_s \right] + \frac{1}{e_3} \frac{\partial}{\partial s} [w - u \sigma_1 - v \sigma_2]
 \end{aligned}$$

Here, w is the vertical velocity relative to the z -coordinate system. Using the first form of (equation A.2) and the definitions (equation A.1) and (equation A.6) for σ_1 , σ_2 and w_s , one can show that the vertical velocity, w_p of a point moving with the horizontal velocity of the fluid along an s surface is given by

$$\begin{aligned}
 w_p &= \frac{\partial z}{\partial t} \Big|_s + \frac{u}{e_1} \frac{\partial z}{\partial i} \Big|_s + \frac{v}{e_2} \frac{\partial z}{\partial j} \Big|_s \\
 &= w_s + u \sigma_1 + v \sigma_2.
 \end{aligned} \tag{A.9}$$

The vertical velocity across this surface is denoted by

$$\omega = w - w_p = w - (w_s + \sigma_1 u + \sigma_2 v). \tag{A.10}$$

Hence

$$\frac{1}{e_3} \frac{\partial}{\partial s} [w - u \sigma_1 - v \sigma_2] = \frac{1}{e_3} \frac{\partial}{\partial s} [\omega + w_s] = \frac{1}{e_3} \left[\frac{\partial \omega}{\partial s} + \frac{\partial}{\partial t} \Big|_s \frac{\partial z}{\partial s} \right] = \frac{1}{e_3} \frac{\partial \omega}{\partial s} + \frac{1}{e_3} \frac{\partial e_3}{\partial t} \Big|_s. \tag{A.11}$$

Using (equation A.10) in our expression for $\nabla \cdot \mathbf{U}$ we obtain our final expression for the divergence of the velocity in the curvilinear s -coordinate system:

$$\nabla \cdot \mathbf{U} = \frac{1}{e_1 e_2 e_3} \left[\frac{\partial(e_2 e_3 u)}{\partial i} \Big|_s + \frac{\partial(e_1 e_3 v)}{\partial j} \Big|_s \right] + \frac{1}{e_3} \frac{\partial \omega}{\partial s} + \frac{1}{e_3} \frac{\partial e_3}{\partial t} \Big|_s. \tag{A.12}$$

As a result, the continuity equation equation 1.3 in the s -coordinates is:

$$\frac{1}{e_3} \frac{\partial e_3}{\partial t} + \frac{1}{e_1 e_2 e_3} \left[\frac{\partial(e_2 e_3 u)}{\partial i} \Big|_s + \frac{\partial(e_1 e_3 v)}{\partial j} \Big|_s \right] + \frac{1}{e_3} \frac{\partial \omega}{\partial s} = 0. \tag{A.13}$$

An additional term has appeared that takes 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 equation 1.13, 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:

$$\begin{aligned} \frac{Du}{Dt} \Big|_z &= \frac{\partial u}{\partial t} \Big|_z - \zeta \Big|_z v + \frac{1}{2e_1} \frac{\partial(u^2+v^2)}{\partial i} \Big|_z + w \frac{\partial u}{\partial z} \\ &= \frac{\partial u}{\partial t} \Big|_z - \frac{1}{e_1 e_2} \left[\frac{\partial(e_2 v)}{\partial i} \Big|_z - \frac{\partial(e_1 u)}{\partial j} \Big|_z \right] v + \frac{1}{2e_1} \frac{\partial(u^2+v^2)}{\partial i} \Big|_z + w \frac{\partial u}{\partial z} \end{aligned}$$

introducing the chain rule (equation A.3)

$$\begin{aligned} &= \frac{\partial u}{\partial t} \Big|_z - \frac{1}{e_1 e_2} \left[\frac{\partial(e_2 v)}{\partial i} \Big|_s - \frac{\partial(e_1 u)}{\partial j} \Big|_s - \frac{e_1}{e_3} \sigma_1 \frac{\partial(e_2 v)}{\partial s} + \frac{e_2}{e_3} \sigma_2 \frac{\partial(e_1 u)}{\partial s} \right] v \\ &\quad + \frac{1}{2e_1} \left(\frac{\partial(u^2+v^2)}{\partial i} \Big|_s - \frac{e_1}{e_3} \sigma_1 \frac{\partial(u^2+v^2)}{\partial s} \right) + \frac{w}{e_3} \frac{\partial u}{\partial s} \\ &= \frac{\partial u}{\partial t} \Big|_z - \zeta \Big|_s v + \frac{1}{2e_1} \frac{\partial(u^2+v^2)}{\partial i} \Big|_s \\ &\quad + \frac{w}{e_3} \frac{\partial u}{\partial s} + \left[\frac{\sigma_1}{e_3} \frac{\partial v}{\partial s} - \frac{\sigma_2}{e_3} \frac{\partial u}{\partial s} \right] v - \frac{\sigma_1}{2e_3} \frac{\partial(u^2+v^2)}{\partial s} \\ &= \frac{\partial u}{\partial t} \Big|_z - \zeta \Big|_s v + \frac{1}{2e_1} \frac{\partial(u^2+v^2)}{\partial i} \Big|_s \\ &\quad + \frac{1}{e_3} \left[w \frac{\partial u}{\partial s} + \sigma_1 v \frac{\partial v}{\partial s} - \sigma_2 v \frac{\partial u}{\partial s} - \sigma_1 u \frac{\partial u}{\partial s} - \sigma_1 v \frac{\partial v}{\partial s} \right] \\ &= \frac{\partial u}{\partial t} \Big|_z - \zeta \Big|_s v + \frac{1}{2e_1} \frac{\partial(u^2+v^2)}{\partial i} \Big|_s + \frac{1}{e_3} [w - \sigma_2 v - \sigma_1 u] \frac{\partial u}{\partial s}. \end{aligned}$$

Introducing ω , the dia- s -surface velocity given by (equation A.10)

$$= \frac{\partial u}{\partial t} \Big|_z - \zeta \Big|_s v + \frac{1}{2e_1} \frac{\partial(u^2+v^2)}{\partial i} \Big|_s + \frac{1}{e_3} (\omega + w_s) \frac{\partial u}{\partial s}$$

Applying the time derivative chain rule (first equation of (equation A.3)) to u and using (equation A.6) provides the expression of the last term of the right hand side,

$$\frac{w_s}{e_3} \frac{\partial u}{\partial s} = - \frac{\partial s}{\partial t} \Big|_z \frac{\partial u}{\partial s} = \frac{\partial u}{\partial t} \Big|_s - \frac{\partial u}{\partial t} \Big|_z.$$

This leads to the s -coordinate formulation of the total z -coordinate time derivative, *i.e.* the total s -coordinate time derivative :

$$\frac{Du}{Dt} \Big|_s = \frac{\partial u}{\partial t} \Big|_s - \zeta \Big|_s v + \frac{1}{2e_1} \frac{\partial(u^2+v^2)}{\partial i} \Big|_s + \frac{1}{e_3} \omega \frac{\partial u}{\partial s}. \quad (\text{A.15})$$

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 (equation 1.12), it can be transformed into :

$$\begin{aligned} \frac{Du}{Dt} \Big|_s &= \frac{\partial u}{\partial t} \Big|_s - \zeta v + \frac{1}{2e_1} \frac{\partial(u^2+v^2)}{\partial i} + \frac{1}{e_3} \omega \frac{\partial u}{\partial s} \\ &= \frac{\partial u}{\partial t} \Big|_s + \frac{1}{e_1 e_2} \left(\frac{\partial(e_2 u u)}{\partial i} + \frac{\partial(e_1 u v)}{\partial j} \right) + \frac{1}{e_3} \frac{\partial(\omega u)}{\partial s} \\ &\quad - u \left[\frac{1}{e_1 e_2} \left(\frac{\partial(e_2 u)}{\partial i} + \frac{\partial(e_1 v)}{\partial j} \right) + \frac{1}{e_3} \frac{\partial \omega}{\partial s} \right] \\ &\quad - \frac{v}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right). \end{aligned}$$

Introducing the vertical scale factor inside the horizontal derivative of the first two terms (*i.e.* the horizontal divergence),

it becomes :

$$\begin{aligned}
 &= \left. \frac{\partial u}{\partial t} \right|_s + \frac{1}{e_1 e_2 e_3} \left(\frac{\partial(e_2 e_3 u^2)}{\partial i} + \frac{\partial(e_1 e_3 u v)}{\partial j} - e_2 u u \frac{\partial e_3}{\partial i} - e_1 u v \frac{\partial e_3}{\partial j} \right) + \frac{1}{e_3} \frac{\partial(\omega u)}{\partial s} \\
 &\quad - u \left[\frac{1}{e_1 e_2 e_3} \left(\frac{\partial(e_2 e_3 u)}{\partial i} + \frac{\partial(e_1 e_3 v)}{\partial j} - e_2 u \frac{\partial e_3}{\partial i} - e_1 v \frac{\partial e_3}{\partial j} \right) + \frac{1}{e_3} \frac{\partial \omega}{\partial s} \right] \\
 &\quad - \frac{v}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \\
 &= \left. \frac{\partial u}{\partial t} \right|_s + \frac{1}{e_1 e_2 e_3} \left(\frac{\partial(e_2 e_3 u u)}{\partial i} + \frac{\partial(e_1 e_3 u v)}{\partial j} \right) + \frac{1}{e_3} \frac{\partial(\omega u)}{\partial s} \\
 &\quad - u \left[\frac{1}{e_1 e_2 e_3} \left(\frac{\partial(e_2 e_3 u)}{\partial i} + \frac{\partial(e_1 e_3 v)}{\partial j} \right) + \frac{1}{e_3} \frac{\partial \omega}{\partial s} \right] - \frac{v}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right).
 \end{aligned}$$

Introducing a more compact form for the divergence of the momentum fluxes, and using (equation A.13), the s -coordinate continuity equation, it becomes :

$$= \left. \frac{\partial u}{\partial t} \right|_s + \nabla \cdot (U u)|_s + u \frac{1}{e_3} \frac{\partial e_3}{\partial t} - \frac{v}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right)$$

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:

$$\left. \frac{Du}{Dt} \right|_s = \frac{1}{e_3} \left. \frac{\partial(e_3 u)}{\partial t} \right|_s + \nabla \cdot (U u)|_s - \frac{v}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right). \quad (\text{A.16})$$

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 (equation A.13), 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 equation A.1, it becomes:

$$\begin{aligned}
 -\frac{1}{\rho_o e_1} \left. \frac{\partial p}{\partial i} \right|_z &= -\frac{1}{\rho_o e_1} \left(\left. \frac{\partial p}{\partial i} \right|_s + g \rho \left. \frac{\partial z}{\partial i} \right|_s \right) \\
 -\frac{1}{\rho_o e_2} \left. \frac{\partial p}{\partial j} \right|_z &= -\frac{1}{\rho_o e_2} \left(\left. \frac{\partial p}{\partial j} \right|_s + g \rho \left. \frac{\partial z}{\partial j} \right|_s \right). \quad (\text{A.17})
 \end{aligned}$$

An additional term appears in (equation A.17) 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 Marsaleix et al. (2008). 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:

$$\begin{aligned}
 p &= g \int_z^\eta \rho e_3 dk = g \int_z^\eta \rho_o (d + 1) e_3 dk \\
 &= g \rho_o \int_z^\eta d e_3 dk + \rho_o g \int_z^\eta e_3 dk.
 \end{aligned}$$

Therefore, p and p'_h are linked through:

$$p = \rho_o p'_h + \rho_o g (\eta - z) \quad (\text{A.18})$$

and the hydrostatic pressure balance expressed in terms of p'_h and d is:

$$\frac{\partial p'_h}{\partial k} = -d g e_3.$$

Substituting equation A.18 in equation A.17 and using the definition of the density anomaly it becomes an expression in two parts:

$$\begin{aligned} -\frac{1}{\rho_o e_1} \frac{\partial p}{\partial i} \Big|_z &= -\frac{1}{e_1} \left(\frac{\partial p'_h}{\partial i} \Big|_s + g d \frac{\partial z}{\partial i} \Big|_s \right) - \frac{g}{e_1} \frac{\partial \eta}{\partial i}, \\ -\frac{1}{\rho_o e_2} \frac{\partial p}{\partial j} \Big|_z &= -\frac{1}{e_2} \left(\frac{\partial p'_h}{\partial j} \Big|_s + g d \frac{\partial z}{\partial j} \Big|_s \right) - \frac{g}{e_2} \frac{\partial \eta}{\partial j}. \end{aligned} \quad (\text{A.19})$$

This formulation of the pressure gradient is characterised by the appearance of a term depending on the sea surface height only (last term on the right hand side of expression equation A.19). This term will be loosely termed *surface pressure gradient* whereas the first term will be termed the *hydrostatic pressure gradient* by analogy to the z -coordinate formulation. In fact, 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, except for the pressure gradient term:

$$\begin{aligned} \frac{\partial u}{\partial t} &= +(\zeta + f) v - \frac{1}{2 e_1} \frac{\partial}{\partial i} (u^2 + v^2) - \frac{1}{e_3} \omega \frac{\partial u}{\partial k} \\ &\quad - \frac{1}{e_1} \left(\frac{\partial p'_h}{\partial i} + g d \frac{\partial z}{\partial i} \right) - \frac{g}{e_1} \frac{\partial \eta}{\partial i} + D_u^U + F_u^U, \end{aligned} \quad (\text{A.20a})$$

$$\begin{aligned} \frac{\partial v}{\partial t} &= -(\zeta + f) u - \frac{1}{2 e_2} \frac{\partial}{\partial j} (u^2 + v^2) - \frac{1}{e_3} \omega \frac{\partial v}{\partial k} \\ &\quad - \frac{1}{e_2} \left(\frac{\partial p'_h}{\partial j} + g d \frac{\partial z}{\partial j} \right) - \frac{g}{e_2} \frac{\partial \eta}{\partial j} + D_v^U + F_v^U. \end{aligned} \quad (\text{A.20b})$$

whereas the flux form momentum equation differs from it by the formulation of both the time derivative and the pressure gradient term:

$$\begin{aligned} \frac{1}{e_3} \frac{\partial (e_3 u)}{\partial t} &= -\nabla \cdot (U u) + \left\{ f + \frac{1}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right\} v \\ &\quad - \frac{1}{e_1} \left(\frac{\partial p'_h}{\partial i} + g d \frac{\partial z}{\partial i} \right) - \frac{g}{e_1} \frac{\partial \eta}{\partial i} + D_u^U + F_u^U, \end{aligned} \quad (\text{A.21a})$$

$$\begin{aligned} \frac{1}{e_3} \frac{\partial (e_3 v)}{\partial t} &= -\nabla \cdot (U v) - \left\{ f + \frac{1}{e_1 e_2} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right\} u \\ &\quad - \frac{1}{e_2} \left(\frac{\partial p'_h}{\partial j} + g d \frac{\partial z}{\partial j} \right) - \frac{g}{e_2} \frac{\partial \eta}{\partial j} + D_v^U + F_v^U. \end{aligned} \quad (\text{A.21b})$$

Both formulation share the same hydrostatic pressure balance expressed in terms of hydrostatic pressure and density anomalies, p'_h and $d = (\frac{\rho}{\rho_o} - 1)$:

$$\frac{\partial p'_h}{\partial k} = -d g e_3. \quad (\text{A.22})$$

It is important to realize that the change in coordinate system has only concerned the position on the vertical. It has not affected $(\mathbf{i}, \mathbf{j}, \mathbf{k})$, the orthogonal curvilinear set of unit vectors. (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 volume flux across the moving s -surfaces per unit horizontal area.

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 :

$$\frac{1}{e_3} \frac{\partial (e_3 T)}{\partial t} = -\frac{1}{e_1 e_2 e_3} \left[\frac{\partial}{\partial i} (e_2 e_3 T u) + \frac{\partial}{\partial j} (e_1 e_3 T v) \right] - \frac{1}{e_3} \frac{\partial}{\partial k} (T w) + D^T + F^T \quad (\text{A.23})$$

The expression for the advection term is a straight consequence of (equation A.13), the expression of the 3D divergence in the s -coordinates established above.



Diffusive Operators

Table of contents

- B.1. Horizontal/Vertical 2nd order tracer diffusive operators 205
- B.2. Iso/Diapycnal 2nd order tracer diffusive operators 206
- B.3. Lateral/Vertical momentum diffusive operators 208

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

B.1. Horizontal/Vertical 2^{nd} order tracer diffusive operators

In z-coordinates

In z -coordinates, 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 generalized vertical coordinates

In s -coordinates, we defined the slopes of s -surfaces, σ_1 and σ_2 by [equation A.1](#) and the vertical/horizontal ratio of diffusion coefficient by $\epsilon = A^{vT}/A^{lT}$. The diffusion operator is given by:

$$D^T = \nabla|_s \cdot [A^{lT} \mathfrak{R} \cdot \nabla|_s T] \quad \text{where } \mathfrak{R} = \begin{pmatrix} 1 & 0 & -\sigma_1 \\ 0 & 1 & -\sigma_2 \\ -\sigma_1 & -\sigma_2 & \epsilon + \sigma_1^2 + \sigma_2^2 \end{pmatrix} \quad (\text{B.2})$$

or in expanded form:

$$D^T = \frac{1}{e_1 e_2 e_3} \left\{ \begin{aligned} & \frac{\partial}{\partial i} \left[e_2 e_3 A^{lT} \left(\frac{1}{e_1} \frac{\partial T}{\partial i} \Big|_s - \frac{\sigma_1}{e_3} \frac{\partial T}{\partial s} \right) \right] \Big|_s \\ & + \frac{\partial}{\partial j} \left[e_1 e_3 A^{lT} \left(\frac{1}{e_2} \frac{\partial T}{\partial j} \Big|_s - \frac{\sigma_2}{e_3} \frac{\partial T}{\partial s} \right) \right] \Big|_s \\ & + e_1 e_2 \frac{\partial}{\partial s} \left[A^{lT} \left(-\frac{\sigma_1}{e_1} \frac{\partial T}{\partial i} \Big|_s - \frac{\sigma_2}{e_2} \frac{\partial T}{\partial j} \Big|_s + (\epsilon + \sigma_1^2 + \sigma_2^2) \frac{1}{e_3} \frac{\partial T}{\partial s} \right) \right] \end{aligned} \right\}.$$

[equation B.2](#) is obtained from [equation 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 [equation A.1](#) and [equation A.3](#). Since no cross horizontal derivative $\partial_i \partial_j$ appears in [equation B.1](#), the (i, z) and (j, z) planes are independent. The derivation can then be demonstrated for the $(i, z) \rightarrow (j, s)$ transformation without any loss of generality:

$$\begin{aligned}
 D^T &= \frac{1}{e_1 e_2} \frac{\partial}{\partial i} \left(\frac{e_2}{e_1} A^{lT} \frac{\partial T}{\partial i} \Big|_z \right) \Big|_z + \frac{\partial}{\partial z} \left(A^{vT} \frac{\partial T}{\partial z} \right) \\
 &= \frac{1}{e_1 e_2} \left[\frac{\partial}{\partial i} \left(\frac{e_2}{e_1} A^{lT} \left(\frac{\partial T}{\partial i} \Big|_s - \frac{e_1 \sigma_1}{e_3} \frac{\partial T}{\partial s} \right) \right) \Big|_s \right. \\
 &\quad \left. - \frac{e_1 \sigma_1}{e_3} \frac{\partial}{\partial s} \left(\frac{e_2}{e_1} A^{lT} \left(\frac{\partial T}{\partial i} \Big|_s - \frac{e_1 \sigma_1}{e_3} \frac{\partial T}{\partial s} \right) \right) \Big|_s \right] + \frac{1}{e_3} \frac{\partial}{\partial s} \left[\frac{A^{vT}}{e_3} \frac{\partial T}{\partial s} \right] \\
 &= \frac{1}{e_1 e_2 e_3} \left[\frac{\partial}{\partial i} \left(\frac{e_2 e_3}{e_1} A^{lT} \frac{\partial T}{\partial i} \Big|_s \right) \Big|_s - \frac{e_2}{e_1} A^{lT} \frac{\partial e_3}{\partial i} \Big|_s \frac{\partial T}{\partial i} \Big|_s \right. \\
 &\quad \left. - e_3 \frac{\partial}{\partial i} \left(\frac{e_2 \sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \right) \Big|_s - e_1 \sigma_1 \frac{\partial}{\partial s} \left(\frac{e_2}{e_1} A^{lT} \frac{\partial T}{\partial i} \Big|_s \right) \right. \\
 &\quad \left. - e_1 \sigma_1 \frac{\partial}{\partial s} \left(-\frac{e_2 \sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \right) \right. \quad \left. + \frac{\partial}{\partial s} \left(\frac{e_1 e_2}{e_3} A^{vT} \frac{\partial T}{\partial s} \right) \right]
 \end{aligned}$$

Noting that $\frac{1}{e_1} \frac{\partial e_3}{\partial i} \Big|_s = \frac{\partial \sigma_1}{\partial s}$, this becomes:

$$\begin{aligned}
 D^T &= \frac{1}{e_1 e_2 e_3} \left[\frac{\partial}{\partial i} \left(\frac{e_2 e_3}{e_1} A^{lT} \frac{\partial T}{\partial i} \Big|_s \right) \Big|_s - e_3 \frac{\partial}{\partial i} \left(\frac{e_2 \sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \right) \Big|_s \right. \\
 &\quad \left. - e_2 A^{lT} \frac{\partial \sigma_1}{\partial s} \frac{\partial T}{\partial i} \Big|_s - e_1 \sigma_1 \frac{\partial}{\partial s} \left(\frac{e_2}{e_1} A^{lT} \frac{\partial T}{\partial i} \Big|_s \right) \right. \\
 &\quad \left. + e_1 \sigma_1 \frac{\partial}{\partial s} \left(\frac{e_2 \sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \right) + \frac{\partial}{\partial s} \left(\frac{e_1 e_2}{e_3} A^{vT} \frac{\partial T}{\partial s} \right) \right] \\
 &= \frac{1}{e_1 e_2 e_3} \left[\frac{\partial}{\partial i} \left(\frac{e_2 e_3}{e_1} A^{lT} \frac{\partial T}{\partial i} \Big|_s \right) \Big|_s - \frac{\partial}{\partial i} \left(e_2 \sigma_1 A^{lT} \frac{\partial T}{\partial s} \right) \Big|_s \right. \\
 &\quad \left. + \frac{e_2 \sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \frac{\partial e_3}{\partial i} \Big|_s - e_2 A^{lT} \frac{\partial \sigma_1}{\partial s} \frac{\partial T}{\partial i} \Big|_s \right. \\
 &\quad \left. - e_2 \sigma_1 \frac{\partial}{\partial s} \left(A^{lT} \frac{\partial T}{\partial i} \Big|_s \right) + \frac{\partial}{\partial s} \left(\frac{e_1 e_2 \sigma_1^2}{e_3} A^{lT} \frac{\partial T}{\partial s} \right) \right. \\
 &\quad \left. - \frac{\partial(e_1 e_2 \sigma_1)}{\partial s} \left(\frac{\sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \right) + \frac{\partial}{\partial s} \left(\frac{e_1 e_2}{e_3} A^{vT} \frac{\partial T}{\partial s} \right) \right] .
 \end{aligned}$$

Using the same remark as just above, D^T becomes:

$$\begin{aligned}
 D^T &= \frac{1}{e_1 e_2 e_3} \left[\frac{\partial}{\partial i} \left(\frac{e_2 e_3}{e_1} A^{lT} \frac{\partial T}{\partial i} \Big|_s - e_2 \sigma_1 A^{lT} \frac{\partial T}{\partial s} \right) \Big|_s \right. \\
 &\quad \left. + \frac{e_1 e_2 \sigma_1}{e_3} A^{lT} \frac{\partial T}{\partial s} \frac{\partial \sigma_1}{\partial s} - \frac{\sigma_1}{e_3} A^{lT} \frac{\partial(e_1 e_2 \sigma_1)}{\partial s} \frac{\partial T}{\partial s} \right. \\
 &\quad \left. - e_2 \left(A^{lT} \frac{\partial \sigma_1}{\partial s} \frac{\partial T}{\partial i} \Big|_s + \frac{\partial}{\partial s} \left(\sigma_1 A^{lT} \frac{\partial T}{\partial i} \Big|_s \right) - \frac{\partial \sigma_1}{\partial s} A^{lT} \frac{\partial T}{\partial i} \Big|_s \right) \right. \\
 &\quad \left. + \frac{\partial}{\partial s} \left(\frac{e_1 e_2 \sigma_1^2}{e_3} A^{lT} \frac{\partial T}{\partial s} + \frac{e_1 e_2}{e_3} A^{vT} \frac{\partial T}{\partial s} \right) \right] .
 \end{aligned}$$

Since the horizontal scale factors do not depend on the vertical coordinate, the two terms on the second line cancel, while the third line reduces to a single vertical derivative, so it becomes:

$$\begin{aligned}
 D^T &= \frac{1}{e_1 e_2 e_3} \left[\frac{\partial}{\partial i} \left(\frac{e_2 e_3}{e_1} A^{lT} \frac{\partial T}{\partial i} \Big|_s - e_2 \sigma_1 A^{lT} \frac{\partial T}{\partial s} \right) \Big|_s \right. \\
 &\quad \left. + \frac{\partial}{\partial s} \left(-e_2 \sigma_1 A^{lT} \frac{\partial T}{\partial i} \Big|_s + A^{lT} \frac{e_1 e_2}{e_3} (\varepsilon + \sigma_1^2) \frac{\partial T}{\partial s} \right) \right]
 \end{aligned}$$

In other words, the horizontal/vertical Laplacian operator in the (i,s) plane takes the following form:

$$\frac{1}{e_1 e_2 e_3} \left(\frac{\partial(e_2 e_3 \bullet)}{\partial i} \Big|_s \right) \cdot \left[A^{lT} \begin{pmatrix} 1 & -\sigma_1 \\ -\sigma_1 & \varepsilon + \sigma_1^2 \end{pmatrix} \cdot \left(\frac{1}{e_3} \frac{\partial \bullet}{\partial i} \Big|_s \right) \right] \cdot \left(\frac{1}{e_3} \frac{\partial \bullet}{\partial s} \right) (T)$$

B.2. Iso/Diapycnal 2nd order tracer diffusive operators

In z-coordinates

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 (Redi, 1982):

$$\mathbf{A}_I = \frac{A^{lT}}{(1 + a_1^2 + a_2^2)} \begin{bmatrix} 1 + a_2^2 + \varepsilon a_1^2 & -a_1 a_2 (1 - \varepsilon) & -a_1 (1 - \varepsilon) \\ -a_1 a_2 (1 - \varepsilon) & 1 + a_1^2 + \varepsilon a_2^2 & -a_2 (1 - \varepsilon) \\ -a_1 (1 - \varepsilon) & -a_2 (1 - \varepsilon) & \varepsilon + a_1^2 + a_2^2 \end{bmatrix} \quad (\text{B.3})$$

where (a_1, a_2) are $(-1) \times$ the isopycnal slopes in (\mathbf{i}, \mathbf{j}) directions, relative to geopotentials (or equivalently the slopes of the geopotential surfaces in the isopycnal coordinate framework):

$$a_1 = \frac{e_3}{e_1} \left(\frac{\partial \rho}{\partial i} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1}, \quad a_2 = \frac{e_3}{e_2} \left(\frac{\partial \rho}{\partial j} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1}$$

and, as before, $\epsilon = A^{vT}/A^{lT}$.

In practice, ϵ is small and isopycnal slopes are generally less than 10^{-2} in the ocean, so \mathbf{A}_I can be simplified appreciably (Cox, 1987). Keeping leading order terms*:

$$\mathbf{A}_I \approx A^{lT} \mathfrak{R} \text{ where } \mathfrak{R} = \begin{bmatrix} 1 & 0 & -a_1 \\ 0 & 1 & -a_2 \\ -a_1 & -a_2 & \epsilon + a_1^2 + a_2^2 \end{bmatrix}, \quad (\text{B.4a})$$

and the iso/dianeutral diffusive operator in z -coordinates is then

$$D^T = \nabla|_z \cdot [A^{lT} \mathfrak{R} \cdot \nabla|_z T]. \quad (\text{B.4b})$$

Physically, the full tensor [equation B.3](#) represents strong isoneutral diffusion on a plane parallel to the isoneutral surface and weak dianeutral diffusion perpendicular to this plane. However, the approximate ‘weak-slope’ tensor [equation B.4a](#) represents strong diffusion along the isoneutral surface, with weak *vertical* diffusion – the principal axes of the tensor are no longer orthogonal. This simplification also decouples the (i, z) and (j, z) planes of the tensor. The weak-slope operator therefore takes the same form, [equation B.4](#), as [equation B.2](#), the diffusion operator for geopotential diffusion written in non-orthogonal i, j, s -coordinates. Written out explicitly,

$$D^T = \frac{1}{e_1 e_2} \left\{ \frac{\partial}{\partial i} \left[A_h \left(\frac{e_2}{e_1} \frac{\partial T}{\partial i} - a_1 \frac{e_2}{e_3} \frac{\partial T}{\partial k} \right) \right] + \frac{\partial}{\partial j} \left[A_h \left(\frac{e_1}{e_2} \frac{\partial T}{\partial j} - a_2 \frac{e_1}{e_3} \frac{\partial T}{\partial k} \right) \right] \right\} \\ + \frac{1}{e_3} \frac{\partial}{\partial k} \left[A_h \left(-\frac{a_1}{e_1} \frac{\partial T}{\partial i} - \frac{a_2}{e_2} \frac{\partial T}{\partial j} + \frac{(a_1^2 + a_2^2 + \epsilon)}{e_3} \frac{\partial T}{\partial k} \right) \right] \quad (\text{B.5})$$

The isopycnal diffusion operator [equation B.4](#), [equation B.5](#) conserves tracer quantity and dissipates its square. As [equation B.4](#) is the divergence of a flux, the demonstration of the first property is trivial, providing that the flux normal to the boundary is zero (as it is when A_h is zero at the boundary). Let us demonstrate the second one:

$$\iiint_D T \nabla \cdot (\mathbf{A}_I \nabla T) dv = - \iiint_D \nabla T \cdot (\mathbf{A}_I \nabla T) dv,$$

and since

$$\begin{aligned} \nabla T \cdot (\mathbf{A}_I \nabla T) &= A^{lT} \left[\left(\frac{\partial T}{\partial i} \right)^2 - 2a_1 \frac{\partial T}{\partial i} \frac{\partial T}{\partial k} + \left(\frac{\partial T}{\partial j} \right)^2 \right. \\ &\quad \left. - 2a_2 \frac{\partial T}{\partial j} \frac{\partial T}{\partial k} + (a_1^2 + a_2^2 + \epsilon) \left(\frac{\partial T}{\partial k} \right)^2 \right] \\ &= A_h \left[\left(\frac{\partial T}{\partial i} - a_1 \frac{\partial T}{\partial k} \right)^2 + \left(\frac{\partial T}{\partial j} - a_2 \frac{\partial T}{\partial k} \right)^2 + \epsilon \left(\frac{\partial T}{\partial k} \right)^2 \right] \\ &\geq 0. \end{aligned}$$

the property becomes obvious.

In generalized vertical coordinates

Because the weak-slope operator [equation B.4](#), [equation B.5](#) is decoupled in the (i, z) and (j, z) planes, it may be transformed into generalized s -coordinates in the same way as [section B.1](#) was transformed into [section B.2](#). The resulting operator then takes the simple form

$$D^T = \nabla|_s \cdot [A^{lT} \mathfrak{R} \cdot \nabla|_s T] \text{ where } \mathfrak{R} = \begin{pmatrix} 1 & 0 & -r_1 \\ 0 & 1 & -r_2 \\ -r_1 & -r_2 & \epsilon + r_1^2 + r_2^2 \end{pmatrix}, \quad (\text{B.6})$$

*Apart from the (1,0) and (0,1) elements which are set to zero. See Griffies (2004), section 14.1.4.1 for a discussion of this point.

where (r_1, r_2) are $(-1) \times$ the isopycnal slopes in (\mathbf{i}, \mathbf{j}) directions, relative to s -coordinate surfaces (or equivalently the slopes of the s -coordinate surfaces in the isopycnal coordinate framework):

$$r_1 = \frac{e_3}{e_1} \left(\frac{\partial \rho}{\partial i} \right) \left(\frac{\partial \rho}{\partial s} \right)^{-1}, \quad r_2 = \frac{e_3}{e_2} \left(\frac{\partial \rho}{\partial j} \right) \left(\frac{\partial \rho}{\partial s} \right)^{-1}.$$

To prove [equation B.6](#) by direct re-expression of [equation B.5](#) is straightforward, but laborious. An easier way is first to note (by reversing the derivation of [section B.2](#) from [section B.1](#)) that the weak-slope operator may be *exactly* reexpressed in non-orthogonal i, j, ρ -coordinates as

$$D^T = \nabla|_\rho \cdot \left[A^{lT} \mathfrak{R} \cdot \nabla|_\rho T \right] \quad \text{where } \mathfrak{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \varepsilon \end{pmatrix}. \quad (\text{B.7})$$

Then direct transformation from i, j, ρ -coordinates to i, j, s -coordinates gives [equation B.6](#) immediately.

Note that the weak-slope approximation is only made in transforming from the (rotated, orthogonal) isoneutral axes to the non-orthogonal i, j, ρ -coordinates. The further transformation into i, j, s -coordinates is exact, whatever the steepness of the s -surfaces, in the same way as the transformation of horizontal/vertical Laplacian diffusion in z -coordinates in [section B.1](#) onto s -coordinates is exact, however steep the s -surfaces.

B.3. Lateral/Vertical momentum diffusive operators

The second order momentum diffusion operator (Laplacian) in z -coordinates is found by applying [equation 1.8e](#), the expression for the Laplacian of a vector, to the horizontal velocity vector:

$$\begin{aligned} \Delta \mathbf{U}_h &= \nabla (\nabla \cdot \mathbf{U}_h) - \nabla \times (\nabla \times \mathbf{U}_h) \\ &= \begin{pmatrix} \frac{1}{e_1} \frac{\partial \chi}{\partial i} \\ \frac{1}{e_2} \frac{\partial \chi}{\partial j} \\ \frac{1}{e_3} \frac{\partial \chi}{\partial k} \end{pmatrix} - \begin{pmatrix} \frac{1}{e_2} \frac{\partial \zeta}{\partial j} - \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial u}{\partial k} \right) \\ \frac{1}{e_3} \frac{\partial}{\partial k} \left(-\frac{1}{e_3} \frac{\partial v}{\partial k} \right) - \frac{1}{e_1} \frac{\partial \zeta}{\partial i} \\ \frac{1}{e_1 e_2} \left[\frac{\partial}{\partial i} \left(\frac{e_2}{e_3} \frac{\partial u}{\partial k} \right) - \frac{\partial}{\partial j} \left(-\frac{e_1}{e_3} \frac{\partial v}{\partial k} \right) \right] \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{e_1} \frac{\partial \chi}{\partial i} - \frac{1}{e_2} \frac{\partial \zeta}{\partial j} \\ \frac{1}{e_2} \frac{\partial \chi}{\partial j} + \frac{1}{e_1} \frac{\partial \zeta}{\partial i} \\ 0 \end{pmatrix} + \frac{1}{e_3} \begin{pmatrix} \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial u}{\partial k} \right) \\ \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial v}{\partial k} \right) \\ \frac{\partial \chi}{\partial k} - \frac{1}{e_1 e_2} \left(\frac{\partial^2 (e_2 u)}{\partial i \partial k} + \frac{\partial^2 (e_1 v)}{\partial j \partial k} \right) \end{pmatrix} \end{aligned}$$

Using [equation 1.8b](#), the definition of the horizontal divergence, the third component of the second vector is obviously zero and thus :

$$\Delta \mathbf{U}_h = \nabla_h (\chi) - \nabla_h \times (\zeta \mathbf{k}) + \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{1}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right).$$

Note that this operator ensures a full separation between the vorticity and horizontal divergence fields (see [appendix C](#)). It is only equal to a Laplacian applied to each component in Cartesian coordinates, not on the sphere.

The horizontal/vertical second order (Laplacian type) operator used to diffuse horizontal momentum in the z -coordinate therefore takes the following form:

$$\mathbf{D}^U = \nabla_h (A^{lm} \chi) - \nabla_h \times (A^{lm} \zeta \mathbf{k}) + \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right), \quad (\text{B.8})$$

that is, in expanded form:

$$\begin{aligned} D_u^U &= \frac{1}{e_1} \frac{\partial (A^{lm} \chi)}{\partial i} - \frac{1}{e_2} \frac{\partial (A^{lm} \zeta)}{\partial j} + \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial u}{\partial k} \right), \\ D_v^U &= \frac{1}{e_2} \frac{\partial (A^{lm} \chi)}{\partial j} + \frac{1}{e_1} \frac{\partial (A^{lm} \zeta)}{\partial i} + \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial v}{\partial k} \right). \end{aligned}$$

Note Bene: introducing a rotation in [equation B.8](#) does not lead to a useful expression for the iso/diapycnal Laplacian operator in the z -coordinate. Similarly, we did not find an expression of practical use for the geopotential horizontal/vertical Laplacian operator in the s -coordinate. Generally, [equation B.8](#) is used in both z - and s -coordinate systems, that is a Laplacian diffusion is applied on momentum along the coordinate directions.



Discrete Invariants of the Equations

Table of contents

- C.1. Introduction / Notations 210
- C.2. Continuous conservation 211
- C.3. Discrete total energy conservation: vector invariant form 213
 - C.3.1. Total energy conservation 213
 - C.3.2. Vorticity term (coriolis + vorticity part of the advection) 213
 - C.3.3. Pressure gradient term 216
- C.4. Discrete total energy conservation: flux form 217
 - C.4.1. Total energy conservation 217
 - C.4.2. Coriolis and advection terms: flux form 217
- C.5. Discrete enstrophy conservation 218
- C.6. Conservation properties on tracers 220
 - C.6.1. Advection term 220
- C.7. Conservation properties on lateral momentum physics 220
 - C.7.1. Conservation of potential vorticity 221
 - C.7.2. Dissipation of horizontal kinetic energy 221
 - C.7.3. Dissipation of enstrophy 222
 - C.7.4. Conservation of horizontal divergence 222
 - C.7.5. Dissipation of horizontal divergence variance 222
- C.8. Conservation properties on vertical momentum physics 222
- C.9. Conservation properties on tracer physics 224
 - C.9.1. Conservation of tracers 224
 - C.9.2. Dissipation of tracer variance 225

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

C.1. Introduction / Notations

Notation used in this appendix in the demonstrations:

fluxes at the faces of a T -box:

$$U = e_{2u} e_{3u} u \quad V = e_{1v} e_{3v} v \quad W = e_{1w} e_{2w} w$$

volume of cells at u -, v -, and T -points:

$$b_u = e_{1u} e_{2u} e_{3u} \quad b_v = e_{1v} e_{2v} e_{3v} \quad b_t = e_{1t} e_{2t} e_{3t}$$

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

$$\partial_t \left(\int_D Q dv \right) = \int_D \partial_t (e_3 Q) e_1 e_2 di dj dk = \int_D \frac{1}{e_3} \partial_t (e_3 Q) dv = 0$$

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:

$$\begin{aligned} \partial_t \left(\int_D \frac{1}{2} Q^2 dv \right) &= \int_D \frac{1}{2} \partial_t \left(\frac{1}{e_3} (e_3 Q)^2 \right) e_1 e_2 di dj dk \\ &= \int_D Q \partial_t (e_3 Q) e_1 e_2 di dj dk - \int_D \frac{1}{2} Q^2 \partial_t (e_3) e_1 e_2 di dj dk \end{aligned}$$

that is in a more compact form :

$$\partial_t \left(\int_D \frac{1}{2} Q^2 dv \right) = \int_D \frac{Q}{e_3} \partial_t (e_3 Q) dv - \frac{1}{2} \int_D \frac{Q^2}{e_3} \partial_t (e_3) dv \quad (\text{C.1})$$

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:

$$\begin{aligned} \partial_t \left(\int_D \frac{1}{2} Q^2 dv \right) &= \int_D \frac{1}{2} \partial_t (e_3 Q^2) e_1 e_2 di dj dk \\ &= \int_D Q \partial_t Q e_1 e_2 e_3 di dj dk + \int_D \frac{1}{2} Q^2 \partial_t e_3 e_1 e_2 di dj dk \end{aligned}$$

that is in a more compact form:

$$\partial_t \left(\int_D \frac{1}{2} Q^2 dv \right) = \int_D Q \partial_t Q dv + \frac{1}{2} \int_D \frac{1}{e_3} Q^2 \partial_t e_3 dv \quad (\text{C.2})$$

C.2. Continuous conservation

The discretization of primitive 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 constraints on energy and enstrophy.

Let us first establish those constraints in the continuous world. The total energy (*i.e.* kinetic plus potential energies) is conserved:

$$\partial_t \left(\int_D \left(\frac{1}{2} \mathbf{U}_h^2 + \rho g z \right) dv \right) = 0 \quad (\text{C.3})$$

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 equation C.2 and introducing equation A.20 in equation C.3 for the former form and using equation C.1 and introducing equation A.21 in equation C.3 for the latter form leads to:

advection term (vector invariant form):

$$\begin{aligned} \int_D \zeta (\mathbf{k} \times \mathbf{U}_h) \cdot \mathbf{U}_h dv &= 0 \\ \int_D \mathbf{U}_h \cdot \nabla_h \left(\frac{\mathbf{U}_h^2}{2} \right) dv + \int_D \mathbf{U}_h \cdot \nabla_z \mathbf{U}_h dv - \int_D \frac{\mathbf{U}_h^2}{2} \frac{1}{e_3} \partial_t e_3 dv &= 0 \end{aligned}$$

advection term (flux form):

$$\begin{aligned} \int_D \frac{1}{e_1 e_2} (v \partial_i e_2 - u \partial_j e_1) (\mathbf{k} \times \mathbf{U}_h) \cdot \mathbf{U}_h dv &= 0 \\ \int_D \mathbf{U}_h \cdot \left(\begin{array}{c} \nabla \cdot (\mathbf{U} u) \\ \nabla \cdot (\mathbf{U} v) \end{array} \right) dv + \frac{1}{2} \int_D \mathbf{U}_h^2 \frac{1}{e_3} \partial_t e_3 dv &= 0 \end{aligned}$$

coriolis term

$$\int_D f (\mathbf{k} \times \mathbf{U}_h) \cdot \mathbf{U}_h dv = 0$$

pressure gradient:

$$- \int_D \nabla p|_z \cdot \mathbf{U}_h dv = - \int_D \nabla \cdot (\rho \mathbf{U}) g z dv + \int_D g \rho \partial_t z dv$$

where $\nabla_h = \nabla|_k$ is the gradient along the s -surfaces.

blah blah....

The prognostic ocean dynamics equation can be summarized as follows:

$$\text{NXT} = \left(\begin{array}{c} \text{VOR} + \text{KEG} + \text{ZAD} \\ \text{COR} + \text{ADV} \end{array} \right) + \text{HPG} + \text{SPG} + \text{LDF} + \text{ZDF}$$

Vector invariant form:

$$\begin{aligned} \int_D \mathbf{U}_h \cdot \text{VOR} dv &= 0 \\ \int_D \mathbf{U}_h \cdot \text{KEG} dv + \int_D \mathbf{U}_h \cdot \text{ZAD} dv - \int_D \frac{\mathbf{U}_h^2}{2} \frac{1}{e_3} \partial_t e_3 dv &= 0 \\ - \int_D \mathbf{U}_h \cdot (\text{HPG} + \text{SPG}) dv = - \int_D \nabla \cdot (\rho \mathbf{U}) g z dv + \int_D g \rho \partial_t z dv \end{aligned}$$

Flux form:

$$\begin{aligned} \int_D \mathbf{U}_h \cdot \text{COR} dv &= 0 \\ \int_D \mathbf{U}_h \cdot \text{ADV} dv + \frac{1}{2} \int_D \mathbf{U}_h^2 \frac{1}{e_3} \partial_t e_3 dv &= 0 \end{aligned}$$

$$-\int_D \mathbf{U}_h \cdot (\text{HPG} + \text{SPG}) \, dv = -\int_D \nabla \cdot (\rho \mathbf{U}) \, g \, z \, dv + \int_D g \rho \, \partial_t z \, dv \quad (\text{C.4a})$$

equation C.4a is the balance between the conversion KE to PE and PE to KE. Indeed the left hand side of equation C.4a 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 equation A.10).

The left hand side of equation C.4a 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, [equation 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:

$$\begin{aligned} \sum_{i,j,k} \left\{ u \partial_t u b_u + v \partial_t v b_v \right\} + \frac{1}{2} \sum_{i,j,k} \left\{ \frac{u^2}{e_{3u}} \partial_t e_{3u} b_u + \frac{v^2}{e_{3v}} \partial_t e_{3v} b_v \right\} \\ = - \sum_{i,j,k} \left\{ \frac{1}{e_{3t}} \partial_t (e_{3t} \rho) g z_t b_t \right\} - \sum_{i,j,k} \left\{ \rho g \partial_t (z_t) b_t \right\} \end{aligned} \quad (C.5)$$

Substituting the discrete expression of the time derivative of the velocity either in vector invariant, leads to the discrete equivalent of the four equations [equation C.4](#).

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:

$$-e_3 q \mathbf{k} \times \mathbf{U}_h \equiv \begin{pmatrix} +\frac{1}{e_{1u}} q \overline{(e_{1v} e_{3v} v)^{i+1/2}}^j \\ -\frac{1}{e_{2v}} q \overline{(e_{2u} e_{3u} u)^{j+1/2}}^i \end{pmatrix}$$

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:

$$\begin{aligned} \int_D - (e_3 q \mathbf{k} \times \mathbf{U}_h) \cdot \mathbf{U}_h dv \\ \equiv \sum_{i,j,k} \left\{ \frac{1}{e_{1u}} q \overline{V^{i+1/2}}^j u b_u - \frac{1}{e_{2v}} q \overline{U^{j+1/2}}^i v b_v \right\} \\ \equiv \sum_{i,j,k} \left\{ q \overline{V^{i+1/2}}^j U - q \overline{U^{j+1/2}}^i V \right\} \\ \equiv \sum_{i,j,k} q \left\{ \overline{V^{i+1/2}} \overline{U^{j+1/2}} - \overline{U^{j+1/2}} \overline{V^{i+1/2}} \right\} \equiv 0 \end{aligned}$$

In other words, the domain averaged kinetic energy does not change due to the vorticity term.

Vorticity term with EEN scheme (`ln_dynvor_eeen=.true.`)

With the EEN scheme, the vorticity terms are represented as:

$$\begin{cases} +q e_3 v \equiv +\frac{1}{e_{1u}} \sum_{i_p, k_p}^{i+1/2-i_p} \mathbb{Q}_{j_p}^{i_p} (e_{1v} e_{3v} v)_{j+ j_p}^{i+ i_p - 1/2} \\ -q e_3 u \equiv -\frac{1}{e_{2v}} \sum_{i_p, k_p}^{i+1/2-j_p} \mathbb{Q}_{j_p}^{i_p} (e_{2u} e_{3u} u)_{j+ j_p - 1/2}^{i+ i_p} \end{cases} \quad (C.6)$$

where the indices i_p and j_p take the following value: $i_p = -1/2$ or $1/2$ and $j_p = -1/2$ or $1/2$, and the vorticity triads, ${}^i_j \mathbb{Q}_{j_p}^{i_p}$, defined at T -point, are given by:

$${}^j_i \mathbb{Q}_{j_p}^{i_p} = \frac{1}{12} \left(q_{j+ j_p}^{i- i_p} + q_{j+ i_p}^{i+ j_p} + q_{j- j_p}^{i+ i_p} \right) \quad (C.7)$$

This formulation does conserve the total kinetic energy. Indeed,

$$\begin{aligned}
 & \int_D -\mathbf{U}_h \cdot (\zeta \mathbf{k} \times \mathbf{U}_h) \, dv \\
 \equiv & \sum_{i,j,k} \left\{ \left[\sum_{i_p, k_p} \sum_j^{i+1/2-i_p} \mathbb{Q}_{j_p}^{i_p} V_{j+j_p}^{i+1/2-i_p} \right] U_j^{i+1/2} - \left[\sum_{i_p, k_p} \sum_j^{i+1/2-j_p} \mathbb{Q}_{j_p}^{i_p} U_{j+1/2-j_p}^{i+i_p} \right] V_{j+1/2}^i \right\} \\
 \equiv & \sum_{i,j,k} \sum_{i_p, k_p} \left\{ \sum_j^{i+1/2-i_p} \mathbb{Q}_{j_p}^{i_p} V_{j+j_p}^{i+1/2-i_p} U_j^{i+1/2} - \sum_j^{i+1/2-j_p} \mathbb{Q}_{j_p}^{i_p} U_{j+1/2-j_p}^{i+i_p} V_{j+1/2}^i \right\}
 \end{aligned}$$

Expanding the summation on i_p and k_p , it becomes:

$$\begin{aligned}
 \equiv & \sum_{i,j,k} \left\{ \begin{aligned}
 & \sum_j^{i+1} \mathbb{Q}_{+1/2}^{-1/2} V_{j+1/2}^{i+1} U_j^{i+1/2} - \sum_j^i \mathbb{Q}_{+1/2}^{-1/2} U_j^{i-1/2} V_{j+1/2}^i \\
 & + \sum_j^{i+1} \mathbb{Q}_{-1/2}^{-1/2} V_{j-1/2}^{i+1} U_j^{i+1/2} - \sum_{j+1}^i \mathbb{Q}_{-1/2}^{-1/2} U_{j+1}^{i-1/2} V_{j+1/2}^i \\
 & + \sum_j^i \mathbb{Q}_{+1/2}^{+1/2} V_{j+1/2}^i U_j^{i+1/2} - \sum_j^i \mathbb{Q}_{+1/2}^{+1/2} U_j^{i+1/2} V_{j+1/2}^i \\
 & + \sum_j^i \mathbb{Q}_{-1/2}^{+1/2} V_{j-1/2}^i U_j^{i+1/2} - \sum_{j+1}^i \mathbb{Q}_{-1/2}^{+1/2} U_{j+1}^{i+1/2} V_{j+1/2}^i \end{aligned} \right\}
 \end{aligned}$$

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 doing 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:

$$\begin{aligned}
 \equiv & \sum_{i,j,k} \left\{ \begin{aligned}
 & \sum_j^i \mathbb{Q}_{+1/2}^{-1/2} \left[V_{j+1/2}^i U_j^{i-1/2} - U_j^{i-1/2} V_{j+1/2}^i \right] \\
 & + \sum_j^i \mathbb{Q}_{-1/2}^{-1/2} \left[V_{j-1/2}^i U_j^{i-1/2} - U_j^{i-1/2} V_{j-1/2}^i \right] \\
 & + \sum_j^i \mathbb{Q}_{+1/2}^{+1/2} \left[V_{j+1/2}^i U_j^{i+1/2} - U_j^{i+1/2} V_{j+1/2}^i \right] \\
 & + \sum_j^i \mathbb{Q}_{-1/2}^{+1/2} \left[V_{j-1/2}^i U_j^{i+1/2} - U_{j-1}^{i+1/2} V_{j-1/2}^i \right] \end{aligned} \right\} \equiv 0
 \end{aligned}$$

Gradient of kinetic energy / Vertical advection

The change of Kinetic Energy (KE) due to the vertical advection is exactly balanced by the change of KE due to the horizontal gradient of KE :

$$\int_D \mathbf{U}_h \cdot \frac{1}{e_3} \omega \partial_k \mathbf{U}_h \, dv = - \int_D \mathbf{U}_h \cdot \nabla_h \left(\frac{1}{2} \mathbf{U}_h^2 \right) \, dv + \frac{1}{2} \int_D \frac{\mathbf{U}_h^2}{e_3} \partial_t (e_3) \, dv$$

Indeed, using successively [equation 3.4](#) (*i.e.* the skew symmetry property of the δ operator) and the continuity equation, then [equation 3.4](#) again, then the commutativity of operators $\bar{\cdot}$ and δ , and finally [equation 3.5](#) (*i.e.* the symmetry property

of the $\bar{\cdot}$ operator) applied in the horizontal and vertical directions, it becomes:

$$\begin{aligned}
 & - \int_D \mathbf{U}_h \cdot \text{KEG} \, dv = - \int_D \mathbf{U}_h \cdot \nabla_h \left(\frac{1}{2} \mathbf{U}_h^2 \right) \, dv \\
 & \equiv - \sum_{i,j,k} \frac{1}{2} \left\{ \frac{1}{e_{1u}} \delta_{i+1/2} \left[\overline{u^{2^i}} + \overline{v^{2^j}} \right] u \, b_u + \frac{1}{e_{2v}} \delta_{j+1/2} \left[\overline{u^{2^i}} + \overline{v^{2^j}} \right] v \, b_v \right\} \\
 & \equiv + \sum_{i,j,k} \frac{1}{2} \left(\overline{u^{2^i}} + \overline{v^{2^j}} \right) \left\{ \delta_i [U] + \delta_j [V] \right\} \\
 & \equiv - \sum_{i,j,k} \frac{1}{2} \left(\overline{u^{2^i}} + \overline{v^{2^j}} \right) \left\{ \frac{b_t}{e_{3t}} \partial_t (e_{3t}) + \delta_k [W] \right\} \\
 & \equiv + \sum_{i,j,k} \frac{1}{2} \delta_{k+1/2} \left[\overline{u^{2^i}} + \overline{v^{2^j}} \right] W - \sum_{i,j,k} \frac{1}{2} \left(\overline{u^{2^i}} + \overline{v^{2^j}} \right) \partial_t b_t \\
 & \equiv + \sum_{i,j,k} \frac{1}{2} \left(\overline{\delta_{k+1/2} [u^2]^i} + \overline{\delta_{k+1/2} [v^2]^j} \right) W - \sum_{i,j,k} \left(\frac{u^2}{2} \partial_t \overline{b_t^{i+1/2}} + \frac{v^2}{2} \partial_t \overline{b_t^{j+1/2}} \right)
 \end{aligned}$$

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:

$$\begin{aligned}
 & \equiv \sum_{i,j,k} \frac{1}{2} \left\{ \overline{W^{i+1/2}} \delta_{k+1/2} [u^2] + \overline{W^{j+1/2}} \delta_{k+1/2} [v^2] \right\} - \sum_{i,j,k} \left(\frac{u^2}{2} \partial_t b_u + \frac{v^2}{2} \partial_t b_v \right) \\
 & \equiv \sum_{i,j,k} \left\{ \overline{W^{i+1/2}} \overline{u^{k+1/2}} \delta_{k+1/2} [u] + \overline{W^{j+1/2}} \overline{v^{k+1/2}} \delta_{k+1/2} [v] \right\} - \sum_{i,j,k} \left(\frac{u^2}{2} \partial_t b_u + \frac{v^2}{2} \partial_t b_v \right) \\
 & \equiv \sum_{i,j,k} \left\{ \frac{1}{b_u} \overline{W^{i+1/2}} \overline{\delta_{k+1/2} [u]^k} u \, b_u + \frac{1}{b_v} \overline{W^{j+1/2}} \overline{\delta_{k+1/2} [v]^k} v \, b_v \right\} - \sum_{i,j,k} \left(\frac{u^2}{2} \partial_t b_u + \frac{v^2}{2} \partial_t b_v \right)
 \end{aligned}$$

The first term provides the discrete expression for the vertical advection of momentum (ZAD), while the second term corresponds exactly to [equation C.5](#), therefore:

$$\begin{aligned}
 & \equiv \int_D \mathbf{U}_h \cdot \text{ZAD} \, dv + \frac{1}{2} \int_D \mathbf{U}_h^2 \frac{1}{e_3} \partial_t (e_3) \, dv \\
 & \equiv \int_D \mathbf{U}_h \cdot w \partial_k \mathbf{U}_h \, dv + \frac{1}{2} \int_D \mathbf{U}_h^2 \frac{1}{e_3} \partial_t (e_3) \, dv
 \end{aligned}$$

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:

$$\frac{1}{e_3} \omega \partial_k \mathbf{U}_h \equiv \left(\begin{array}{c} \frac{1}{e_{1u} e_{2u} e_{3u}} \overline{\overline{\overline{e_{1t} e_{2t} \omega \delta_{k+1/2} [\overline{u^{i+1/2}}]^{i+1/2,k}}}}} \\ \frac{1}{e_{1v} e_{2v} e_{3v}} \overline{\overline{\overline{e_{1t} e_{2t} \omega \delta_{k+1/2} [\overline{v^{j+1/2}}]^{j+1/2,k}}}}} \end{array} \right)$$

a formulation that requires an additional horizontal mean in contrast with the one used in *NEMO*. Nine velocity points have to be used instead of 3. This is the reason why it has not been chosen.

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:

$$\begin{aligned}
 e_{1u} e_{2u} \partial_t (e_{3u}) &= \overline{\overline{\overline{e_{1t} e_{2t} \partial_t (e_{3t})^{i+1/2}}}}} \\
 e_{1v} e_{2v} \partial_t (e_{3v}) &= \overline{\overline{\overline{e_{1t} e_{2t} \partial_t (e_{3t})^{j+1/2}}}}}
 \end{aligned}$$

which is (over-)satisfied by defining the vertical scale factor as follows:

$$\begin{aligned}
 e_{3u} &= \frac{1}{e_{1u} e_{2u}} \overline{\overline{\overline{e_{1t} e_{2t} e_{3t}^{i+1/2}}}}} \\
 e_{3v} &= \frac{1}{e_{1v} e_{2v}} \overline{\overline{\overline{e_{1t} e_{2t} e_{3t}^{j+1/2}}}}}
 \end{aligned}$$

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:

$$-\int_D \nabla p|_z \cdot \mathbf{U}_h \, dv \equiv \sum_{i,j,k} \left\{ -\frac{1}{e_{1u}} \left(\delta_{i+1/2}[p_t] - g \bar{\rho}^{i+1/2} \delta_{i+1/2}[z_t] \right) u \, b_u \right. \\ \left. - \frac{1}{e_{2v}} \left(\delta_{j+1/2}[p_t] - g \bar{\rho}^{j+1/2} \delta_{j+1/2}[z_t] \right) v \, b_v \right\}$$

Using successively [equation 3.4](#), *i.e.* the skew symmetry property of the δ operator, [equation 5.3](#), the continuity equation, [equation 5.13](#), the hydrostatic equation in the s -coordinate, and $\delta_{k+1/2}[z_t] \equiv e_{3w}$, which comes from the definition of z_t , it becomes:

$$\equiv + \sum_{i,j,k} g \left\{ \bar{\rho}^{i+1/2} U \delta_{i+1/2}[z_t] + \bar{\rho}^{j+1/2} V \delta_{j+1/2}[z_t] + \left(\delta_i[U] + \delta_j[V] \right) \frac{p_t}{g} \right\} \\ \equiv + \sum_{i,j,k} g \left\{ \bar{\rho}^{i+1/2} U \delta_{i+1/2}[z_t] + \bar{\rho}^{j+1/2} V \delta_{j+1/2}[z_t] - \left(\frac{b_t}{e_{3t}} \partial_t(e_{3t}) + \delta_k[W] \right) \frac{p_t}{g} \right\} \\ \equiv + \sum_{i,j,k} g \left\{ \bar{\rho}^{i+1/2} U \delta_{i+1/2}[z_t] + \bar{\rho}^{j+1/2} V \delta_{j+1/2}[z_t] + \frac{W}{g} \delta_{k+1/2}[p_t] - \frac{p_t}{g} \partial_t b_t \right\} \\ \equiv + \sum_{i,j,k} g \left\{ \bar{\rho}^{i+1/2} U \delta_{i+1/2}[z_t] + \bar{\rho}^{j+1/2} V \delta_{j+1/2}[z_t] - W e_{3w} \bar{\rho}^{k+1/2} - \frac{p_t}{g} \partial_t b_t \right\} \\ \equiv + \sum_{i,j,k} g \left\{ \bar{\rho}^{i+1/2} U \delta_{i+1/2}[z_t] + \bar{\rho}^{j+1/2} V \delta_{j+1/2}[z_t] + W \bar{\rho}^{k+1/2} \delta_{k+1/2}[z_t] - \frac{p_t}{g} \partial_t b_t \right\} \\ \equiv - \sum_{i,j,k} g z_t \left\{ \delta_i \left[U \bar{\rho}^{i+1/2} \right] + \delta_j \left[V \bar{\rho}^{j+1/2} \right] + \delta_k \left[W \bar{\rho}^{k+1/2} \right] \right\} - \sum_{i,j,k} \left\{ p_t \partial_t b_t \right\} \\ \equiv + \sum_{i,j,k} g z_t \left\{ \partial_t(e_{3t} \rho) \right\} b_t - \sum_{i,j,k} \left\{ p_t \partial_t b_t \right\}$$

The first term is exactly the first term of the right-hand-side of [equation C.5](#). 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 [equation C.5](#). In other words, the following property must be satisfied:

$$\sum_{i,j,k} \left\{ p_t \partial_t b_t \right\} \equiv \sum_{i,j,k} \left\{ \rho g \partial_t(z_t) b_t \right\}$$

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 $p_w = \bar{p}_t^{k+1/2}$

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, [equation 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:

$$\begin{aligned} \sum_{i,j,k} \left\{ \frac{u}{e_{3u}} \frac{\partial(e_{3u}u)}{\partial t} b_u + \frac{v}{e_{3v}} \frac{\partial(e_{3v}v)}{\partial t} b_v \right\} - \frac{1}{2} \sum_{i,j,k} \left\{ \frac{u^2}{e_{3u}} \frac{\partial e_{3u}}{\partial t} b_u + \frac{v^2}{e_{3v}} \frac{\partial e_{3v}}{\partial t} b_v \right\} \\ = - \sum_{i,j,k} \left\{ \frac{1}{e_{3t}} \frac{\partial e_{3t} \rho}{\partial t} g z_t b_t \right\} - \sum_{i,j,k} \left\{ \rho g \frac{\partial z_t}{\partial t} b_t \right\} \end{aligned}$$

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:

$$f + \frac{1}{e_{1e_2}} \left(v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \equiv f + \frac{1}{e_{1f} e_{2f}} \left(\bar{v}^{i+1/2} \delta_{i+1/2} [e_{2u}] - \bar{u}^{j+1/2} \delta_{j+1/2} [e_{1u}] \right)$$

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 ([subsection 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:

$$- \int_D \mathbf{U}_h \cdot \left(\frac{\nabla \cdot (\mathbf{U}u)}{\nabla \cdot (\mathbf{U}v)} \right) dv - \frac{1}{2} \int_D \mathbf{U}_h^2 \frac{1}{e_3} \frac{\partial e_3}{\partial t} dv = 0 \quad (\text{C.8})$$

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):

$$\begin{aligned}
& - \int_D u \cdot \nabla \cdot (\mathbf{U} u) \, dv \\
\equiv & - \sum_{i,j,k} \left\{ \frac{1}{b_u} \left(\delta_{i+1/2} [\bar{U}^i \bar{u}^i] + \delta_j [\bar{V}^{i+1/2} \bar{u}^{j+1/2}] + \delta_k [\bar{W}^{i+1/2} \bar{u}^{k+1/2}] \right) \right\} b_u u \\
\equiv & - \sum_{i,j,k} \left\{ \delta_{i+1/2} [\bar{U}^i \bar{u}^i] + \delta_j [\bar{V}^{i+1/2} \bar{u}^{j+1/2}] + \delta_k [\bar{W}^{i+1/2} \bar{u}^{k+1/2}] \right\} u \\
\equiv & + \sum_{i,j,k} \left\{ \bar{U}^i \bar{u}^i \delta_i [u] + \bar{V}^{i+1/2} \bar{u}^{j+1/2} \delta_{j+1/2} [u] + \bar{W}^{i+1/2} \bar{u}^{k+1/2} \delta_{k+1/2} [u] \right\} \\
\equiv & + \frac{1}{2} \sum_{i,j,k} \left\{ \bar{U}^i \delta_i [u^2] + \bar{V}^{i+1/2} \delta_{j+1/2} [u^2] + \bar{W}^{i+1/2} \delta_{k+1/2} [u^2] \right\} \\
\equiv & - \sum_{i,j,k} \frac{1}{2} \left\{ U \delta_{i+1/2} [\bar{u}^2]^i + V \delta_{j+1/2} [\bar{u}^2]^j + W \delta_{k+1/2} [\bar{u}^2]^k \right\} \\
\equiv & - \sum_{i,j,k} \frac{1}{2} \bar{u}^2{}^i \left\{ \delta_{i+1/2} [U] + \delta_{j+1/2} [V] + \delta_{k+1/2} [W] \right\} \\
\equiv & + \sum_{i,j,k} \frac{1}{2} \bar{u}^2{}^i \left\{ \left(\frac{1}{e_{3t}} \frac{\partial e_{3t}}{\partial t} \right) b_t \right\}
\end{aligned}$$

Applying similar manipulation applied to the second term of the scalar product leads to:

$$- \int_D \mathbf{U}_h \cdot \left(\begin{array}{c} \nabla \cdot (\mathbf{U} u) \\ \nabla \cdot (\mathbf{U} v) \end{array} \right) dv \equiv + \sum_{i,j,k} \frac{1}{2} (\bar{u}^2{}^i + \bar{v}^2{}^j) \left\{ \left(\frac{1}{e_{3t}} \frac{\partial e_{3t}}{\partial t} \right) b_t \right\}$$

which is the discrete form of $\frac{1}{2} \int_D u \cdot \nabla \cdot (\mathbf{U} u) \, dv$. [equation C.8](#) 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 discretized as follows:

$$\left\{ \begin{array}{l} + \frac{1}{e_{1u}} \bar{q}^i \quad \overline{\overline{(e_{1v} e_{3v} v)}}^{i,j+1/2} \\ - \frac{1}{e_{2v}} \bar{q}^j \quad \overline{\overline{(e_{2u} e_{3u} u)}}^{i+1/2,j} \end{array} \right. \quad (\text{C.9})$$

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 $\chi=0$). Indeed, using the symmetry or skew symmetry properties of the operators ([equation 3.5](#) and [equation 3.4](#)), it can be shown that:

$$\int_D q \mathbf{k} \cdot \frac{1}{e_3} \nabla \times (e_3 q \mathbf{k} \times \mathbf{U}_h) \, dv \equiv 0 \quad (\text{C.10})$$

where $dv = e_1 e_2 e_3 \, di \, dj \, dk$ is the volume element. Indeed, using [equation 5.4](#), the discrete form of the right hand side

of equation C.10 can be transformed as follow:

$$\begin{aligned}
 & \int_D q \mathbf{k} \cdot \frac{1}{e_3} \nabla \times (e_3 q \mathbf{k} \times \mathbf{U}_h) dv \\
 & \equiv \sum_{i,j,k} q \left\{ \delta_{i+1/2} \left[-\bar{q}^i \bar{U}^{i,j+1/2} \right] - \delta_{j+1/2} \left[\bar{q}^j \bar{V}^{i+1/2,j} \right] \right\} \\
 & \equiv \sum_{i,j,k} \left\{ \delta_i [q] \bar{q}^i \bar{U}^{i,j+1/2} + \delta_j [q] \bar{q}^j \bar{V}^{i+1/2,j} \right\} \\
 & \equiv \frac{1}{2} \sum_{i,j,k} \left\{ \delta_i [q^2] \bar{U}^{i,j+1/2} + \delta_j [q^2] \bar{V}^{i+1/2,j} \right\} \\
 & \equiv -\frac{1}{2} \sum_{i,j,k} q^2 \left\{ \delta_{i+1/2} \left[\bar{U}^{i,j+1/2} \right] + \delta_{j+1/2} \left[\bar{V}^{i+1/2,j} \right] \right\}
 \end{aligned}$$

Since $\bar{\cdot}$ and δ operators commute: $\delta_{i+1/2} [\bar{a}^i] = \bar{\delta}_i [a]^{i+1/2}$, and introducing the horizontal divergence χ , it becomes:

$$\equiv \sum_{i,j,k} -\frac{1}{2} q^2 \overline{\overline{e_{1t} e_{2t} e_{3t} \chi}}^{i+1/2, j+1/2} \equiv 0$$

The later equality is obtain only when the flow is horizontally non-divergent, *i.e.* $\chi=0$.

Vorticity Term with EEN scheme (`ln_dynvor_een=.true.`)

With the EEN scheme, the vorticity terms are represented as:

$$\begin{cases} +q e_3 v \equiv +\frac{1}{e_{1u}} \sum_{i_p, k_p}^{i+1/2-i_p} \mathbb{Q}_{j_p}^{i_p} (e_{1v} e_{3v} v)_{j+j_p}^{i+i_p-1/2} \\ -q e_3 u \equiv -\frac{1}{e_{2v}} \sum_{i_p, k_p}^{i+1/2-j_p} \mathbb{Q}_{j_p}^{i_p} (e_{2u} e_{3u} u)_{j+j_p-1/2}^{i+i_p} \end{cases} \quad (\text{C.11})$$

where the indices i_p and k_p take the following values: $i_p = -1/2$ or $1/2$ and $j_p = -1/2$ or $1/2$, and the vorticity triads, ${}^i \mathbb{Q}_{j_p}^{i_p}$, defined at T -point, are given by:

$${}^j \mathbb{Q}_{j_p}^{i_p} = \frac{1}{12} \left(q_{j+j_p}^{i-i_p} + q_{j+i_p}^{i+j_p} + q_{j-j_p}^{i+i_p} \right) \quad (\text{C.7})$$

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 ${}^i \mathbb{Q}_{+1/2}^{+1/2}$, similar manipulation can be done for the 3 others. The discrete form of the right hand side of equation C.10 applied to this triad only can be transformed as follow:

$$\begin{aligned}
 & \int_D q \mathbf{k} \cdot \frac{1}{e_3} \nabla \times (e_3 q \mathbf{k} \times \mathbf{U}_h) dv \\
 & \equiv \sum_{i,j,k} q \left\{ \delta_{i+1/2} \left[-{}^i \mathbb{Q}_{+1/2}^{+1/2} U_j^{i+1/2} \right] - \delta_{j+1/2} \left[{}^i \mathbb{Q}_{+1/2}^{+1/2} V_{j+1/2}^i \right] \right\} \\
 & \equiv \sum_{i,j,k} \left\{ \delta_i [q] {}^i \mathbb{Q}_{+1/2}^{+1/2} U_j^{i+1/2} + \delta_j [q] {}^i \mathbb{Q}_{+1/2}^{+1/2} V_{j+1/2}^i \right\} \\
 & \dots \\
 & \text{Demonstration to be done...} \\
 & \dots \\
 & \equiv \frac{1}{2} \sum_{i,j,k} \left\{ \delta_i \left[\left({}^i \mathbb{Q}_{+1/2}^{+1/2} \right)^2 \right] \bar{U}^{i,j+1/2} + \delta_j \left[\left({}^i \mathbb{Q}_{+1/2}^{+1/2} \right)^2 \right] \bar{V}^{i+1/2,j} \right\} \\
 & \equiv -\frac{1}{2} \sum_{i,j,k} \left({}^i \mathbb{Q}_{+1/2}^{+1/2} \right)^2 \left\{ \delta_{i+1/2} \left[\bar{U}^{i,j+1/2} \right] + \delta_{j+1/2} \left[\bar{V}^{i+1/2,j} \right] \right\} \\
 & \equiv \sum_{i,j,k} -\frac{1}{2} \left({}^i \mathbb{Q}_{+1/2}^{+1/2} \right)^2 \overline{\overline{b_t \chi}}^{i+1/2, j+1/2} \\
 & \equiv 0
 \end{aligned}$$

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.* 2nd 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 :

$$\frac{\partial}{\partial t} \left(\int_D T \, dv \right) = \int_D \frac{1}{e_3} \frac{\partial (e_3 T)}{\partial t} \, dv = 0$$

conservation of its variance:

$$\frac{\partial}{\partial t} \left(\int_D \frac{1}{2} T^2 \, dv \right) = \int_D \frac{1}{e_3} Q \frac{\partial (e_3 T)}{\partial t} \, dv - \frac{1}{2} \int_D T^2 \frac{1}{e_3} \frac{\partial e_3}{\partial t} \, dv$$

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 its τ_u , τ_v , and τ_w interpolated values at velocity point (whatever the interpolation is), the conservation of the tracer content due to the advection tendency is obtained as follows:

$$\begin{aligned} & \int_D \frac{1}{e_3} \frac{\partial (e_3 T)}{\partial t} \, dv = - \int_D \nabla \cdot (T \mathbf{U}) \, dv \\ & \equiv - \sum_{i,j,k} \left\{ \frac{1}{b_t} (\delta_i [U \tau_u] + \delta_j [V \tau_v]) + \frac{1}{e_{3t}} \delta_k [w \tau_w] \right\} b_t \\ & \equiv - \sum_{i,j,k} \{ \delta_i [U \tau_u] + \delta_j [V \tau_v] + \delta_k [W \tau_w] \} \\ & \equiv 0 \end{aligned}$$

The conservation of the variance of tracer due to the advection tendency can be achieved only with the CEN2 scheme, *i.e.* when $\tau_u = \bar{T}^{i+1/2}$, $\tau_v = \bar{T}^{j+1/2}$, and $\tau_w = \bar{T}^{k+1/2}$. It can be demonstrated as follows:

$$\begin{aligned} & \int_D \frac{1}{e_3} Q \frac{\partial (e_3 T)}{\partial t} \, dv = - \int_D \tau \nabla \cdot (T \mathbf{U}) \, dv \\ & \equiv - \sum_{i,j,k} T \left\{ \delta_i [U \bar{T}^{i+1/2}] + \delta_j [V \bar{T}^{j+1/2}] + \delta_k [W \bar{T}^{k+1/2}] \right\} \\ & \equiv + \sum_{i,j,k} \left\{ U \bar{T}^{i+1/2} \delta_{i+1/2} [T] + V \bar{T}^{j+1/2} \delta_{j+1/2} [T] + W \bar{T}^{k+1/2} \delta_{k+1/2} [T] \right\} \\ & \equiv + \frac{1}{2} \sum_{i,j,k} \left\{ U \delta_{i+1/2} [T^2] + V \delta_{j+1/2} [T^2] + W \delta_{k+1/2} [T^2] \right\} \\ & \equiv - \frac{1}{2} \sum_{i,j,k} T^2 \left\{ \delta_i [U] + \delta_j [V] + \delta_k [W] \right\} \\ & \equiv + \frac{1}{2} \sum_{i,j,k} T^2 \left\{ \frac{1}{e_{3t}} \frac{\partial e_{3t}}{\partial t} \right\} \end{aligned}$$

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 [equation 3.2](#) and [equation 3.3](#). 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:

$$\begin{aligned} & \int_D \frac{1}{e_3} \mathbf{k} \cdot \nabla \times \left[\nabla_h (A^{lm} \chi) - \nabla_h \times (A^{lm} \zeta \mathbf{k}) \right] dv \\ &= \int_D -\frac{1}{e_3} \mathbf{k} \cdot \nabla \times \left[\nabla_h \times (A^{lm} \zeta \mathbf{k}) \right] dv \\ &\equiv \sum_{i,j} \left\{ \delta_{i+1/2} \left[\frac{e_{2v}}{e_{1v} e_{3v}} \delta_i [A_f^{lm} e_{3f} \zeta] \right] + \delta_{j+1/2} \left[\frac{e_{1u}}{e_{2u} e_{3u}} \delta_j [A_f^{lm} e_{3f} \zeta] \right] \right\} \end{aligned}$$

Using [equation 3.4](#), it follows:

$$\equiv \sum_{i,j,k} - \left\{ \frac{e_{2v}}{e_{1v} e_{3v}} \delta_i [A_f^{lm} e_{3f} \zeta] \delta_i [1] + \frac{e_{1u}}{e_{2u} e_{3u}} \delta_j [A_f^{lm} e_{3f} \zeta] \delta_j [1] \right\} \equiv 0$$

C.7.2. Dissipation of horizontal kinetic energy

The lateral momentum diffusion term dissipates the horizontal kinetic energy:

$$\begin{aligned} & \int_D \mathbf{U}_h \cdot \left[\nabla_h (A^{lm} \chi) - \nabla_h \times (A^{lm} \zeta \mathbf{k}) \right] dv \\ &\equiv \sum_{i,j,k} \left\{ \frac{1}{e_{1u}} \delta_{i+1/2} [A_T^{lm} \chi] - \frac{1}{e_{2u} e_{3u}} \delta_j [A_f^{lm} e_{3f} \zeta] \right\} e_{1u} e_{2u} e_{3u} u \\ &\quad + \left\{ \frac{1}{e_{2u}} \delta_{j+1/2} [A_T^{lm} \chi] + \frac{1}{e_{1v} e_{3v}} \delta_i [A_f^{lm} e_{3f} \zeta] \right\} e_{1v} e_{2u} e_{3v} v \\ &\equiv \sum_{i,j,k} \left\{ e_{2u} e_{3u} u \delta_{i+1/2} [A_T^{lm} \chi] - e_{1u} u \delta_j [A_f^{lm} e_{3f} \zeta] \right\} \\ &\quad + \left\{ e_{1v} e_{3v} v \delta_{j+1/2} [A_T^{lm} \chi] + e_{2v} v \delta_i [A_f^{lm} e_{3f} \zeta] \right\} \\ &\equiv \sum_{i,j,k} - \left(\delta_i [e_{2u} e_{3u} u] + \delta_j [e_{1v} e_{3v} v] \right) A_T^{lm} \chi \\ &\quad - \left(\delta_{i+1/2} [e_{2v} v] - \delta_{j+1/2} [e_{1u} u] \right) A_f^{lm} e_{3f} \zeta \\ &\equiv \sum_{i,j,k} -A_T^{lm} \chi^2 e_{1t} e_{2t} e_{3t} - A_f^{lm} \zeta^2 e_{1f} e_{2f} e_{3f} \leq 0 \end{aligned}$$

C.7.3. Dissipation of enstrophy

The lateral momentum diffusion term dissipates the enstrophy when the eddy coefficients are horizontally uniform:

$$\begin{aligned}
 & \int_D \zeta \mathbf{k} \cdot \nabla \times [\nabla_h (A^{lm} \chi) - \nabla_h \times (A^{lm} \zeta \mathbf{k})] dv \\
 &= A^{lm} \int_D \zeta \mathbf{k} \cdot \nabla \times [\nabla_h \times (\zeta \mathbf{k})] dv \\
 &\equiv A^{lm} \sum_{i,j,k} \zeta e_{3f} \left\{ \delta_{i+1/2} \left[\frac{e_{2v}}{e_{1v} e_{3v}} \delta_i [e_{3f} \zeta] \right] + \delta_{j+1/2} \left[\frac{e_{1u}}{e_{2u} e_{3u}} \delta_j [e_{3f} \zeta] \right] \right\}
 \end{aligned}$$

Using equation 3.4, it follows:

$$\equiv -A^{lm} \sum_{i,j,k} \left\{ \left(\frac{1}{e_{1v} e_{3v}} \delta_i [e_{3f} \zeta] \right)^2 b_v + \left(\frac{1}{e_{2u} e_{3u}} \delta_j [e_{3f} \zeta] \right)^2 b_u \right\} \leq 0$$

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 equation 3.3. The resulting term conserves the χ and dissipates χ^2 when the eddy coefficients are horizontally uniform.

$$\begin{aligned}
 & \int_D \nabla_h \cdot [\nabla_h (A^{lm} \chi) - \nabla_h \times (A^{lm} \zeta \mathbf{k})] dv = \int_D \nabla_h \cdot \nabla_h (A^{lm} \chi) dv \\
 &\equiv \sum_{i,j,k} \left\{ \delta_i \left[A_u^{lm} \frac{e_{2u} e_{3u}}{e_{1u}} \delta_{i+1/2} [\chi] \right] + \delta_j \left[A_v^{lm} \frac{e_{1v} e_{3v}}{e_{2v}} \delta_{j+1/2} [\chi] \right] \right\}
 \end{aligned}$$

Using equation 3.4, it follows:

$$\equiv \sum_{i,j,k} - \left\{ \frac{e_{2u} e_{3u}}{e_{1u}} A_u^{lm} \delta_{i+1/2} [\chi] \delta_{i+1/2} [1] + \frac{e_{1v} e_{3v}}{e_{2v}} A_v^{lm} \delta_{j+1/2} [\chi] \delta_{j+1/2} [1] \right\} \equiv 0$$

C.7.5. Dissipation of horizontal divergence variance

$$\begin{aligned}
 & \int_D \chi \nabla_h \cdot [\nabla_h (A^{lm} \chi) - \nabla_h \times (A^{lm} \zeta \mathbf{k})] dv = A^{lm} \int_D \chi \nabla_h \cdot \nabla_h (\chi) dv \\
 &\equiv A^{lm} \sum_{i,j,k} \frac{1}{e_{1t} e_{2t} e_{3t}} \chi \left\{ \delta_i \left[\frac{e_{2u} e_{3u}}{e_{1u}} \delta_{i+1/2} [\chi] \right] + \delta_j \left[\frac{e_{1v} e_{3v}}{e_{2v}} \delta_{j+1/2} [\chi] \right] \right\} e_{1t} e_{2t} e_{3t}
 \end{aligned}$$

Using equation 3.4, it turns out to be:

$$\equiv -A^{lm} \sum_{i,j,k} \left\{ \left(\frac{1}{e_{1u}} \delta_{i+1/2} [\chi] \right)^2 b_u + \left(\frac{1}{e_{2v}} \delta_{j+1/2} [\chi] \right)^2 b_v \right\} \leq 0$$

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:

$$\int_D \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) dv = \vec{\mathbf{0}}$$

and

$$\int_D \mathbf{U}_h \cdot \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) dv \leq 0$$

The first property is obvious. The second results from:

$$\begin{aligned} & \int_D \mathbf{U}_h \cdot \frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) dv \\ & \equiv \sum_{i,j,k} \left(u \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2} [u] \right] e_{1u} e_{2u} + v \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2} [v] \right] e_{1v} e_{2v} \right) \end{aligned}$$

since the horizontal scale factor does not depend on k , it follows:

$$\equiv - \sum_{i,j,k} \left(\frac{A_u^{vm}}{e_{3uw}} (\delta_{k+1/2} [u])^2 e_{1u} e_{2u} + \frac{A_v^{vm}}{e_{3vw}} (\delta_{k+1/2} [v])^2 e_{1v} e_{2v} \right) \leq 0$$

The vorticity is also conserved. Indeed:

$$\begin{aligned} & \int_D \frac{1}{e_3} \mathbf{k} \cdot \nabla \times \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) dv \\ & \equiv \sum_{i,j,k} \frac{1}{e_{3f}} \frac{1}{e_{1f} e_{2f}} \left\{ \delta_{i+1/2} \left(\frac{e_{2v}}{e_{3v}} \delta_k \left[\frac{1}{e_{3vw}} \delta_{k+1/2} [v] \right] \right) \right. \\ & \quad \left. - \delta_{j+1/2} \left(\frac{e_{1u}}{e_{3u}} \delta_k \left[\frac{1}{e_{3uw}} \delta_{k+1/2} [u] \right] \right) \right\} e_{1f} e_{2f} e_{3f} \equiv 0 \end{aligned}$$

If the vertical diffusion coefficient is uniform over the whole domain, the enstrophy is dissipated, *i.e.*

$$\int_D \zeta \mathbf{k} \cdot \nabla \times \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) dv = 0$$

This property is only satisfied in z -coordinates:

$$\begin{aligned} & \int_D \zeta \mathbf{k} \cdot \nabla \times \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) dv \\ & \equiv \sum_{i,j,k} \zeta e_{3f} \left\{ \delta_{i+1/2} \left(\frac{e_{2v}}{e_{3v}} \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2} [v] \right] \right) \right. \\ & \quad \left. - \delta_{j+1/2} \left(\frac{e_{1u}}{e_{3u}} \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2} [u] \right] \right) \right\} \\ & \equiv \sum_{i,j,k} \zeta e_{3f} \left\{ \frac{1}{e_{3v}} \delta_k \left[\frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2} [\delta_{i+1/2} [e_{2v} v]] \right] \right. \\ & \quad \left. - \frac{1}{e_{3u}} \delta_k \left[\frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2} [\delta_{j+1/2} [e_{1u} u]] \right] \right\} \end{aligned}$$

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:

$$\equiv A^{vm} \sum_{i,j,k} \zeta \delta_k \left[\frac{1}{e_{3w}} \delta_{k+1/2} [\delta_{i+1/2} [e_{2v} v] - \delta_{j+1/2} [e_{1u} u]] \right]$$

$$\equiv -A^{vm} \sum_{i,j,k} \frac{1}{e_{3w}} (\delta_{k+1/2} [\zeta])^2 e_{1f} e_{2f} \leq 0$$

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:

$$\begin{aligned} & \int_D \chi \nabla \cdot \left(\frac{1}{e_3} \frac{\partial}{\partial k} \left(\frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) dv \\ & \equiv \sum_{i,j,k} \frac{\chi}{e_{1t} e_{2t}} \left\{ \delta_{i+1/2} \left(\frac{e_{2u}}{e_{3u}} \delta_k \left[\frac{A^{vm}}{e_{3uw}} \delta_{k+1/2} [u] \right] \right) \right. \\ & \quad \left. + \delta_{j+1/2} \left(\frac{e_{1v}}{e_{3v}} \delta_k \left[\frac{A^{vm}}{e_{3vw}} \delta_{k+1/2} [v] \right] \right) \right\} e_{1t} e_{2t} e_{3t} \\ & \equiv A^{vm} \sum_{i,j,k} \chi \left\{ \delta_{i+1/2} \left(\delta_k \left[\frac{1}{e_{3uw}} \delta_{k+1/2} [e_{2u} u] \right] \right) \right. \\ & \quad \left. + \delta_{j+1/2} \left(\delta_k \left[\frac{1}{e_{3vw}} \delta_{k+1/2} [e_{1v} v] \right] \right) \right\} \\ & \equiv -A^{vm} \sum_{i,j,k} \frac{\delta_{k+1/2} [\chi]}{e_{3w}} \left\{ \delta_{k+1/2} [\delta_{i+1/2} [e_{2u} u] + \delta_{j+1/2} [e_{1v} v]] \right\} \\ & \equiv -A^{vm} \sum_{i,j,k} \frac{1}{e_{3w}} \delta_{k+1/2} [\chi] \delta_{k+1/2} [e_{1t} e_{2t} \chi] \\ & \equiv -A^{vm} \sum_{i,j,k} \frac{e_{1t} e_{2t}}{e_{3w}} (\delta_{k+1/2} [\chi])^2 \equiv 0 \end{aligned}$$

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). 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 conservation of mass only if the Equation Of Seawater is linear.

C.9.1. Conservation of tracers

constraint of conservation of tracers:

$$\begin{aligned} & \int_D \nabla \cdot (A \nabla T) dv \\ & \equiv \sum_{i,j,k} \left\{ \delta_i \left[A_u^{iT} \frac{e_{2u} e_{3u}}{e_{1u}} \delta_{i+1/2} [T] \right] + \delta_j \left[A_v^{iT} \frac{e_{1v} e_{3v}}{e_{2v}} \delta_{j+1/2} [T] \right] \right. \\ & \quad \left. + \delta_k \left[A_w^{iT} \frac{e_{1t} e_{2t}}{e_{3t}} \delta_{k+1/2} [T] \right] \right\} \equiv 0 \end{aligned}$$

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:

$$\begin{aligned}
 & \int_D T \nabla \cdot (A \nabla T) \, dv \\
 & \equiv \sum_{i,j,k} T \left\{ \delta_i \left[A_u^{lT} \frac{e_{2u} e_{3u}}{e_{1u}} \delta_{i+1/2} [T] \right] \right. \\
 & \qquad \qquad \qquad \left. + \delta_j \left[A_v^{lT} \frac{e_{1v} e_{3v}}{e_{2v}} \delta_{j+1/2} [T] \right] \right. \\
 & \qquad \qquad \qquad \left. + \delta_k \left[A_w^{vT} \frac{e_{1t} e_{2t}}{e_{3t}} \delta_{k+1/2} [T] \right] \right\} \\
 & \equiv - \sum_{i,j,k} \left\{ A_u^{lT} \left(\frac{1}{e_{1u}} \delta_{i+1/2} [T] \right)^2 e_{1u} e_{2u} e_{3u} \right. \\
 & \qquad \qquad \qquad + A_v^{lT} \left(\frac{1}{e_{2v}} \delta_{j+1/2} [T] \right)^2 e_{1v} e_{2v} e_{3v} \\
 & \qquad \qquad \qquad \left. + A_w^{vT} \left(\frac{1}{e_{3w}} \delta_{k+1/2} [T] \right)^2 e_{1w} e_{2w} e_{3w} \right\} \leq 0
 \end{aligned}$$



Iso-Neutral Diffusion and Eddy Advection using Triads

Table of contents

D.1. Choice of <code>namtra</code> and <code>ldf</code> namelist parameters	227
D.2. Triad formulation of iso-neutral diffusion	227
D.2.1. Iso-neutral diffusion operator	227
D.2.2. Standard discretization	228
D.2.3. Expression of the skew-flux in terms of triad slopes	228
D.2.4. Full triad fluxes	229
D.2.5. Ensuring the scheme does not increase tracer variance	230
D.2.6. Triad volumes in Griffes's scheme and in <i>NEMO</i>	231
D.2.7. Summary of the scheme	231
D.2.8. Treatment of the triads at the boundaries	233
D.2.9. Limiting of the slopes within the interior	233
D.2.10. Tapering within the surface mixed layer	234
D.3. Eddy induced advection formulated as a skew flux	235
D.3.1. Continuous skew flux formulation	235
D.3.2. Discrete skew flux formulation	236
D.3.3. Treatment of the triads at the boundaries	237
D.3.4. Limiting of the slopes within the interior	238
D.3.5. Tapering within the surface mixed layer	238
D.3.6. Streamfunction diagnostics	238

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4

D.1. Choice of `&namtra_ldf` (namelist 4.2) namelist parameters

Two scheme are available to perform the iso-neutral diffusion. If the namelist logical `ln_traldf_triad` is set true, *NEMO* updates both active and passive tracers using the Griffies triad representation of iso-neutral diffusion and the eddy-induced advective skew (GM) fluxes. If the namelist logical `ln_traldf_iso` is set true, the filtered version of Cox's original scheme (the Standard scheme) is employed (section 8.2). In the present implementation of the Griffies scheme, the advective skew fluxes are implemented even if `ln_traldf_eiv` is false.

Values of iso-neutral diffusivity and GM coefficient are set as described in section 8.3. Note that when GM fluxes are used, the eddy-advective (GM) velocities are output for diagnostic purposes using XIOS, even though the eddy advection is accomplished by means of the skew fluxes.

The options specific to the Griffies scheme include:

ln_triad_iso See subsection D.2.10. If this is set false (the default), then 'iso-neutral' mixing is accomplished within the surface mixed-layer along slopes linearly decreasing with depth from the value immediately below the mixed-layer to zero (flat) at the surface (section D.2.10). This is the same treatment as used in the default implementation subsection 8.2.2; figure 8.2. Where `ln_triad_iso` is set true, the vertical skew flux is further reduced to ensure no vertical buoyancy flux, giving an almost pure horizontal diffusive tracer flux within the mixed layer. This is similar to the tapering suggested by Gerdes et al. (1991). See section D.2.10

ln_botmix_triad See subsection D.2.8. If this is set false (the default) then the lateral diffusive fluxes associated with triads partly masked by topography are neglected. If it is set true, however, then these lateral diffusive fluxes are applied, giving smoother bottom tracer fields at the cost of introducing diapycnal mixing.

rn_sw_triad blah blah to be added....

The options shared with the Standard scheme include:

ln_traldf_msc blah blah to be added

rn_slpmax blah blah to be added

D.2. Triad formulation of iso-neutral diffusion

We have implemented into *NEMO* a scheme inspired by Griffies et al. (1998), but formulated within the *NEMO* framework, using scale factors rather than grid-sizes.

D.2.1. Iso-neutral diffusion operator

The iso-neutral second order tracer diffusive operator for small angles between iso-neutral surfaces and geopotentials is given by equation D.1:

$$D^{IT} = -\nabla \cdot f^{IT} \equiv -\frac{1}{e_1 e_2 e_3} \left[\frac{\partial}{\partial i} (f_1^{IT} e_2 e_3) + \frac{\partial}{\partial j} (f_2^{IT} e_2 e_3) + \frac{\partial}{\partial k} (f_3^{IT} e_1 e_2) \right], \quad (\text{D.1a})$$

where the diffusive flux per unit area of physical space

$$f^{IT} = -A^{IT} \mathfrak{R} \cdot \nabla T, \quad (\text{D.1b})$$

$$\text{with } \mathfrak{R} = \begin{pmatrix} 1 & 0 & -r_1 \\ 0 & 1 & -r_2 \\ -r_1 & -r_2 & r_1^2 + r_2^2 \end{pmatrix} \quad \text{and} \quad \nabla T = \begin{pmatrix} \frac{1}{e_1} \frac{\partial T}{\partial i} \\ \frac{1}{e_2} \frac{\partial T}{\partial j} \\ \frac{1}{e_3} \frac{\partial T}{\partial k} \end{pmatrix}. \quad (\text{D.1c})$$

Here equation 1.19

$$\begin{aligned} r_1 &= -\frac{e_3}{e_1} \left(\frac{\partial \rho}{\partial i} \right) \left(\frac{\partial \rho}{\partial k} \right)^{-1} \\ &= -\frac{e_3}{e_1} \left(-\alpha \frac{\partial T}{\partial i} + \beta \frac{\partial S}{\partial i} \right) \left(-\alpha \frac{\partial T}{\partial k} + \beta \frac{\partial S}{\partial k} \right)^{-1} \end{aligned}$$

is the *i*-component of the slope of the iso-neutral surface relative to the computational surface, and *r*₂ is the *j*-component.

We will find it useful to consider the fluxes per unit area in i, j, k space; we write

$$F_{\text{iso}} = (f_1^{lT} e_2 e_3, f_2^{lT} e_1 e_3, f_3^{lT} e_1 e_2).$$

Additionally, we will sometimes write the contributions towards the fluxes f and F_{iso} from the component R_{ij} of \mathfrak{R} as f_{ij} , $F_{\text{iso } ij}$, with $f_{ij} = R_{ij} e_i^{-1} \partial T / \partial x_i$ (no summation) etc.

The off-diagonal terms of the small angle diffusion tensor [equation D.1](#), [equation D.1c](#) produce skew-fluxes along the i - and j -directions resulting from the vertical tracer gradient:

$$f_{13} = + A^{lT} r_1 \frac{1}{e_3} \frac{\partial T}{\partial k}, \quad f_{23} = + A^{lT} r_2 \frac{1}{e_3} \frac{\partial T}{\partial k} \quad (\text{D.2})$$

and in the k -direction resulting from the lateral tracer gradients

$$f_{31} + f_{32} = A^{lT} r_1 \frac{1}{e_1} \frac{\partial T}{\partial i} + A^{lT} r_2 \frac{1}{e_1} \frac{\partial T}{\partial i} \quad (\text{D.3})$$

The vertical diffusive flux associated with the 33 component of the small angle diffusion tensor is

$$f_{33} = -A^{lT} (r_1^2 + r_2^2) \frac{1}{e_3} \frac{\partial T}{\partial k}. \quad (\text{D.4})$$

Since there are no cross terms involving r_1 and r_2 in the above, we can consider the iso-neutral 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, [equation D.2](#), 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, [equation D.3](#), is evaluated at w -points but involves horizontal gradients defined at u -points.

D.2.2. Standard discretization

The straightforward approach to discretize the lateral skew flux [equation D.2](#) from tracer cell i, k to $i + 1, k$, introduced in 1995 into OPA, [equation 4.8](#), 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 (flux per unit area in ijk space) from tracer cell i, k to $i + 1, k$, noting that the $e_{3i+1/2}^k$ in the area $e_{3i+1/2}^k e_{2i+1/2} i^k$ at the u -point cancels out with the $1/e_{3i+1/2}^k$ associated with the vertical tracer gradient, is then [equation 4.8](#)

$$(F_u^{13})_{i+\frac{1}{2}}^k = A_{i+\frac{1}{2}}^k e_{2i+1/2}^k \bar{r}_1^{i,k} \overline{\delta_k T}^{i,k},$$

where

$$\bar{r}_1^{i,k} = - \frac{e_{3u_{i+1/2}}^k \delta_{i+1/2}[\rho]}{e_{1u_{i+1/2}}^k \overline{\delta_k \rho}^{i,k}},$$

and here and in the following we drop the lT superscript from A^{lT} for simplicity. Unfortunately the resulting combination $\overline{\delta_k \bullet}^{i,k}$ of a k average and a k difference of the tracer 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 [chapter 8](#)). This technique works for T and S in so far as they are active tracers (*i.e.* they enter the computation of density), but it does not work for a passive tracer.

D.2.3. Expression of the skew-flux in terms of triad slopes

([Griffies et al., 1998](#)) 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 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 [figure D.1a](#), where the thick lines denote the tracer gradients, and the thin lines the corresponding triads, with slopes s_1, \dots, s_4 . The total area-integrated skew-flux from tracer cell i, k to $i + 1, k$

$$(F_u^{13})_{i+\frac{1}{2}}^k = A_{i+1}^k a_1 s_1 \delta_{k+\frac{1}{2}} [T^{i+1}] / e_{3w_{i+1}}^{k+\frac{1}{2}} + A_i^k a_2 s_2 \delta_{k+\frac{1}{2}} [T^i] / e_{3w_{i+1}}^{k+\frac{1}{2}} \\ + A_{i+1}^k a_3 s_3 \delta_{k-\frac{1}{2}} [T^{i+1}] / e_{3w_{i+1}}^{k+\frac{1}{2}} + A_i^k a_4 s_4 \delta_{k-\frac{1}{2}} [T^i] / e_{3w_{i+1}}^{k+\frac{1}{2}}, \quad (\text{D.5})$$

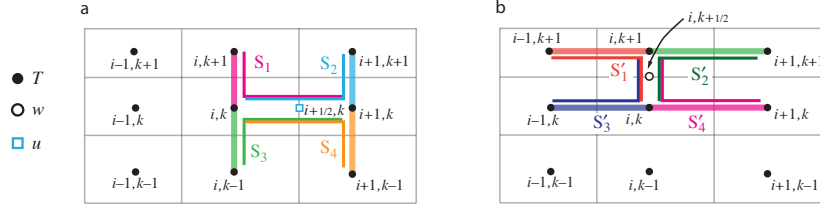


Figure D.1.: (a) Arrangement of triads S_i and tracer gradients to give lateral tracer flux from box i, k to $i + 1, k$ (b) Triads S'_i and tracer gradients to give vertical tracer flux from box i, k to $i, k + 1$.

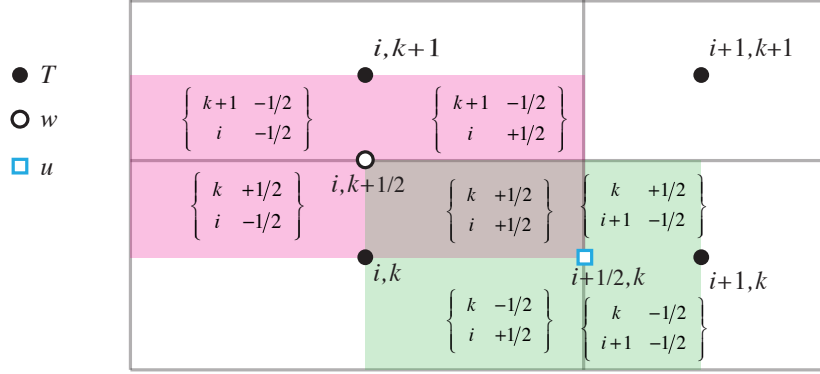


Figure D.2.: Triad notation for quarter cells. T -cells are inside boxes, while the $i + \frac{1}{2}, k$ u -cell is shaded in green and the $i, k + \frac{1}{2}$ w -cell is shaded in pink.

where the contributions of the triad fluxes are weighted by areas a_1, \dots, a_4 , and A 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 equation D.3 from tracer cell i, k to $i, k + 1$ at the w -point $i, k + \frac{1}{2}$ is constructed similarly (figure D.1b) by multiplying lateral tracer gradients from each of the four surrounding u -points by the appropriate triad slope:

$$(F_w^{31})_i^{k+\frac{1}{2}} = A_i^{k+1} a'_1 s'_1 \delta_{i-\frac{1}{2}} [T^{k+1}] / e_{3u_{i-\frac{1}{2}}^{k+1}} + A_i^{k+1} a'_2 s'_2 \delta_{i+\frac{1}{2}} [T^{k+1}] / e_{3u_{i+\frac{1}{2}}^{k+1}} \\ + A_i^k a'_3 s'_3 \delta_{i-\frac{1}{2}} [T^k] / e_{3u_{i-\frac{1}{2}}^k} + A_i^k a'_4 s'_4 \delta_{i+\frac{1}{2}} [T^k] / e_{3u_{i+\frac{1}{2}}^k}. \quad (\text{D.6})$$

We notate the triad slopes s_i and s'_i 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 figure D.1):

$${}^k \mathbb{R}_{i_p}^{k_p} = - \frac{e_{3w_{i+k_p}}^{k+k_p}}{e_{1u_{i+i_p}}^k} \frac{\alpha_i^k \delta_{i+i_p} [T^k] - \beta_i^k \delta_{i+i_p} [S^k]}{\alpha_i^k \delta_{k+k_p} [T^i] - \beta_i^k \delta_{k+k_p} [S^i]}. \quad (\text{D.7})$$

In calculating the slopes of the local neutral surfaces, the expansion coefficients α and β are evaluated at the anchor points of the triad, while the metrics are calculated at the u - and w -points on the arms.

Each triad $\{i, i_p, k, k_p\}$ is associated (figure D.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 s'_i in equation D.5 and equation D.6 in this notation, we have e.g. $s_1 = s'_1 = {}^k \mathbb{R}_{1/2}^{1/2}$. Each triad slope ${}^k \mathbb{R}_{i_p}^{k_p}$ 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 notate these areas, similarly to the triad slopes, as ${}^k \mathbb{A}_{u_{i_p}}^{k_p}$, ${}^k \mathbb{A}_{w_{i_p}}^{k_p}$, where e.g. in equation D.5 $a_1 = {}^k \mathbb{A}_{u_{1/2}}^{1/2}$, and in equation D.6 $a'_1 = {}^k \mathbb{A}_{w_{1/2}}^{1/2}$.

D.2.4. Full triad fluxes

A key property of iso-neutral 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

$$(F_u^{11})_{i+\frac{1}{2}}^k = - (A_i^{k+1} a_1 + A_i^{k+1} a_2 + A_i^k a_3 + A_i^k a_4) \frac{\delta_{i+1/2} [T^k]}{e_{1u_{i+1/2}}^k}, \quad (\text{D.8})$$

where the areas a_i are as in [equation D.5](#). In this case, separating the total lateral flux, the sum of [equation D.5](#) and [equation D.8](#), into triad components, a lateral tracer flux

$${}^k\mathbb{F}_{u_{i_p}}^{k_p}(T) = -A_i^k {}^k\mathbb{A}_{u_{i_p}}^{k_p} \left(\frac{\delta_{i+i_p}[T^k]}{e_{1u}^k} - {}^k\mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w}^i} \right) \quad (\text{D.9})$$

can be identified with each triad. Then, because the same metric factors $e_{3w}^{k+k_p}$ and e_{1u}^k are employed for both the density gradients in ${}^k\mathbb{R}_{i_p}^{k_p}$ and the tracer gradients, the lateral density flux associated with each triad separately disappears.

$$\mathbb{F}_{u_{i_p}}^{k_p}(\rho) = -\alpha_i^k {}^k\mathbb{F}_{u_{i_p}}^{k_p}(T) + \beta_i^k {}^k\mathbb{F}_{u_{i_p}}^{k_p}(S) = 0 \quad (\text{D.10})$$

Thus the total flux $(F_u^{31})_{i,k+\frac{1}{2}}^i + (F_u^{11})_{i,k+\frac{1}{2}}^i$ 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 [equation D.4](#) 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

$$(F_w^{33})_{i,k+\frac{1}{2}}^{k+\frac{1}{2}} = - (A_i^{k+1} a_1' s_1'^2 + A_i^{k+1} a_2' s_2'^2 + A_i^k a_3' s_3'^2 + A_i^k a_4' s_4'^2) \delta_{k+\frac{1}{2}} [T^{i+1}], \quad (\text{D.11})$$

where the areas a' and slopes s' are the same as in [equation D.6](#). Then, separating the total vertical flux, the sum of [equation D.6](#) and [equation D.11](#), into triad components, a vertical flux

$${}^k\mathbb{F}_{w_{i_p}}^{k_p}(T) = A_i^k {}^k\mathbb{A}_{w_{i_p}}^{k_p} \left({}^k\mathbb{R}_{i_p}^{k_p} \frac{\delta_{i+i_p}[T^k]}{e_{1u}^k} - \left({}^k\mathbb{R}_{i_p}^{k_p} \right)^2 \frac{\delta_{k+k_p}[T^i]}{e_{3w}^i} \right) \quad (\text{D.12})$$

$$= - \left({}^k\mathbb{A}_{w_{i_p}}^{k_p} / {}^k\mathbb{A}_{u_{i_p}}^{k_p} \right) {}^k\mathbb{R}_{i_p}^{k_p} {}^k\mathbb{F}_{u_{i_p}}^{k_p}(T) \quad (\text{D.13})$$

may be associated with each triad. Each vertical density flux ${}^k\mathbb{F}_{w_{i_p}}^{k_p}(\rho)$ associated with a triad then separately disappears (because the lateral flux ${}^k\mathbb{F}_{u_{i_p}}^{k_p}(\rho)$ disappears). Consequently the total vertical density flux $(F_w^{31})_{i,k+\frac{1}{2}}^{k+\frac{1}{2}} + (F_w^{33})_{i,k+\frac{1}{2}}^{k+\frac{1}{2}}$ 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 ([figure D.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 [equation D.6](#), [equation D.5](#), [equation D.8](#) [equation D.11](#) and [figure D.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:

$$F_{\text{iso}}(T) \equiv \sum_{i_p, k_p} \left(\begin{matrix} k \\ i+1/2-i_p \end{matrix} \mathbb{F}_{u_{i_p}}^{k_p}(T) \right) - \sum_{i_p, k_p} \left(\begin{matrix} k+1/2-k_p \\ i \end{matrix} \mathbb{F}_{w_{i_p}}^{k_p}(T) \right). \quad (\text{D.14})$$

D.2.5. Ensuring the scheme does not increase tracer variance

We now require that this operator should not increase the globally-integrated tracer variance. Each triad slope ${}^k\mathbb{R}_{i_p}^{k_p}$ drives a lateral flux ${}^k\mathbb{F}_{u_{i_p}}^{k_p}(T)$ across the u -point $i+i_p, k$ and a vertical flux ${}^k\mathbb{F}_{w_{i_p}}^{k_p}(T)$ across the w -point $i, k+k_p$. The lateral flux drives a net rate of change of variance, summed over the two T -points $i+i_p-\frac{1}{2}, k$ and $i+i_p+\frac{1}{2}, k$, of

$$\begin{aligned} b_{T_{i+i_p-1/2}}^k \left(\frac{\partial T}{\partial t} T \right)_{i+i_p-1/2}^k &+ b_{T_{i+i_p+1/2}}^k \left(\frac{\partial T}{\partial t} T \right)_{i+i_p+1/2}^k \\ &= -T_{i+i_p-1/2}^k {}^k\mathbb{F}_{u_{i_p}}^{k_p}(T) + T_{i+i_p+1/2}^k {}^k\mathbb{F}_{u_{i_p}}^{k_p}(T) \\ &= {}^k\mathbb{F}_{u_{i_p}}^{k_p}(T) \delta_{i+i_p}[T^k], \end{aligned} \quad (\text{D.15})$$

while the vertical flux similarly drives a net rate of change of variance summed over the T -points $i, k+k_p-\frac{1}{2}$ (above) and $i, k+k_p+\frac{1}{2}$ (below) of

$${}^k\mathbb{F}_{w_{i_p}}^{k_p}(T) \delta_{k+k_p}[T^i]. \quad (\text{D.16})$$

The total variance tendency driven by the triad is the sum of these two. Expanding ${}^k\mathbb{F}_{u_{i_p}}^{k_p}(T)$ and ${}^k\mathbb{F}_{w_{i_p}}^{k_p}(T)$ with [equation D.9](#) and [equation D.12](#), it is

$$\begin{aligned} &-A_i^k \left\{ {}^k\mathbb{A}_{u_{i_p}}^{k_p} \left(\frac{\delta_{i+i_p}[T^k]}{e_{1u}^k} - {}^k\mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w}^i} \right) \delta_{i+i_p}[T^k] \right. \\ &\quad \left. - {}^k\mathbb{A}_{w_{i_p}}^{k_p} \left(\frac{\delta_{i+i_p}[T^k]}{e_{1u}^k} - {}^k\mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w}^i} \right) {}^k\mathbb{R}_{i_p}^{k_p} \delta_{k+k_p}[T^i] \right\}. \end{aligned}$$

The key point is then that if we require ${}^k_i\mathbb{A}_{u i_p}^{k_p}$ and ${}^k_i\mathbb{A}_{w i_p}^{k_p}$ to be related to a triad volume ${}^k_i\mathbb{V}_{i_p}^{k_p}$ by

$${}^k_i\mathbb{V}_{i_p}^{k_p} = {}^k_i\mathbb{A}_{u i_p}^{k_p} e_{1u i+i_p}^k = {}^k_i\mathbb{A}_{w i_p}^{k_p} e_{3w i}^{k+k_p}, \quad (\text{D.17})$$

the variance tendency reduces to the perfect square

$$-A_i^k {}^k_i\mathbb{V}_{i_p}^{k_p} \left(\frac{\delta_{i+i_p}[T^k]}{e_{1u i+i_p}^k} - {}^k_i\mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w i}^{k+k_p}} \right)^2 \leq 0. \quad (\text{D.18})$$

Thus, the constraint [equation D.17](#) ensures that the fluxes ([equation D.9](#), [equation D.12](#)) associated with a given slope triad ${}^k_i\mathbb{R}_{i_p}^{k_p}$ 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 [equation D.17](#) can be interpreted as a discretization of the global integral

$$\frac{\partial}{\partial t} \int \frac{1}{2} T^2 dV = \int \mathbf{F} \cdot \nabla T dV, \quad (\text{D.19})$$

where, within each triad volume ${}^k_i\mathbb{V}_{i_p}^{k_p}$, the lateral and vertical fluxes/unit area

$$\mathbf{F} = \left({}^k_i\mathbb{F}_{u i_p}^{k_p}(T) / {}^k_i\mathbb{A}_{u i_p}^{k_p}, {}^k_i\mathbb{F}_{w i_p}^{k_p}(T) / {}^k_i\mathbb{A}_{w i_p}^{k_p} \right)$$

and the gradient

$$\nabla T = \left(\delta_{i+i_p}[T^k] / e_{1u i+i_p}^k, \delta_{k+k_p}[T^i] / e_{3w i}^{k+k_p} \right)$$

D.2.6. Triad volumes in Griffes's scheme and in NEMO

To complete the discretization we now need only specify the triad volumes ${}^k_i\mathbb{V}_{i_p}^{k_p}$. [Griffes et al. \(1998\)](#) identifies these ${}^k_i\mathbb{V}_{i_p}^{k_p}$ 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 [equation D.19](#). 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

$${}^k_i\mathbb{V}_{i_p}^{k_p} = \frac{1}{4} b_{u i+i_p}^k, \quad (\text{D.20})$$

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

$$-\bar{A}_{i+1/2}^k \frac{b_{u i+1/2}^k}{e_{1u i+i_p}^k} \frac{\delta_{i+1/2}[T^k]}{e_{1u i+i_p}^k} = -\bar{A}_{i+1/2}^k \frac{e_{1w i+1/2}^k e_{1v i+1/2}^k}{e_{1u i+1/2}^k} \frac{\delta_{i+1/2}[T^k]}{e_{1u i+1/2}^k}. \quad (\text{D.21})$$

In fact if the diffusive coefficient is defined at u -points, so that we employ $A_{i+i_p}^k$ instead of A_i^k in the definitions of the triad fluxes [equation D.9](#) and [equation D.12](#), we can replace $\bar{A}_{i+1/2}^k$ by $A_{i+1/2}^k$ in the above.

D.2.7. Summary of the scheme

The iso-neutral fluxes at u - and w -points are the sums of the triad fluxes that cross the u - and w -faces [equation D.14](#):

$$F_{\text{iso}}(T) \equiv \sum_{i_p, k_p} \left(\begin{matrix} k \\ i+1/2-i_p \end{matrix} \mathbb{F}_{u i_p}^{k_p}(T) \right),$$

where [equation D.9](#):

$${}^k_i\mathbb{F}_{u i_p}^{k_p}(T) = -A_i^k \frac{{}^k_i\mathbb{V}_{i_p}^{k_p}}{e_{1u i+i_p}^k} \left(\frac{\delta_{i+i_p}[T^k]}{e_{1u i+i_p}^k} - {}^k_i\mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w i}^{k+k_p}} \right), \quad (\text{D.22a})$$

and

$${}^k_i\mathbb{F}_{w i_p}^{k_p}(T) = A_i^k \frac{{}^k_i\mathbb{V}_{i_p}^{k_p}}{e_{3w i}^{k+k_p}} \left({}^k_i\mathbb{R}_{i_p}^{k_p} \frac{\delta_{i+i_p}[T^k]}{e_{1u i+i_p}^k} - \left({}^k_i\mathbb{R}_{i_p}^{k_p} \right)^2 \frac{\delta_{k+k_p}[T^i]}{e_{3w i}^{k+k_p}} \right), \quad (\text{D.22b})$$

with equation D.20

$${}_i^k \nabla_{i_p}^{k_p} = \frac{1}{4} b_{u_{i+i_p}}^k.$$

The divergence of the expression equation D.14 for the fluxes gives the iso-neutral diffusion tendency at each tracer point:

$$D_l^T = \frac{1}{b_T} \sum_{i_p, k_p} \left\{ \delta_i \left[{}_i^k \nabla_{i_p}^{k_p} \mathbb{F}_{u_{i_p}}^{k_p} \right] + \delta_k \left[{}_i^{k+1/2-k_p} \mathbb{F}_{w_{i_p}}^{k_p} \right] \right\}$$

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 the traditional five-point Laplacian equation D.21 in the limit of flat iso-neutral direction:

$$D_l^T = \frac{1}{b_T} \delta_i \left[\frac{e_{2u} e_{3u}}{e_{1u}} \bar{A}^i \delta_{i+1/2} [T] \right] \quad \text{when} \quad {}_i^k \mathbb{R}_{i_p}^{k_p} = 0$$

Implicit treatment in the vertical Only tracer values associated with a single water column appear in the expression equation D.11 for the ${}_{33}$ fluxes, vertical fluxes driven by vertical gradients. This is of paramount importance since it means that a time-implicit algorithm can be used to solve the vertical diffusion equation. This is necessary since the vertical eddy diffusivity associated with this term,

$$\frac{1}{b_w} \sum_{i_p, k_p} \left\{ {}_i^k \nabla_{i_p}^{k_p} A_i^k \left({}_i^k \mathbb{R}_{i_p}^{k_p} \right)^2 \right\} = \frac{1}{4b_w} \sum_{i_p, k_p} \left\{ b_{u_{i+i_p}}^k A_i^k \left({}_i^k \mathbb{R}_{i_p}^{k_p} \right)^2 \right\},$$

(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 equation D.10 and equation D.13.

Conservation of tracer The iso-neutral diffusion conserves tracer content, *i.e.*

$$\sum_{i,j,k} \{ D_l^T b_T \} = 0$$

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.*

$$\sum_{i,j,k} \{ T D_l^T b_T \} \leq 0$$

The property is demonstrated in subsection D.2.5 above. It is a key property for a diffusion term. It means that it is also a dissipation term, *i.e.* it dissipates 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 becomes free of grid-point noise.

Self-adjoint operator The iso-neutral diffusion operator is self-adjoint, *i.e.*

$$\sum_{i,j,k} \{ S D_l^T b_T \} = \sum_{i,j,k} \{ D_l^S T b_T \} \quad (\text{D.23})$$

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 equation D.23, can be found by replacing $\delta[T]$ by $\delta[S]$ in equation D.15 and equation D.16. This results in a term similar to equation D.18,

$$-A_i^k {}_i^k \nabla_{i_p}^{k_p} \left(\frac{\delta_{i+i_p} [T^k]}{e_{1u}^k} - {}_i^k \mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p} [T^i]}{e_{3w}^i} \right) \left(\frac{\delta_{i+i_p} [S^k]}{e_{1u}^k} - {}_i^k \mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p} [S^i]}{e_{3w}^i} \right).$$

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 equation D.23.

D.2.8. Treatment of the triads at the boundaries

The triad slope can only be defined where both the grid boxes centred at the end of the arms exist. Triads that would poke up through the upper ocean surface into the atmosphere, or down into the ocean floor, must be masked out. See figure D.3. Surface layer triads ${}^1_i\mathbb{R}_{1/2}^{-1/2}$ (magenta) and ${}^1_{i+1}\mathbb{R}_{-1/2}^{-1/2}$ (blue) that require density to be specified above the ocean surface are masked (figure D.3a): this ensures that lateral tracer gradients produce no flux through the ocean surface. However, to prevent surface noise, it is customary to retain the ${}_{11}$ contributions towards the lateral triad fluxes ${}^1_i\mathbb{F}_{u_{1/2}}^{-1/2}$ and ${}^1_{i+1}\mathbb{F}_{u_{-1/2}}^{-1/2}$; this drives diapycnal tracer fluxes. Similar comments apply to triads that would intersect the ocean floor (figure D.3b). Note that both near bottom triad slopes ${}^k_i\mathbb{R}_{1/2}^{1/2}$ and ${}^k_{i+1}\mathbb{R}_{-1/2}^{1/2}$ are masked when either of the $i, k + 1$ or $i + 1, k + 1$ tracer points is masked, *i.e.* the $i, k + 1$ u -point is masked. The associated lateral fluxes (grey-black dashed line) are masked if `ln_botmix_triad=.false.`, but left unmasked, giving bottom mixing, if `ln_botmix_triad=.true.`

The default option `ln_botmix_triad=.false.` is suitable when the bbl mixing option is enabled (`ln_trabbl=.true.`, with `nn_bbl_ldf=1`), or for simple idealized problems. For setups with topography without bbl mixing, `ln_botmix_triad=.true.` may be necessary.

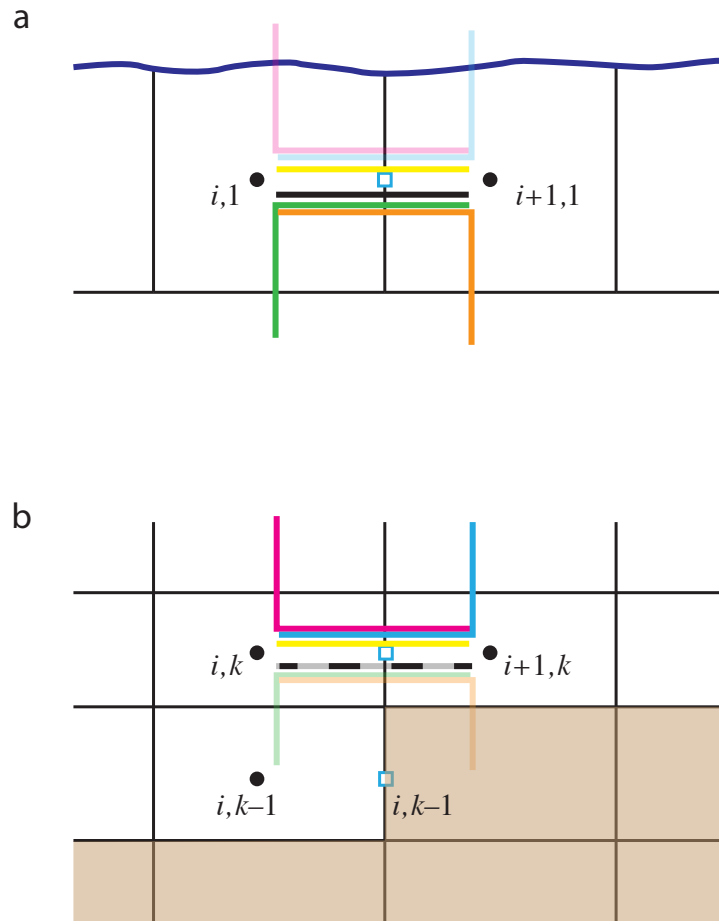


Figure D.3.: (a) Uppermost model layer $k = 1$ with $i, 1$ and $i + 1, 1$ tracer points (black dots), and $i + 1/2, 1$ u -point (blue square). Triad slopes ${}^1_i\mathbb{R}_{1/2}^{-1/2}$ (magenta) and ${}^1_{i+1}\mathbb{R}_{-1/2}^{-1/2}$ (blue) poking through the ocean surface are masked (faded in figure). However, the lateral ${}_{11}$ contributions towards ${}^1_i\mathbb{F}_{u_{1/2}}^{-1/2}$ and ${}^1_{i+1}\mathbb{F}_{u_{-1/2}}^{-1/2}$ (yellow line) are still applied, giving diapycnal diffusive fluxes. (b) Both near bottom triad slopes ${}^k_i\mathbb{R}_{1/2}^{1/2}$ and ${}^k_{i+1}\mathbb{R}_{-1/2}^{1/2}$ are masked when either of the $i, k + 1$ or $i + 1, k + 1$ tracer points is masked, *i.e.* the $i, k + 1$ u -point is masked. The associated lateral fluxes (grey-black dashed line) are masked if `ln_botmix_triad=.false.`, but left unmasked, giving bottom mixing, if `ln_botmix_triad=.true.`

D.2.9. Limiting of the slopes within the interior

As discussed in subsection 8.2.2, iso-neutral slopes relative to geopotentials must be bounded everywhere, both for consistency with the small-slope approximation and for numerical stability (Cox, 1987; Griffies, 2004). The bound chosen in NEMO is applied to each component of the slope separately and has a value of 1/100 in the ocean interior. It is of

course relevant to the iso-neutral slopes $\tilde{r}_i = r_i + \sigma_i$ relative to geopotentials (here the σ_i are the slopes of the coordinate surfaces relative to geopotentials) [equation 1.20](#) rather than the slope r_i relative to coordinate surfaces, so we require

$$|\tilde{r}_i| \leq \tilde{r}_{\max} = 0.01.$$

and then recalculate the slopes r_i relative to coordinates. Each individual triad slope

$${}^k\tilde{\mathbb{R}}_{i_p}^{k_p} = {}^k\mathbb{R}_{i_p}^{k_p} + \frac{\delta_{i+i_p}[z_T^k]}{e_{1u}^k}_{i+i_p} \quad (\text{D.24})$$

is limited like this and then the corresponding ${}^k\tilde{\mathbb{R}}_{i_p}^{k_p}$ are recalculated and combined to form the fluxes. Note that where the slopes have been limited, there is now a non-zero iso-neutral density flux that drives dianeutral mixing. In particular this iso-neutral density flux is always downwards, and so acts to reduce gravitational potential energy.

D.2.10. Tapering within the surface mixed layer

Additional tapering of the iso-neutral fluxes is necessary within the surface mixed layer. When the Griffies triads are used, we offer two options for this.

Linear slope tapering within the surface mixed layer

This is the option activated by the default choice `ln_triad_iso=.false.`. Slopes \tilde{r}_i relative to geopotentials are tapered linearly from their value immediately below the mixed layer to zero at the surface, as described in option (c) of [figure 8.2](#), to values

$$\tilde{r}_{\text{ML}i} = -\frac{z}{h} \tilde{r}_i|_{z=-h} \quad \text{for } z > -h, \quad (\text{D.25})$$

and then the r_i relative to vertical coordinate surfaces are appropriately adjusted to

$$\text{ML}i = \tilde{r}_{\text{ML}i} - \sigma_i \quad \text{for } z > -h.$$

Thus the diffusion operator within the mixed layer is given by:

$$D^{lT} = \nabla \cdot (A^{lT} \mathfrak{R} \nabla T) \quad \text{with} \quad \mathfrak{R} = \begin{pmatrix} 1 & 0 & -\text{ML}1 \\ 0 & 1 & -\text{ML}2 \\ -\text{ML}1 & -\text{ML}2 & \text{ML}1^2 + \text{ML}2^2 \end{pmatrix}$$

This slope tapering gives a natural connection between tracer in the mixed-layer and in isopycnal layers immediately below, in the thermocline. It is consistent with the way the \tilde{r}_i are tapered within the mixed layer (see [subsection D.3.5](#) below) so as to ensure a uniform GM eddy-induced velocity throughout the mixed layer. However, it gives a downwards density flux and so acts so as to reduce potential energy in the same way as does the slope limiting discussed above in [subsection D.2.9](#).

As in [subsection D.2.9](#) above, the tapering [equation D.25](#) is applied separately to each triad ${}^k\tilde{\mathbb{R}}_{i_p}^{k_p}$, and the ${}^k\mathbb{R}_{i_p}^{k_p}$ adjusted. For clarity, we assume z -coordinates in the following; the conversion from \mathbb{R} to $\tilde{\mathbb{R}}$ and back to \mathbb{R} follows exactly as described above by [equation D.24](#).

1. Mixed-layer depth is defined so as to avoid including regions of weak vertical stratification in the slope definition. At each i, j (simplified to i in [figure D.4](#)), we define the mixed-layer by setting the vertical index of the tracer point immediately below the mixed layer, k_{ML} , as the maximum k (shallowest tracer point) such that the potential density $\rho_{0i,k} > \rho_{0i,k_{10}} + \Delta\rho_c$, where i, k_{10} is the tracer gridbox within which the depth reaches 10 m. See the left side of [figure D.4](#). We use the k_{10} -gridbox instead of the surface gridbox to avoid problems *e.g.* with thin daytime mixed-layers. Currently we use the same $\Delta\rho_c = 0.01 \text{ kg m}^{-3}$ for ML triad tapering as is used to output the diagnosed mixed-layer depth $h_{\text{ML}} = |z_W|_{k_{\text{ML}}+1/2}$, the depth of the w -point above the i, k_{ML} tracer point.
2. We define ‘basal’ triad slopes ${}^i\mathbb{R}_{\text{base}i_p}^{k_p}$ as the slopes of those triads whose vertical ‘arms’ go down from the i, k_{ML} tracer point to the $i, k_{\text{ML}} - 1$ tracer point below. This is to ensure that the vertical density gradients associated with these basal triad slopes ${}^i\mathbb{R}_{\text{base}i_p}^{k_p}$ are representative of the thermocline. The four basal triads defined in the bottom part of [figure D.4](#) are then

$${}^i\mathbb{R}_{\text{base}i_p}^{k_p} = {}^i{}^{k_{\text{ML}}-k_p-1/2}\mathbb{R}_{\text{base}i_p}^{k_p},$$

with *e.g.* the green triad

$${}^i\mathbb{R}_{\text{base}1/2}^{-1/2} = {}^i{}^{k_{\text{ML}}}\mathbb{R}_{\text{base}1/2}^{-1/2}.$$

The vertical flux associated with each of these triads passes through the w -point $i, k_{\text{ML}} - 1/2$ lying below the i, k_{ML} tracer point, so it is this depth

$$z_{\text{base } i} = z_{w k_{\text{ML}} - 1/2}$$

one gridbox deeper than the diagnosed ML depth z_{ML}) that sets the h used to taper the slopes in [equation D.25](#).

3. Finally, we calculate the adjusted triads ${}^k_i \mathbb{R}_{\text{ML}}^{k_p}$ within the mixed layer, by multiplying the appropriate ${}^i \mathbb{R}_{\text{base } i_p}^{k_p}$ by the ratio of the depth of the w -point $z_{w k+k_p}$ to $z_{\text{base } i}$. For instance the green triad centred on i, k

$${}^k_i \mathbb{R}_{\text{ML}}^{-1/2} = \frac{z_{w k-1/2}}{z_{\text{base } i}} {}^i \mathbb{R}_{\text{base } 1/2}^{-1/2}$$

and more generally

$${}^k_i \mathbb{R}_{\text{ML}}^{k_p} = \frac{z_{w k+k_p}}{z_{\text{base } i}} {}^i \mathbb{R}_{\text{base } i_p}^{k_p}.$$

Additional truncation of skew iso-neutral flux components

The alternative option is activated by setting `ln_triad_iso = true`. This retains the same tapered slope ${}_{\text{ML } i}$ described above for the calculation of the ${}_{33}$ term of the iso-neutral diffusion tensor (the vertical tracer flux driven by vertical tracer gradients), but replaces the ${}_{\text{ML } i}$ in the skew term by

$${}^*_{\text{ML } i} = \tilde{r}_{\text{ML } i}^2 / \tilde{r}_i - \sigma_i, \quad (\text{D.26})$$

giving a ML diffusive operator

$$D^{lT} = \nabla \cdot (A^{lT} \mathfrak{R} \nabla T) \quad \text{with} \quad \mathfrak{R} = \begin{pmatrix} 1 & 0 & -{}^*_{\text{ML } 1} \\ 0 & 1 & -{}^*_{\text{ML } 2} \\ -{}^*_{\text{ML } 1} & -{}^*_{\text{ML } 2} & {}^2_{\text{ML } 1} + {}^2_{\text{ML } 2} \end{pmatrix}.$$

This operator * then has the property it gives no vertical density flux, and so does not change the potential energy. This approach is similar to multiplying the iso-neutral diffusion coefficient by $\tilde{r}_{\text{max}}^{-2} \tilde{r}_i^{-2}$ for steep slopes, as suggested by [Gerdes et al. \(1991\)](#) (see also [Griffies \(2004\)](#)). Again it is applied separately to each triad ${}^k_i \mathbb{R}_{i_p}^{k_p}$

In practice, this approach gives weak vertical tracer fluxes through the mixed-layer, as well as vanishing density fluxes. While it is theoretically advantageous that it does not change the potential energy, it may give a discontinuity between the fluxes within the mixed-layer (purely horizontal) and just below (along iso-neutral surfaces).

D.3. Eddy induced advection formulated as a skew flux

D.3.1. Continuous skew flux formulation

When Gent and McWilliams's [1990] diffusion is used, 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.* [equation 8.1](#) is used in z -coordinate, and the sum [equation 8.1](#) + [equation 8.2](#) in z^* or s -coordinates.

The eddy induced velocity is given by:

$$\begin{aligned} u^* &= -\frac{1}{e_3} \partial_i \psi_1, \\ v^* &= -\frac{1}{e_3} \partial_j \psi_2, \\ w^* &= \frac{1}{e_1 e_2} \{ \partial_i (e_2 \psi_1) + \partial_j (e_1 \psi_2) \}, \end{aligned} \quad (\text{D.27a})$$

where the streamfunctions ψ_i are given by

$$\begin{aligned} \psi_1 &= A_e \tilde{r}_1, \\ \psi_2 &= A_e \tilde{r}_2, \end{aligned} \quad (\text{D.27b})$$

*To ensure good behaviour where horizontal density gradients are weak, we in fact follow [Gerdes et al. \(1991\)](#) and set ${}^*_{\text{ML } i} = \text{sgn}(\tilde{r}_i) \min(|\tilde{r}_{\text{ML } i}^2 / \tilde{r}_i|, |\tilde{r}_i|) - \sigma_i$.

with A_e the eddy induced velocity coefficient, and \tilde{r}_1 and \tilde{r}_2 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 is implemented if `traldf_eiv?` is set in the default implementation, where `ln_traldf_triad` is set false. This allows us to take advantage of all the advection schemes offered for the tracers (see section 4.1) 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.

However, when `ln_traldf_triad` is set true, *NEMO* instead implements eddy induced advection according to the so-called skew form (Griffies, 1998). It is based on a transformation of the advective fluxes using the non-divergent nature of the eddy induced velocity. For example in the (\mathbf{i}, \mathbf{k}) plane, the tracer advective fluxes per unit area in ijk space can be transformed as follows:

$$\begin{aligned} \mathbf{F}_{\text{eiv}}^T &= \begin{pmatrix} e_2 e_3 u^* \\ e_1 e_2 w^* \end{pmatrix} T = \begin{pmatrix} -\partial_k (e_2 \psi_1) T \\ +\partial_i (e_2 \psi_1) T \end{pmatrix} \\ &= \begin{pmatrix} -\partial_k (e_2 \psi_1 T) \\ +\partial_i (e_2 \psi_1 T) \end{pmatrix} + \begin{pmatrix} +e_2 \psi_1 \partial_k T \\ -e_2 \psi_1 \partial_i T \end{pmatrix} \end{aligned}$$

and since the eddy induced velocity field is non-divergent, we end up with the skew form of the eddy induced advective fluxes per unit area in ijk space:

$$\mathbf{F}_{\text{eiv}}^T = \begin{pmatrix} +e_2 \psi_1 \partial_k T \\ -e_2 \psi_1 \partial_i T \end{pmatrix} \quad (\text{D.28})$$

The total fluxes per unit physical area are then

$$\begin{aligned} f_1^* &= \frac{1}{e_3} \psi_1 \partial_k T \\ f_2^* &= \frac{1}{e_3} \psi_2 \partial_k T \\ f_3^* &= -\frac{1}{e_1 e_2} \{e_2 \psi_1 \partial_i T + e_1 \psi_2 \partial_j T\}. \end{aligned} \quad (\text{D.29})$$

Note that equation D.29 takes the same form whatever the vertical coordinate, though of course the slopes \tilde{r}_i which define the ψ_i in equation D.27b are relative to geopotentials. The tendency associated with eddy induced velocity is then simply the convergence of the fluxes (equation D.28, equation D.29), so

$$\frac{\partial T}{\partial t} = -\frac{1}{e_1 e_2 e_3} \left[\frac{\partial}{\partial i} (e_2 \psi_1 \partial_k T) + \frac{\partial}{\partial j} (e_1 \psi_2 \partial_k T) - \frac{\partial}{\partial k} (e_2 \psi_1 \partial_i T + e_1 \psi_2 \partial_j T) \right]$$

It naturally conserves the tracer content, as it is expressed in flux form. Since it has the same divergence as the advective form it also preserves the tracer variance.

D.3.2. Discrete skew flux formulation

The skew fluxes in (equation D.29, equation D.28), like the off-diagonal terms (equation D.2, equation D.3) of the small angle diffusion tensor, are best expressed in terms of the triad slopes, as in figure D.1 and (equation D.5, equation D.6); but now in terms of the triad slopes $\tilde{\mathbb{R}}$ relative to geopotentials instead of the \mathbb{R} relative to coordinate surfaces. The discrete form of equation D.28 using the slopes equation D.7 and defining A_e at T -points is then given by:

$$\mathbf{F}_{\text{eiv}}(T) \equiv \sum_{i_p, k_p} \begin{pmatrix} {}^k \mathbb{S}_{u_{i_p}}^{k_p}(T) \\ {}^{k+1/2-k_p} \mathbb{S}_{w_{i_p}}^{k_p}(T) \end{pmatrix},$$

where the skew flux in the i -direction associated with a given triad is (equation D.9, equation D.22a):

$${}^k \mathbb{S}_{u_{i_p}}^{k_p}(T) = +\frac{1}{4} A_e^k \frac{b_{u_{i+i_p}}^k}{e_{1u}^k} \frac{k \tilde{\mathbb{R}}_{i_p}^{k_p}}{e_{3w}^k} \frac{\delta_{k+k_p}[T^i]}{e_{1u}^k}, \quad (\text{D.30a})$$

and equation D.22b in the k -direction, changing the sign to be consistent with equation D.28:

$${}^k \mathbb{S}_{w_{i_p}}^{k_p}(T) = -\frac{1}{4} A_e^k \frac{b_{u_{i+i_p}}^k}{e_{3w}^k} \frac{k \tilde{\mathbb{R}}_{i_p}^{k_p}}{e_{1u}^k} \frac{\delta_{i+i_p}[T^k]}{e_{1u}^k}. \quad (\text{D.30b})$$

Such a discretisation is consistent with the iso-neutral operator as it uses the same definition for the slopes. It also ensures the following two key properties.

No change in tracer variance

The discretization conserves tracer variance, *i.e.* it does not include a diffusive component but is a ‘pure’ advection term. This can be seen by considering the fluxes associated with a given triad slope ${}^k\mathbb{R}_{i_p}^{k_p}(T)$. For, following [subsection D.2.5](#) and [equation D.15](#), the associated horizontal skew-flux ${}^k\mathbb{S}_{u_{i_p}}^{k_p}(T)$ drives a net rate of change of variance, summed over the two T -points $i + i_p - \frac{1}{2}, k$ and $i + i_p + \frac{1}{2}, k$, of

$${}^k\mathbb{S}_{u_{i_p}}^{k_p}(T) \delta_{i+i_p}[T^k], \quad (\text{D.31})$$

while the associated vertical skew-flux gives a variance change summed over the T -points $i, k + k_p - \frac{1}{2}$ (above) and $i, k + k_p + \frac{1}{2}$ (below) of

$${}^k\mathbb{S}_{w_{i_p}}^{k_p}(T) \delta_{k+k_p}[T^i]. \quad (\text{D.32})$$

Inspection of the definitions ([equation D.30a](#), [equation D.30b](#)) shows that these two variance changes ([equation D.31](#), [equation D.32](#)) sum to zero. Hence the two fluxes associated with each triad make no net contribution to the variance budget.

Reduction in gravitational PE

The vertical density flux associated with the vertical skew-flux always has the same sign as the vertical density gradient; thus, so long as the fluid is stable (the vertical density gradient is negative) the vertical density flux is negative (downward) and hence reduces the gravitational PE.

For the change in gravitational PE driven by the k -flux is

$$ge_{3w_i}^{k+k_p} \mathbb{S}_{w_{i_p}}^{k_p}(\rho) = ge_{3w_i}^{k+k_p} \left[-\alpha_i^k {}^k\mathbb{S}_{w_{i_p}}^{k_p}(T) + \beta_i^k {}^k\mathbb{S}_{w_{i_p}}^{k_p}(S) \right].$$

Substituting ${}^k\mathbb{S}_{w_{i_p}}^{k_p}$ from [equation D.30b](#), gives

$$\begin{aligned} &= -\frac{1}{4} g A_{e_i}^k b_{u_{i+i_p}i}^k \tilde{\mathbb{R}}_{i_p}^{k_p} \frac{-\alpha_i^k \delta_{i+i_p}[T^k] + \beta_i^k \delta_{i+i_p}[S^k]}{e_{1u_{i+i_p}}^k} \\ &= +\frac{1}{4} g A_{e_i}^k b_{u_{i+i_p}i}^k \left({}^k\mathbb{R}_{i_p}^{k_p} + \frac{\delta_{i+i_p}[z_T^k]}{e_{1u_{i+i_p}}^k} \right) {}^k\mathbb{R}_{i_p}^{k_p} \frac{-\alpha_i^k \delta_{k+k_p}[T^i] + \beta_i^k \delta_{k+k_p}[S^i]}{e_{3w_i}^{k+k_p}}, \end{aligned} \quad (\text{D.33})$$

using the definition of the triad slope ${}^k\mathbb{R}_{i_p}^{k_p}$, [equation D.7](#) to express $-\alpha_i^k \delta_{i+i_p}[T^k] + \beta_i^k \delta_{i+i_p}[S^k]$ in terms of $-\alpha_i^k \delta_{k+k_p}[T^i] + \beta_i^k \delta_{k+k_p}[S^i]$.

Where the coordinates slope, the i -flux gives a PE change

$$\begin{aligned} &g \delta_{i+i_p}[z_T^k] \left[-\alpha_i^k {}^k\mathbb{S}_{w_{i_p}}^{k_p}(T) + \beta_i^k {}^k\mathbb{S}_{w_{i_p}}^{k_p}(S) \right] \\ &= +\frac{1}{4} g A_{e_i}^k b_{u_{i+i_p}i}^k \frac{\delta_{i+i_p}[z_T^k]}{e_{1u_{i+i_p}}^k} \left({}^k\mathbb{R}_{i_p}^{k_p} + \frac{\delta_{i+i_p}[z_T^k]}{e_{1u_{i+i_p}}^k} \right) \frac{-\alpha_i^k \delta_{k+k_p}[T^i] + \beta_i^k \delta_{k+k_p}[S^i]}{e_{3w_i}^{k+k_p}}, \end{aligned} \quad (\text{D.34})$$

(using [equation D.30a](#)) and so the total PE change [equation D.33](#) + [equation D.34](#) associated with the triad fluxes is

$$\begin{aligned} &ge_{3w_i}^{k+k_p} \mathbb{S}_{w_{i_p}}^{k_p}(\rho) + g \delta_{i+i_p}[z_T^k] {}^k\mathbb{S}_{w_{i_p}}^{k_p}(\rho) \\ &= +\frac{1}{4} g A_{e_i}^k b_{u_{i+i_p}i}^k \left({}^k\mathbb{R}_{i_p}^{k_p} + \frac{\delta_{i+i_p}[z_T^k]}{e_{1u_{i+i_p}}^k} \right)^2 \frac{-\alpha_i^k \delta_{k+k_p}[T^i] + \beta_i^k \delta_{k+k_p}[S^i]}{e_{3w_i}^{k+k_p}}. \end{aligned}$$

Where the fluid is stable, with $-\alpha_i^k \delta_{k+k_p}[T^i] + \beta_i^k \delta_{k+k_p}[S^i] < 0$, this PE change is negative.

D.3.3. Treatment of the triads at the boundaries

Triad slopes ${}^k\tilde{\mathbb{R}}_{i_p}^{k_p}$ used for the calculation of the eddy-induced skew-fluxes are masked at the boundaries in exactly the same way as are the triad slopes ${}^k\mathbb{R}_{i_p}^{k_p}$ used for the iso-neutral diffusive fluxes, as described in [subsection D.2.8](#) and [figure D.3](#). Thus surface layer triads ${}^1\tilde{\mathbb{R}}_{1/2}^{-1/2}$ and ${}^1_{i+1}\tilde{\mathbb{R}}_{-1/2}^{-1/2}$ are masked, and both near bottom triad slopes ${}^k\tilde{\mathbb{R}}_{1/2}^{1/2}$ and ${}^k_{i+1}\tilde{\mathbb{R}}_{-1/2}^{1/2}$ are masked when either of the $i, k + 1$ or $i + 1, k + 1$ tracer points is masked, *i.e.* the $i, k + 1$ u -point is masked. The namelist parameter `ln_botmix_triad` has no effect on the eddy-induced skew-fluxes.

D.3.4. Limiting of the slopes within the interior

Presently, the iso-neutral slopes \tilde{r}_i relative to geopotentials are limited to be less than 1/100, exactly as in calculating the iso-neutral diffusion, §subsection D.2.9. Each individual triad ${}^k_i\tilde{\mathbb{R}}_i^{k_p}$ is so limited.

D.3.5. Tapering within the surface mixed layer

The slopes \tilde{r}_i relative to geopotentials (and thus the individual triads ${}^k_i\tilde{\mathbb{R}}_i^{k_p}$) are always tapered linearly from their value immediately below the mixed layer to zero at the surface [equation D.25](#), as described in [section D.2.10](#). This is option (c) of [figure 8.2](#). This linear tapering for the slopes used to calculate the eddy-induced fluxes is unaffected by the value of `ln_triad_iso`.

The justification for this linear slope tapering is that, for A_e that is constant or varies only in the horizontal (the most commonly used options in *NEMO*: see [section 8.3](#)), it is equivalent to a horizontal eiv (eddy-induced velocity) that is uniform within the mixed layer [equation D.27a](#). This ensures that the eiv velocities do not restratify the mixed layer (Tréguier et al., 1997; Danabasoglu et al., 2008). Equivalently, in terms of the skew-flux formulation we use here, the linear slope tapering within the mixed-layer gives a linearly varying vertical flux, and so a tracer convergence uniform in depth (the horizontal flux convergence is relatively insignificant within the mixed-layer).

D.3.6. Streamfunction diagnostics

Where the namelist parameter `ln_traldf_gdia=.true.`, diagnosed mean eddy-induced velocities are output. Each time step, streamfunctions are calculated in the i - k and j - k planes at uw (integer $+1/2$ i , integer j , integer $+1/2$ k) and vw (integer i , integer $+1/2$ j , integer $+1/2$ k) points (see [Table 3.1](#)) respectively. We follow (Griffies, 2004) and calculate the streamfunction at a given uw -point from the surrounding four triads according to:

$$\psi_{i+1/2}^{k+1/2} = \frac{1}{4} \sum_{i_p, k_p} A_{e_{i+1/2-i_p}^{k+1/2-k_p}} {}^{k+1/2-k_p}_{i+1/2-i_p}\mathbb{R}_{i_p}^{k_p}.$$

The streamfunction ψ_1 is calculated similarly at vw points. The eddy-induced velocities are then calculated from the straightforward discretisation of [equation D.27a](#):

$$\begin{aligned} u_{i+1/2}^{*k} &= -\frac{1}{e_{3u_i^k}} \left(\psi_{i+1/2}^{k+1/2} - \psi_{i+1/2}^{k+1/2} \right), \\ v_{j+1/2}^{*k} &= -\frac{1}{e_{3v_j^k}} \left(\psi_{j+1/2}^{k+1/2} - \psi_{j+1/2}^{k+1/2} \right), \\ w_{i,j}^{*k+1/2} &= \frac{1}{e_{1t}e_{2t}} \left\{ e_{2u_{i+1/2}^{k+1/2}} \psi_{i+1/2}^{k+1/2} - e_{2u_{i-1/2}^{k+1/2}} \psi_{i-1/2}^{k+1/2} + \right. \\ &\quad \left. e_{2v_{j+1/2}^{k+1/2}} \psi_{j+1/2}^{k+1/2} - e_{2v_{j-1/2}^{k+1/2}} \psi_{j-1/2}^{k+1/2} \right\}, \end{aligned}$$

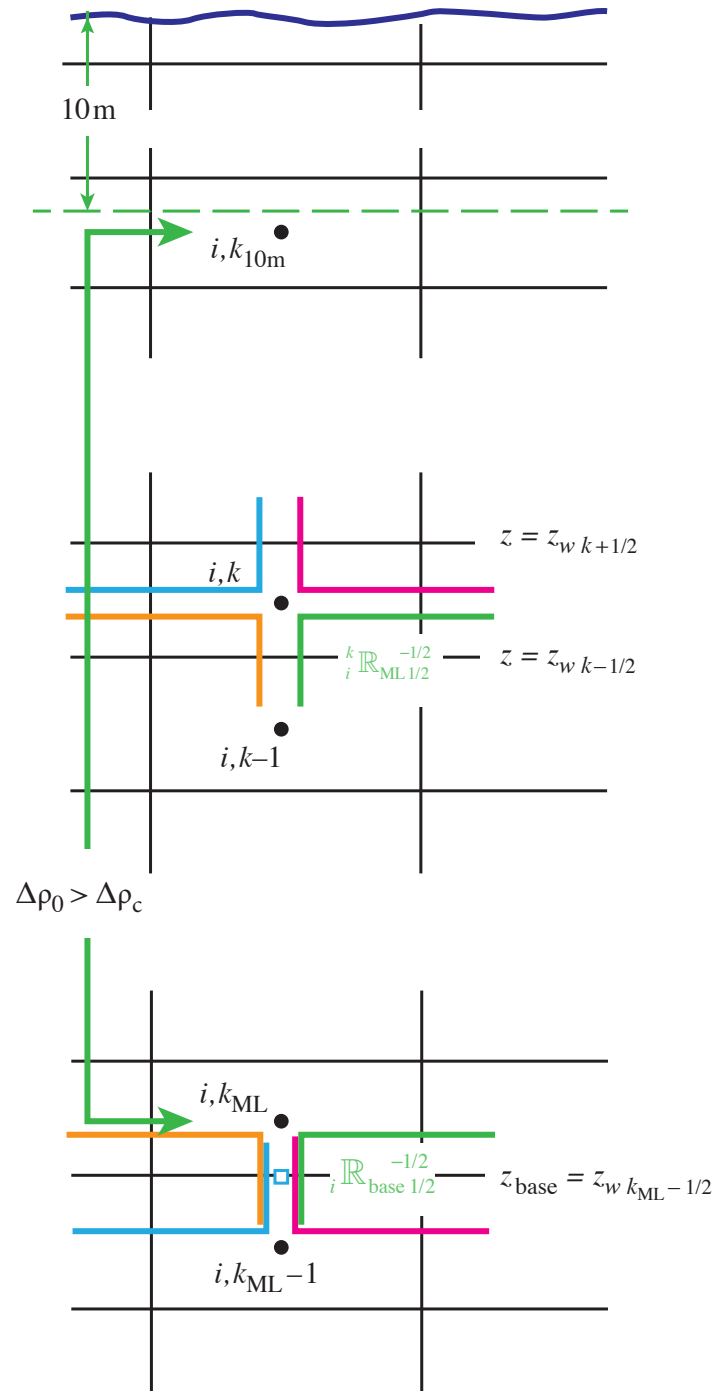


Figure D.4.: Definition of mixed-layer depth and calculation of linearly tapered triads. The figure shows a water column at a given i, j (simplified to i), with the ocean surface at the top. Tracer points are denoted by bullets, and black lines the edges of the tracer cells; k increases upwards. We define the mixed-layer by setting the vertical index of the tracer point immediately below the mixed layer, k_{ML} , as the maximum k (shallowest tracer point) such that $\rho_{0,i,k} > \rho_{0,i,k_{10}} + \Delta\rho_c$, where i, k_{10} is the tracer gridbox within which the depth reaches 10 m. We calculate the triad slopes within the mixed layer by linearly tapering them from zero (at the surface) to the ‘basal’ slopes, the slopes of the four triads passing through the w -point $i, k_{ML} - 1/2$ (blue square), $i \mathbb{R}_{base}^{k_p}$. Triads with different i_p, k_p , denoted by different colours, (e.g. the green triad $i_p = 1/2, k_p = -1/2$) are tapered to the appropriate basal triad.



A brief guide to the DOMAINcfg tool

Table of contents

E.1. Choice of horizontal grid	241
E.2. Vertical grid	242
E.2.1. Vertical reference coordinate	242
E.2.2. Model bathymetry	245
E.2.3. Choice of vertical grid	245

Changes record

Release	Author(s)	Modifications
4.0
3.6
3.4
<=3.4


```

!-----
&namdom      !   space and time domain (bathymetry, mesh, timestep)
!-----
nn_bathy    = 1      !   compute (=0) or read (=1) the bathymetry file
rn_bathy    = 0.     !   value of the bathymetry. if (=0) bottom flat at jpkml
nn_closea   = 0      !   remove (=0) or keep (=1) closed seas and lakes (ORCA)
nn_msh      = 1      !   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_isfhmin  = 1.00   !   threshold (m) to discriminate grounding ice to floating ice
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)
rn_atfp     = 0.1    !   asselin time filter parameter
ln_crs      = .false. !   Logical switch for coarsening module
jphgr_msh   = 0      !   type of horizontal mesh
! = 0 curvilinear coordinate on the sphere read in coordinate.nc
! = 1 geographical mesh on the sphere with regular grid-spacing
! = 2 f-plane with regular grid-spacing
! = 3 beta-plane with regular grid-spacing
! = 4 Mercator grid with T/U point at the equator
! longitude of first raw and column T-point (jphgr_msh = 1)
ppglam0    = 0.0     !
ppgphi0    = -35.0   ! latitude of first raw and column T-point (jphgr_msh = 1)
ppe1_deg   = 1.0     ! zonal grid-spacing (degrees)
ppe2_deg   = 0.5     ! meridional grid-spacing (degrees)
ppe1_m     = 5000.0  ! zonal grid-spacing (degrees)
ppe2_m     = 5000.0  ! meridional grid-spacing (degrees)
ppsur      = -4762.96143546300 ! ORCA r4, r2 and r05 coefficients
ppa0       = 255.58049070440 ! (default coefficients)
ppa1       = 245.58132232490 !
ppkth      = 21.43336197938 !
ppacr      = 3.0     !
ppdzmin    = 10.     ! Minimum vertical spacing
pphmax     = 5000.   ! Maximum depth
ldbletanh  = .TRUE.  ! Use/do not use double tanh function for vertical coordinates
ppa2       = 100.7609285000000 ! Double tanh function parameters
ppkth2     = 48.029893720000 !
ppacr2     = 13.000000000000 !
/
    
```

namelist E.1.: &namdom_domcfg

This appendix briefly describes some of the options available in the `DOMAINcfg` tool mentioned in [chapter 3](#).

This tool will evolve into an independent utility with its own documentation but its current manifestation is mostly a wrapper for `NEMO DOM` modules more aligned to those in the previous versions of `NEMO`. These versions allowed the user to define some horizontal and vertical grids through additional namelist parameters. Explanations of these parameters are retained here for reference pending better documentation for `DOMAINcfg`. Please note that the namelist blocks named in this appendix refer to those read by `DOMAINcfg` via its own `namelist_ref` and `namelist_cfg` files. Although, due to their origins, these namelists share names with those used by `NEMO`, they are not interchangeable and should be considered independent of those described elsewhere in this manual.

E.1. Choice of horizontal grid

The user has three options available in defining a horizontal grid, which involve the namelist variable `jphgr_mesh` of the `&namdom` (namelist 3.1) (`DOMAINcfg` variant only) namelist.

`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. This is now the only option available within `NEMO` itself from v4.0 onwards.

`jphgr_mesh = 1 to 5` A few simple analytical grids are provided (see below). For other analytical grids, the `domhgr.F90` module (`DOMAINcfg` variant) must be modified by the user. In most cases, modifying the `usrdef_hgr.F90` module of `NEMO` is a better alternative since this is designed to allow simple analytical domains to be configured and used without the need for external data files.

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`

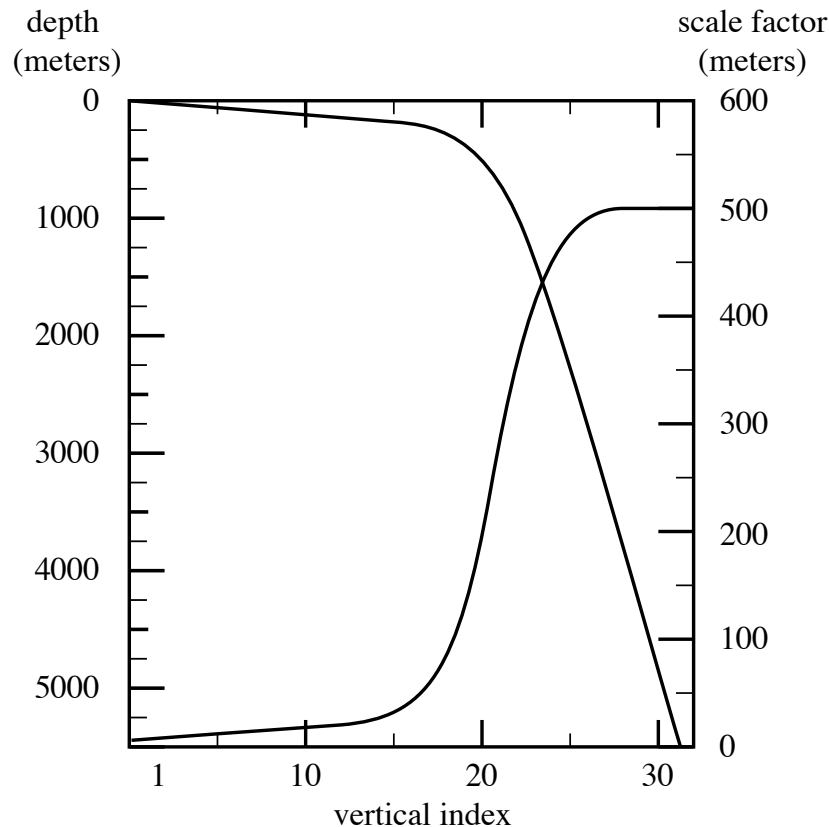


Figure E.1.: Default vertical mesh for ORCA2: 30 ocean levels (L30). Vertical level functions for (a) T-point depth and (b) the associated scale factor for the z -coordinate case.

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 southwesternmost 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 variable `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).

E.2. Vertical grid

E.2.1. Vertical reference coordinate

The reference coordinate transformation $z_0(k)$ defines the arrays $gdept_0$ and $gdepw_0$ for t - and w -points, respectively. See [section E.2.3](#) for the S-coordinate options. As indicated on [figure 3.4](#) `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 `&namdom` ([namelist 3.1](#)) namelist (DOMAINcfg variant).

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):

$$z_0(k) = h_{sur} - h_0 k - h_1 \log \left[\cosh((k - h_{th})/h_{cr}) \right] \quad (E.1)$$

$$e_3^0(k) = \left| -h_0 - h_1 \tanh \left[(k - h_{th})/h_{cr} \right] \right| \quad (E.2)$$

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 (figure E.1).

A double hyperbolic tangent version (`ldbletanh=.true.`) is also available which permits finer control and is used, typically, to obtain a well resolved upper ocean without compromising on resolution at depth using a moderate number of levels.

$$z_0(k) = h_{sur} - h_0 k - h_1 \log \left[\cosh((k - h_{th})/h_{cr}) \right] - h_{21} \log \left[\cosh((k - h_{2th})/h_{2cr}) \right] \quad (E.3)$$

$$e_3^0(k) = \left| -h_0 - h_1 \tanh \left[(k - h_{th})/h_{cr} \right] - h_{21} \tanh \left[(k - h_{2th})/h_{2cr} \right] \right| \quad (E.4)$$

If the ice shelf cavities are opened (`ln_isfcav=.true.`), the definition of z_0 is the same. However, definition of e_3^0 at t - and w -points is respectively changed to:

$$\begin{aligned} e_3^T(k) &= z_W(k+1) - z_W(k) \\ e_3^W(k) &= z_T(k) - z_T(k-1) \end{aligned} \quad (E.5)$$

This formulation decreases the self-generated circulation into the ice shelf cavity (which can, in extreme case, leads to numerical instability). This is now the recommended formulation for all configurations using v4.0 onwards. The analytical derivation of thicknesses is maintained for backwards compatibility.

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

$$\begin{aligned} e_3(1+1/2) &= 10. & z(1) &= 0. \\ e_3(jpk-1/2) &= 500. & z(jpk) &= -5000. \end{aligned} \quad (E.6)$$

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 equation E.5 have been determined such that equation E.6 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 figure E.1 and given in table E.1. Those values correspond to the parameters `ppsur`, `ppa0`, `ppa1`, `ppkth` in `&namcfg (namelist 15.1)` namelist.

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 = 999999.`, in `&namcfg (namelist 15.1)` namelist, and specifies instead the four following parameters:

- `ppacr = hcr`: stretching factor (nondimensional). The larger `ppacr`, the smaller the stretching. Values from 3 to 10 are usual.
- `ppkth = hth`: 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 = 6 m`, `pphmax = 5750 m`.

LEVEL	gdept_1d	gdepw_1d	e3t_1d	e3w_1d
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

Table E.1.: Default vertical mesh in z -coordinate for 30 layers ORCA2 configuration as computed from [equation E.5](#) using the coefficients given in [equation E.6](#)

E.2.2. Model bathymetry

Three options are possible for defining the bathymetry, according to the namelist variable `nn_bathy` (found in `&namdom` (namelist 3.1) namelist (DOMAINCFG variant)):

`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`.

`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 and ice shelf draft (if needed). 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 wet levels are defined (all levels are masked).

The `isfdraft_meter.nc` file (Netcdf format) provides the ice shelf draft (positive, in meters) at each grid point of the model grid. This file is only needed if `ln_isfcav=.true.`. Defining the ice shelf draft will also define the ice shelf edge and the grounding line position.

E.2.3. Choice of vertical grid

After reading the bathymetry, the algorithm for vertical grid definition differs between the different options:

`ln_zco = .true.` set a reference coordinate transformation $z_0(k)$, and set $z(i, j, k, t) = z_0(k)$ where $z_0(k)$ is the closest match to the depth at (i, j) .

`ln_zps = .true.` 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.

`ln_sco = .true.` smooth the bathymetry to fulfill the hydrostatic consistency criteria and set the three-dimensional transformation.

`s-z` and `s-zps` smooth the bathymetry to fulfill 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.

Z-coordinate with uniform thickness levels (`ln_zco`)

With this option the model topography can be fully described by the reference vertical coordinate and a 2D integer field giving the number of wet levels at each location (`bathy_level`). The resulting match to the real topography is likely to be poor though (especially with thick, deep levels) and slopes poorly represented. This option is rarely used in modern simulations but it can be useful for testing purposes.

Z-coordinate with partial step (`ln_zps`)

In z -coordinate partial step, the depths of the model levels are defined by the reference analytical function $z_0(k)$ as described in [subsection E.2.1](#), *except* in the bottom layer. The thickness of the bottom layer is allowed to vary as a function of geographical location (λ, φ) to allow a better representation of the bathymetry, especially in the case of small slopes (where the bathymetry varies by less than one level thickness from one grid point to the next). The reference layer thicknesses e_{3t}^0 have been defined in the absence of bathymetry. With partial steps, layers from 1 to `jpk - 2` can have a thickness smaller than $e_{3t}(jk)$.

The model deepest layer (`jpk - 1`) is allowed to have either a smaller or larger thickness than $e_{3t}(jpk)$: the maximum thickness allowed is $2 * e_{3t}(jpk - 1)$.

This has to be kept in mind when specifying values in `&namdom` (namelist 3.1) namelist (DOMAINCFG variant), such as 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 minimum 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 $e_{3t}(jk)$).

```

!-----
&namzgr_sco      !   s-coordinate or hybrid z-s-coordinate          (default F)
!-----
ln_s_sh94       = .false.    !   Song & Haidvogel 1994 hybrid S-sigma   (T) |
ln_s_sf12       = .false.    !   Siddorn & Furner 2012 hybrid S-z-sigma (T) | if both are false the NEMO tanh
↳ stretching is applied
ln_sigcrit      = .false.    !   use sigma coordinates below critical depth (T) or Z coordinates (F) for Siddorn &
↳ Furner stretch
!-----
!   stretching coefficients for all functions
rn_sbot_min     = 10.0       !   minimum depth of s-bottom surface (>0) (m)
rn_sbot_max     = 7000.0    !   maximum depth of s-bottom surface (= ocean depth) (>0) (m)
rn_hc           = 150.0     !   critical depth for transition to stretched coordinates
!-----
rn_rmax         = 0.3       !   maximum cut-off r-value allowed (0<r_max<1)
!-----
!-----
rn_theta        = 6.0       !   surface control parameter (0<=theta<=20)
rn_bb           = 0.8       !   stretching with SH94 s-sigma
!-----
!-----
rn_alpha        = 4.4       !   stretching with SF12 s-sigma
rn_efold        = 0.0       !   efold length scale for transition to stretched coord
rn_zs           = 1.0       !   depth of surface grid box
!-----
rn_zb_a         = 0.024    !   bathymetry scaling factor for calculating Zb
rn_zb_b         = -0.2     !   offset for calculating Zb
!-----
!-----
rn_thetb       = 1.0       !   bottom control parameter (0<=thetb<= 1)
/

```

namelist E.2.: &namzgr_sco_domcfg

S-coordinate (`ln_sco`)

Options are defined in `&namzgr_sco` (??) (DOMAINcfg only). 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:

$$z(k) = h(i, j) z_0(k)$$

$$e_3(k) = h(i, j) z'_0(k)$$

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 (item 3.5) or an envelop bathymetry can be defined (item 3.5). The namelist parameter `rn_rmax` determines the slope at which the terrain-following coordinate intersects the sea bed and becomes a pseudo z-coordinate. The coordinate can also be hybridised by specifying `rn_sbot_min` and `rn_sbot_max` as the minimum and maximum depths at which the terrain-following vertical coordinate is calculated.

Options for stretching the coordinate are provided as examples, but care must be taken to ensure that the vertical stretch used is appropriate for the application.

The original default *NEMO* s-coordinate stretching is available if neither of the other options are specified as true (`ln_s_SH94=.false.` and `ln_s_SF12=.false.`). This uses a depth independent tanh function for the stretching (Madec et al., 1996):

$$z = s_{min} + C(s)(H - s_{min})$$

where s_{min} is the depth at which the s-coordinate stretching starts and allows a z-coordinate to be placed on top of the stretched coordinate, and z is the depth (negative down from the sea surface).

$$s = -\frac{k}{n-1} \quad \text{and} \quad 0 \leq k \leq n-1$$

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

A stretching function, modified from the commonly used Song and Haidvogel (1994) stretching (`ln_s_SH94=.true.`), is also available and is more commonly used for shelf seas modelling:

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

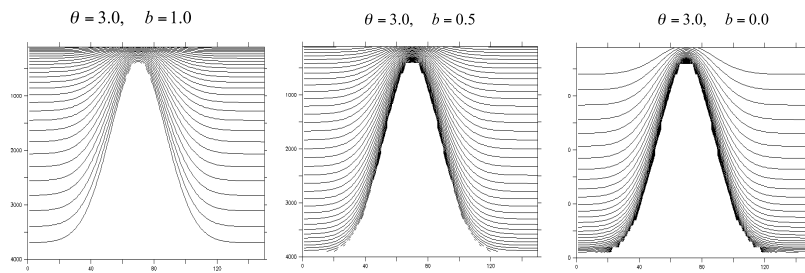


Figure E.2.: Examples of the stretching function applied to a seamount; from left to right: surface, surface and bottom, and bottom intensified resolutions

where H_c is the critical depth (`rn_hc`) at which the coordinate transitions from pure σ to the stretched coordinate, and θ (`rn_theta`) and b (`rn_bb`) are the surface and bottom control parameters such that $0 \leq \theta \leq 20$, and $0 \leq b \leq 1$. b has been designed to allow surface and/or bottom increase of the vertical resolution (figure E.2).

Another example has been provided at version 3.5 (`ln_s_SF12`) that allows a fixed surface resolution in an analytical terrain-following stretching Siddorn and Furner (2013). In this case the a stretching function γ is defined such that:

$$z = -\gamma h \quad \text{with} \quad 0 \leq \gamma \leq 1 \tag{E.7}$$

The function is defined with respect to σ , the unstretched terrain-following coordinate:

$$\gamma = A \left(\sigma - \frac{1}{2}(\sigma^2 + f(\sigma)) \right) + B (\sigma^3 - f(\sigma)) + f(\sigma)$$

Where:

$$f(\sigma) = (\alpha + 2)\sigma^{\alpha+1} - (\alpha + 1)\sigma^{\alpha+2} \quad \text{and} \quad \sigma = \frac{k}{n - 1}$$

This gives an analytical stretching of σ that is solvable in A and B as a function of the user prescribed stretching parameter α (`rn_alpha`) that stretches towards the surface ($\alpha > 1.0$) or the bottom ($\alpha < 1.0$) and user prescribed surface (`rn_zs`) and bottom depths. The bottom cell depth in this example is given as a function of water depth:

$$Z_b = ha + b$$

where the namelist parameters `rn_zb_a` and `rn_zb_b` are a and b respectively.

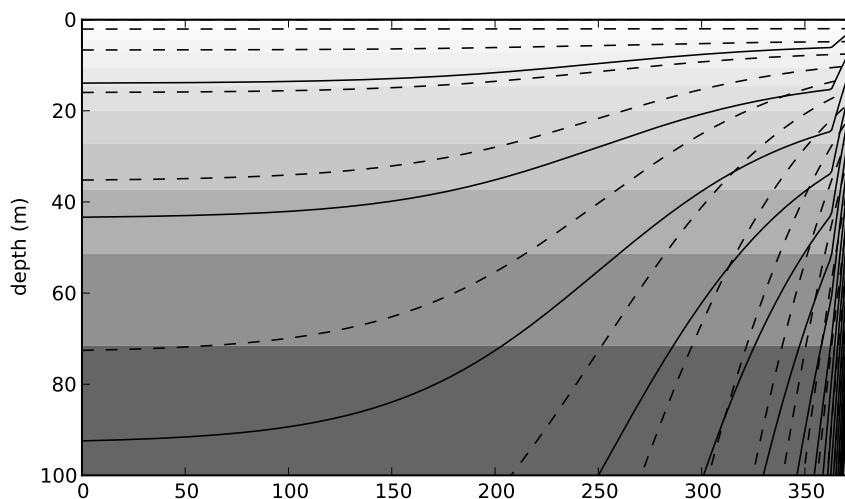


Figure E.3.: A comparison of the Song and Haidvogel (1994) S -coordinate (solid lines), a 50 level Z -coordinate (contoured surfaces) and the Siddorn and Furner (2013) S -coordinate (dashed lines) in the surface 100 m for an idealised bathymetry that goes from 50 m to 5500 m depth. For clarity every third coordinate surface is shown.

This gives a smooth analytical stretching in computational space that is constrained to given specified surface and bottom grid cell thicknesses in real space. This is not to be confused with the hybrid schemes that superimpose geopotential coordinates on terrain following coordinates thus creating a non-analytical vertical coordinate that therefore may suffer from large gradients in the vertical resolutions. This stretching is less straightforward to implement than the Song and

Haidvogel (1994) stretching, but has the advantage of resolving diurnal processes in deep water and has generally flatter slopes.

As with the Song and Haidvogel (1994) stretching the stretch is only applied at depths greater than the critical depth h_c . In this example two options are available in depths shallower than h_c , with pure sigma being applied if the `ln_sigcrit` is true and pure z-coordinates if it is false (the z-coordinate being equal to the depths of the stretched coordinate at h_c).

Minimising the horizontal slope of the vertical coordinate is important in terrain-following systems as large slopes lead to hydrostatic consistency. A hydrostatic consistency parameter diagnostic following Haney (1991) has been implemented, and is output as part of the model mesh file at the start of the run.

z^* - or s^* -coordinate (`ln_linssh`)

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



Table of contents

F.1. Introduction	250
F.2. Overview and general conventions	250
F.3. Architecture	251
F.4. Style rules	251
F.4.1. Argument list format	251
F.4.2. Array syntax	251
F.4.3. Case	251
F.4.4. Comments	252
F.4.5. Continuation lines	252
F.4.6. Declaration of arguments and local variables	252
F.4.7. F90 Standard	252
F.4.8. Free-Form Source	252
F.4.9. Indentation	253
F.4.10. Loops	253
F.4.11. Naming Conventions: files	253
F.4.12. Naming Conventions: modules	253
F.4.13. Naming Conventions: variables	253
F.4.14. Operators	253
F.4.15. Pre processor	253
F.5. Content rules	254
F.5.1. Configurations	254
F.5.2. Constants	254
F.5.3. Declaration for variables and constants	254
F.5.4. Headers	255
F.5.5. Interface blocks	255
F.5.6. I/O Error Conditions	255
F.5.7. PRINT - ASCII output files	255
F.5.8. Precision	256
F.5.9. Structures	256
F.6. Packages coding rules	256
F.6.1. Bounds checking	256
F.6.2. Communication	256
F.6.3. Error conditions	256
F.6.4. Memory management	257
F.6.5. Optimisation	258
F.6.6. Package attribute: PRIVATE, PUBLIC, USE, ONLY	258
F.6.7. Parallelism using MPI	258
F.7. Features to be avoided	258

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 (Gibson, 1986). 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.

F.1. Introduction

This document describes conventions used in *NEMO* coding and suggested for its development. The objectives are to offer a guide to all readers of the *NEMO* code, and to facilitate the work of all the developers, including the validation of their developments, and eventually the implementation of these developments within the *NEMO* platform.

A first approach of these rules can be found in the code in `./src/OCE/module_example` where all the basics coding conventions are illustrated. More details can be found below.

This work is based on the coding conventions in use for the Community Climate System Model *, the previous version of this document ("FORTRAN coding standard in the OPA System") and the expertise of the *NEMO* System Team. After a general overview below, this document will describe:

- The style rules, *i.e.* the syntax, appearance and naming conventions chosen to improve readability of the code;
- The content rules, *i.e.* the conventions to improve the reliability of the different parts of the code;
- The package rules to go a step further by improving the reliability of the whole and interfaces between routines and modules.

F.2. Overview and general conventions

NEMO has 3 major components: ocean dynamics (`./src/OCE`), sea-ice (`./src/ICE`) and marine biogeochemistry (`./src/MBG`). In each directory, one will find some FORTRAN files and/or subdirectories, one per functionality of the code: `./src/OCE/BDY` (boundaries), `./src/OCE/DIA` (diagnostics), `./src/OCE/DOM` (domain), `./src/OCE/DYN` (dynamics), `./src/OCE/LDF` (lateral diffusion), etc...

All name are chosen to be as self-explanatory as possible, in English, all prefixes are 3 digits.

English is used for all variables names, comments, and documentation.

Physical units are MKS. The only exception to this is the temperature, which is expressed in degrees Celsius, except in bulk formulae and part of SI³ sea-ice model where it is in Kelvin. See `./src/OCE/DOM/physcst.F90` files for conversions.

*UCAR conventions

F.3. Architecture

Within each directory, organisation of files is driven by orthogonality, *i.e.* one functionality of the code is intended to be in one and only one directory, and one module and all its related routines are in one file. The functional modules are:

- SBC surface module
- IOM management of the I/O
- NST interface to AGRIF (nesting model) for dynamics and biogeochemistry
- OBC, BDY management of structured and unstructured open boundaries
- C1D 1D (vertical) configuration for dynamics, sea-ice and biogeochemistry
- OFF off-line module: passive tracer or biogeochemistry alone
- ...

For example, the file *domain.F90* contains the module `domain` and all the subroutines related to this module (`dom_init`, `dom_nam`, `dom_ctl`).

F.4. Style rules

F.4.1. Argument list format

Routine argument lists will contain a maximum 5 variables per line, whilst continuation lines can be used. This applies both to the calling routine and the dummy argument list in the routine being called. The purpose is to simplify matching up the arguments between caller and callee.

```
SUBROUTINE tra_adv_eiv( kt, pun, pvn, pwn )
    CALL tra_adv_eiv( kt, zun, zvn, zwn )
```

F.4.2. Array syntax

Except for long loops (see below), array notation should be used if possible. To improve readability the array shape must be shown in brackets, *e.g.*:

```
onedarraya(:) = onedarrayb(:) + onedarrayc(:)
twodarray(:, :) = scalar * another twodarray(:, :)
```

When accessing sections of arrays, for example in finite difference equations, do so by using the triplet notation on the full array, *e.g.*:

```
twodarray(:, 2:len2) = scalar &
& * ( twodarray2(:, 1:len2-1) &
& - twodarray2(:, 2:len2) )
```

For long, complicated loops, explicitly indexed loops should be preferred. In general when using this syntax, the order of the loops indices should reflect the following scheme (for best usage of data locality):

```
DO jk = 1, jpk
  DO jj = 1, jpj
    DO ji = 1, jpi
      threedarray(ji, jj, jk) = ...
    END DO
  END DO
END DO
```

F.4.3. Case

All FORTRAN keywords are in capital: `DIMENSION`, `WRITE`, `DO`, `END DO`, `NAMelist`, ... All other parts of the *NEMO* code will be written in lower case.

F.4.9. Indentation

Code as well as comment lines within loops, if-blocks, continuation lines, **MODULE** or **SUBROUTINE** statements will be indented 3 characters for readability (except for **CONTAINS** that remains at first column).

```

MODULE mod1
  REAL(wp) xx
CONTAINS
  SUBROUTINE sub76( px, py, pz, pw, pa, &
    & pb, pc, pd, pe      )
    <instruction>
  END SUBROUTINE sub76
END MODULE mod1

```

F.4.10. Loops

Loops, if explicit, should be structured with the do-end do construct as opposed to numbered loops. Nevertheless non-numeric labels can be used for a big iterative loop of a recursive algorithm. In the case of a long loop, a self-descriptive label can be used (*i.e.* not just a number).

F.4.11. Naming Conventions: files

A file containing a module will have the same name as the module it contains (because dependency rules used by "make" programs are based on file names).[†]

F.4.12. Naming Conventions: modules

Use a meaningful English name and the "3 letters" naming convention: first 3 letters for the code section, and last 3 to describe the module. For example, *zdf*tke, where "zdf" stands for vertical diffusion, and "tke" for turbulent kinetic energy. Note that by implication multiple modules are not allowed in a single file. The use of common blocks is deprecated in FORTRAN90 and their use in *NEMO* is strongly discouraged. Modules are a better way to declare static data. Among the advantages of modules is the ability to freely mix data of various types, and to limit access to contained variables through the use of the **ONLY** and **PRIVATE** attributes.

F.4.13. Naming Conventions: variables

All variable should be named as explicitly as possible in English. The naming convention concerns prefix letters of these name, in order to identify the variable type and status.

Never use a FORTRAN keyword as a routine or variable name.

The table below lists the starting letter(s) to be used for variable naming, depending on their type and status:

F.4.14. Operators

Use of the operators `<`, `>`, `<=`, `>=`, `==`, `/=` is strongly recommended instead of their deprecated counterparts (`.lt.`, `.gt.`, `.le.`, `.ge.`, `.eq.`, `.ne.`). The motivation is readability. In general use the notation:

`< Blank >< Operator >< Blank >`

F.4.15. Pre processor

Where the use of a language pre-processor is required, it will be the C pre-processor (`cpp`).

The `cpp` key is the main feature used, allowing to ignore some useless parts of the code at compilation step.

The advantage is to reduce the memory use; the drawback is that compilation of this part of the code isn't checked.

The `cpp` key feature should only be used for a few limited options, if it reduces the memory usage. In all cases, a logical variable and a FORTRAN **IF** should be preferred. When using a `cpp` key `key_optionname`, a corresponding logical variable `lk_optionname` should be declared to allow FORTRAN **IF** tests in the code and a FORTRAN module with the same name (*i.e.* `optionname.F90`) should be defined. This module is the only place where a "if defined" command appears, selecting either the whole FORTRAN code or a dummy module. For example, the TKE vertical physics, the module name is `zdf`tke.F90, the CPP key is `key_zdf`tke and the associated logical is `lk_zdf`tke.

The following syntax:

[†]For example, if routine A "USE"s module B, then "make" must be told of the dependency relation which requires B to be compiled before A. If one can assume that module B resides in file B.o, building a tool to generate this dependency rule (*e.g.* A.o: B.o) is quite simple. Put another way, it is difficult (to say nothing of CPU-intensive) to search an entire source tree to find the file in which module B resides for each routine or module which "USE"s B.

Type / Status	integer	real	logical	character	double precision	complex
public or module variable	m n <i>but not nn_</i>	a b e f g h o q to x <i>but not fs rn_</i>	l <i>but not lp ld ll ln_</i>	c <i>but not cp cd cl cn_</i>	d <i>but not dp dd dl dn_</i>	y <i>but not yp yd yl</i>
dummy argument	k <i>but not kf</i>	p <i>but not pp pf</i>	ld	cd	dd	yd
local variable	i	z	ll	cl	cd	yl
loop control	j <i>but not jp</i>					
parameter	jp	pp	lp	cp	dp	yp
namelist	nn_	rn_	ln_	cn_	dn_	
CPP macro	kf	sf				

```
#if defined key_optionname
!! Part of code conditionally compiled if cpp key key_optionname is active
#endif
```

Is to be used rather than the `#ifdef` abbreviate form since it may have conflicts with some Unix scripts.

Tests on cpp keys included in *NEMO* at compilation step:

- The CPP keys used are compared to the previous list of cpp keys (the compilation will stop if trying to specify a non-existing key)
- If a change occurs in the CPP keys used for a given experiment, the whole compilation phase is done again.

F.5. Content rules

F.5.1. Configurations

The configuration defines the domain and the grid on which *NEMO* is running. It may be useful to associate a CPP key and some variables to a given configuration, although the part of the code changed under each of those keys should be minimized. As an example, the "ORCA2" configuration (global ocean, 2 degrees grid size) is associated with the cpp key `key_orca2` for which

```
cp_cfg = "orca"
jp_cfg = 2
```

F.5.2. Constants

Physical constants (*e.g.* π , gas constants) must never be hard-wired into the executable portion of a code. Instead, a mnemonically named variable or parameter should be set to the appropriate value, in the setup routine for the package. We realize that many parameterizations rely on empirically derived constants or fudge factors, which are not easy to name. In these cases it is not forbidden to leave such factors coded as "magic numbers" buried in executable code, but comments should be included referring to the source of the empirical formula. Hard-coded numbers should never be passed through argument lists.

F.5.3. Declaration for variables and constants

Rules

Variables used as constants should be declared with attribute `PARAMETER` and used always without copying to local variables, in order to prevent from using different values for the same constant or changing it accidentally.

- Usage of the `DIMENSION` statement or attribute is required in declaration statements

- The “:” notation is quite useful to show that this program unit declaration part is written in standard FORTRAN syntax, even if there are no attributes to clarify the declaration section. Always use the notation <blank>::<three blanks> to improve readability.
- Declare the length of a character variable using the **CHARACTER** (len=xxx) syntax [‡]
- For all global data (in contrast to module data, that is all data that can be access by other module) must be accompanied with a comment field on the same line [§]. For example:

```
REAL(wp), DIMENSION(jpi,jpj,jpk) :: ua ! i-horizontal velocity (m/s)
```

Implicit None

All subroutines and functions will include an **IMPLICIT NONE** statement. Thus all variables must be explicitly typed. It also allows the compiler to detect typographical errors in variable names. For modules, one **IMPLICIT NONE** statement in the modules definition section is needed. This also removes the need to have **IMPLICIT NONE** statements in any routines that are **CONTAINS**'ed in the module. Improper data initialisation is another common source of errors [¶]. To avoid problems, initialise variables as close as possible to where they are first used.

Attributes

PRIVATE / PUBLIC: All resources of a module are **PUBLIC** by default. A reason to store multiple routines and their data in a single module is that the scope of the data defined in the module can be limited to the routines which are in the same module. This is accomplished with the **PRIVATE** attribute.

INTENT: All dummy arguments of a routine must include the **INTENT** clause in their declaration in order to improve control of variables in routine calls.

F.5.4. Headers

Prologues are not used in *NEMO* for now, although it may become an interesting tool in combination with ProTeX auto documentation script in the future. Rules to code the headers and layout of a module or a routine are illustrated in the example module available with the code: `./src/OCE/module_example`

F.5.5. Interface blocks

Explicit interface blocks are required between routines if optional or keyword arguments are to be used. They also allow the compiler to check that the type, shape and number of arguments specified in the **CALL** are the same as those specified in the subprogram itself. FORTRAN 95 compilers can automatically provide explicit interface blocks for routines contained in a module.

F.5.6. I/O Error Conditions

I/O statements which need to check an error condition will use the `iostat=<integer variable>` construct instead of the outmoded `end=` and `err=`.

Note that a 0 value means success, a positive value means an error has occurred, and a negative value means the end of record or end of file was encountered.

F.5.7. PRINT - ASCII output files

Output listing and errors are directed to `numout` logical unit =6 and produces a file called *ocean.output* (use `ln_prt` to have one output per process in MPP). Logical `lwp` variable allows for less verbose outputs. To output an error from a routine, one can use the following template:

```
IF( nstop /= 0 .AND. lwp ) THEN ! error print
  WRITE(numout,cform_err)
  WRITE(numout,*) nstop, ' error have been found'
ENDIF
```

[‡]The len specifier is important because it is possible to have several kinds for characters (e.g. Unicode using two bytes per character, or there might be a different kind for Japanese e.g. NEC).

[§]This allows a easy research of where and how a variable is declared using the unix command: “`grep var *90 | grep !:`”.

[¶]A variable could contain an initial value you did not expect. This can happen for several reasons, e.g. the variable has never been assigned a value, its value is outdated, memory has been allocated for a pointer but you have forgotten to initialise the variable pointed to.

F.5.8. Precision

Parameterizations should not rely on vendor-supplied flags to supply a default floating point precision or integer size. The F95 `KIND` feature should be used instead. In order to improve portability between 32 and 64 bit platforms, it is necessary to make use of kinds by using a specific module `./src/OCE/par_kind.F90` declaring the "kind definitions" to obtain the required numerical precision and range as well as the size of `INTEGER`. It should be noted that numerical constants need to have a suffix of `_kindvalue` to have the according size.

Thus `wp` being the "working precision" as declared in `./src/OCE/par_kind.F90`, declaring real array `zpc` will take the form:

```
REAL(wp), DIMENSION(jpi,jpj,jpk) :: zpc      ! power consumption
```

F.5.9. Structures

The `TYPE` structure allowing to declare some variables is more often used in *NEMO*, especially in the modules dealing with reading fields, or interfaces. For example:

```
! Definition of a tracer as a structure
TYPE PTRACER
  CHARACTER(len = 20) :: sname ! short name
  CHARACTER(len = 80) :: lname ! long name
  CHARACTER(len = 20) :: unit  ! unit
  LOGICAL              :: lini  ! read in a file or not
  LOGICAL              :: lsav  ! ouput the tracer or not
END TYPE PTRACER

TYPE(PTRACER), DIMENSION(jptr) :: tracer
```

Missing rule on structure name??

F.6. Packages coding rules

F.6.1. Bounds checking

NEMO is able to run when an array bounds checking option is enabled (provided the `cpp` key `key_vectopt_loop` is not defined).

Thus, constructs of the following form are disallowed:

```
REAL(wp) :: arr(1)
```

where "arr" is an input argument into which the user wishes to index beyond 1. Use of the (*) construct in array dimensioning is forbidden also because it effectively disables array bounds checking.

F.6.2. Communication

A package should refer only to its own modules and subprograms and to those intrinsic functions included in the Fortran standard.

All communication with the package will be through the argument list or namelist input. ^{||}

F.6.3. Error conditions

When an error condition occurs inside a package, a message describing what went wrong will be printed (see `PRINT - ASCII` output files). The name of the routine in which the error occurred must be included. It is acceptable to terminate execution within a package, but the developer may instead wish to return an error flag through the argument list, see *stpctl.F90*.

^{||} The point behind this rule is that packages should not have to know details of the surrounding model data structures, or the names of variables outside of the package. A notable exception to this rule is model resolution parameters. The reason for the exception is to allow compile-time array sizing inside the package. This is often important for efficiency.


```

    RETURN
  END IF
  !
  zwrk1 => wrk_3d_5(1:10,1:10,1:10)
  ...
END SUBROUTINE sub

```

Here, instead of “use associating” the variable `zwrk1` with the array `wrk_3d_5` (as in the first example), it is explicitly declared as a pointer to a 3D array. It is then associated with a sub-array of `wrk_3d_5` once the call to `wrk_in_use()` has completed successfully. Note that in F95 (to which *NEMO* conforms) it is not possible for either the upper or lower array bounds of the pointer object to differ from those of the target array.

In addition to the **REAL** (`KIND = wp`) workspace arrays, `wrk_nemo.F90` also contains 2D integer arrays and 2D REAL arrays with extent (`jpi, jpk`), *i.e.* `xz`. The utility routines for the integer workspaces are `iwrk_in_use()` and `iwrk_not_released()` while those for the `xz` workspaces are `wrk_in_use_xz()` and `wrk_not_released_xz()`.

Should a call to one of the `wrk_in_use()` family of utilities fail, an error message is printed along with a table showing which of the workspace arrays are currently in use. This should enable the developer to choose alternatives for use in the subroutine being worked on.

When compiling *NEMO* for production runs, the calls to `wrk_in_use()` / `wrk_not_released()` can be reduced to stubs that just return `.false.` by setting the cpp key `key_no_workspace_check`. These stubs may then be in-lined (and thus effectively removed altogether) by setting appropriate compiler flags (*e.g.* “`-finline`” for the Intel compiler or “`-Q`” for the IBM compiler).

F.6.5. Optimisation

Considering the new computer architecture, optimisation cannot be considered independently from the computer type. In *NEMO*, portability is a priority, before any too specific optimisation.

Some tools are available to help: for vector computers, `key_vectopt_loop` allows to unroll a loop

F.6.6. Package attribute: PRIVATE, PUBLIC, USE, ONLY

Module variables and routines should be encapsulated by using the **PRIVATE** attribute. What shall be used outside the module can be declared **PUBLIC** instead. Use **USE** with the **ONLY** attribute to specify which of the variables, type definitions etc... defined in a module are to be made available to the using routine.

F.6.7. Parallelism using MPI

NEMO is written in order to be able to run on one processor, or on one or more using MPI (*i.e.* activating the cpp key `key_mpp_mpi`). The domain decomposition divides the global domain in cubes (see *NEMO* reference manual). Whilst coding a new development, the MPI compatibility has to be taken in account (see `./src/LBC/lib_mpp.F90`) and should be tested. By default, the `x-z` part of the decomposition is chosen to be as square as possible. However, this may be overridden by specifying the number of sub-domains in latitude and longitude in the `nammpp` section of the namelist file.

F.7. Features to be avoided

The code must follow the current standards of FORTRAN and ANSI C. In particular, the code should not produce any WARNING at compiling phase, so that users can be easily alerted of potential bugs when some appear in their new developments. Below is a list of features to avoid:

- **COMMON** block (use the declaration part of **MODULE** instead)
- **EQUIVALENCE** (use **POINTER** or derived data type instead to form data structure)
- Assigned and computed **GOTO** (use the **CASE** construct instead)
- Arithmetic **IF** statement (use the block **IF**, **ELSE**, **ELSEIF**, **ENDIF** or **SELECT CASE** construct instead)
- Labelled **DO** construct (use unlabelled **END DO** instead)
- **FORMAT** statement (use character parameters or explicit format- specifiers inside the **READ** or **WRITE** statement instead)
- **GOTO** and **CONTINUE** statements (use **IF**, **CASE**, **DO WHILE**, **EXIT** or **CYCLE** statements or a contained ?)

- **PAUSE**
- **ENTRY** statement: a sub-program must only have one entry point.
- **RETURN** is obsolete and so not necessary at the end of program units
- **FUNCTION** statement
- Avoid functions with side effects. **
- **DATA** and **BLOCK DATA** (use initialisers)

**First, the code is easier to understand, if you can rely on the rule that functions don't change their arguments. Second, some compilers generate more efficient code for PURE functions (in FORTRAN 95 there are the attributes PURE and ELEMENTAL), because they can store the arguments in different places. This is especially important on massive parallel and as well on vector machines.

Bibliography

- Adcroft, A. and J.-M. Campin, 2004: Rescaled height coordinates for accurate representation of free-surface flows in ocean circulation models. *Ocean Modelling*, **7** (3-4), 269–284, [doi](#).
- Arakawa, A. and Y.-J. G. Hsu, 1990: Energy conserving and potential-enzrophy dissipating schemes for the shallow water equations. *Monthly Weather Review*, **118** (10), 1960–1969, [doi](#).
- Arakawa, A. and V. R. Lamb, 1981: A potential enstrophy and energy conserving scheme for the shallow water equations. *Monthly Weather Review*, **109** (1), 18–36, [doi](#).
- Arbic, B. K., S. T. Garner, R. W. Hallberg, and H. L. Simmons, 2004: The accuracy of surface elevations in forward global barotropic and baroclinic tide models. *Deep Sea Research*, **51** (25-26), 3069–3101, [doi](#).
- Asselin, R., 1972: Frequency filter for time integrations. *Monthly Weather Review*, **100** (6), 487–490, [doi](#).
- Axell, L. B., 2002: Wind-driven internal waves and langmuir circulations in a numerical ocean model of the southern baltic sea. *Journal of Geophysical Research*, **107** (C11), [doi](#).
- Barnier, B., G. Madec, T. Penduff, J.-M. Molines, A.-M. Tréguier, J. Le Sommer, A. Beckmann, A. Biastoch, C. Böning, J. Dengg, C. Derval, E. Durand, S. Gulev, E. Remy, C. Talandier, S. Theetten, M. Maltrud, J. McClean, and B. de Cuevas, 2006: Impact of partial steps and momentum advection schemes in a global ocean circulation model at eddy-permitting resolution. *Ocean Dynamics*, **56** (5-6), 543–567, [doi](#).
- Beckmann, A. and R. Döscher, 1997: A method for improved representation of dense water spreading over topography in geopotential-coordinate models. *Journal of Physical Oceanography*, **27** (4), 581–591, [doi](#).
- Beckmann, A. and H. Goosse, 2003: A parameterization of ice shelf-ocean interaction for climate models. *Ocean Modelling*, **5** (2), 157–170, [doi](#).
- Beckmann, A. and D. B. Haidvogel, 1993: Numerical simulation of flow around a tall isolated seamount. part I: Problem formulation and model accuracy. *Journal of Physical Oceanography*, **23** (8), 1736–1753, [doi](#).
- Beljaars, A. C. M., 1995: The parametrization of surface fluxes in large-scale models under free convection. *Quarterly Journal of the Royal Meteorological Society*, **121** (522), 255–270, [doi](#).
- Bernie, D. J., E. Guilyardi, G. Madec, J. M. Slingo, and S. J. Woolnough, 2007: Impact of resolving the diurnal cycle in an ocean-atmosphere GCM. part 1: a diurnally forced OGCM. *Climate Dynamics*, **29** (6), 575–590, [doi](#).
- Bernie, D. J., S. J. Woolnough, J. M. Slingo, and E. Guilyardi, 2005: Modeling diurnal and intraseasonal variability of the ocean mixed layer. *Journal of Climate*, **18** (8), 1190–1202, [doi](#).
- Blanke, B. and P. Delécluse, 1993: Variability of the tropical atlantic ocean simulated by a general circulation model with two different mixed-layer physics. *Journal of Physical Oceanography*, **23** (7), 1363–1388, [doi](#).
- Blanke, B. and S. Raynaud, 1997: Kinematics of the pacific equatorial undercurrent: An eulerian and lagrangian approach from GCM results. *Journal of Physical Oceanography*, **27** (6), 1038–1053, [doi](#).
- Bloom, S. C., L. L. Takacs, A. M. D. Silva, and D. Ledvina, 1996: Data assimilation using incremental analysis updates. *Monthly Weather Review*, **124** (6), 1256–1271, [doi](#).
- Bouffard, D. and L. Boegman, 2013: A diapycnal diffusivity model for stratified environmental flows. *Dynamics of Atmospheres and Oceans*, **61-62**, 14–34, [doi](#).
- Bougeault, P. and P. Lacarrere, 1989: Parameterization of orography-induced turbulence in a mesobeta-scale model. *Monthly Weather Review*, **117** (8), 1872–1890, [doi](#).
- Brankart, J.-M., 2013: Impact of uncertainties in the horizontal density gradient upon low resolution global ocean modelling. *Ocean Modelling*, **66**, 64–76, [doi](#).
- Brankart, J.-M., G. Candille, F. Garnier, C. Calone, A. Melet, P.-A. Bouttier, P. Brasseur, and J. Verron, 2015: A generic approach to explicit simulation of uncertainty in the NEMO ocean model. *Geoscientific Model Development*, **8** (5), 1285–1297, [doi](#).
- Breivik, Ø., J.-R. Bidlot, and P. A. E. M. Janssen, 2016: A stokes drift approximation based on the phillips spectrum. *Ocean Modelling*, **100**, 49–56, [doi](#).
- Breivik, Ø., P. A. E. M. Janssen, and J.-R. Bidlot, 2014: Approximate stokes drift profiles in deep water. *Journal of Physical Oceanography*, **44** (9), 2433–2445, [doi](#).
- Brodeau, L., B. Barnier, S. K. Gulev, and C. Woods, 2016: Climatologically significant effects of some approximations in the bulk parameterizations of turbulent airsea fluxes. *Journal of Physical Oceanography*, **47** (1), 5–28, [doi](#).

- Brodeau, L., B. Barnier, A.-M. Tréguier, T. Penduff, and S. Gulev, 2010: An ERA40-based atmospheric forcing for global ocean circulation models. *Ocean Modelling*, **31** (3-4), 88–104, [doi](#).
- Brown, J. A. and K. A. Campana, 1978: An economical time-differencing system for numerical weather prediction. *Monthly Weather Review*, **106** (8), 1125–1136, [doi](#).
- Burchard, H., 2002: Energy-conserving discretisation of turbulent shear and buoyancy production. *Ocean Modelling*, **4** (3-4), 347–361, [doi](#).
- Campin, J.-M., A. Adcroft, C. Hill, and J. Marshall, 2004: Conservation of properties in a free-surface model. *Ocean Modelling*, **6** (3-4), 221–244, [doi](#).
- Campin, J.-M. and H. Goosse, 1999: Parameterization of density-driven downsloping flow for a coarse-resolution ocean model in z-coordinate. *Tellus A*, **51** (3), 412–430, [doi](#).
- Campin, J.-M., J. Marshall, and D. Ferreira, 2008: Sea ice-ocean coupling using a rescaled vertical coordinate z^* . *Ocean Modelling*, **24** (1-2), 1–14, [doi](#).
- Canuto, V. M., A. Howard, Y. Cheng, and M. S. Dubovikov, 2001: Ocean turbulence. part I: One-point closure model-momentum and heat vertical diffusivities. *Journal of Physical Oceanography*, **31** (6), 1413–1426, [doi](#).
- Chanut, J., 2005: Nesting code for NEMO. Tech. rep., European Union: Marine Environment and Security for the European Area (MERSEA) Integrated Project, [doi](#).
- Chassignet, E. P., L. T. Smith, G. R. Halliwell, and R. Bleck, 2003: North atlantic simulations with the hybrid coordinate ocean model (HYCOM): Impact of the vertical coordinate choice, reference pressure, and thermobaricity. *Journal of Physical Oceanography*, **33** (12), 2504–2526, [doi](#).
- Cox, M. D., 1987: Isopycnal diffusion in a z-coordinate ocean model. *Ocean Modelling*, **74**, 1–9, [URL](#).
- Craig, P. D. and M. L. Banner, 1994: Modeling wave-enhanced turbulence in the ocean surface layer. *Journal of Physical Oceanography*, **24** (12), 2546–2559, [doi](#).
- Craik, A. D. D. and S. Leibovich, 1976: A rational model for Langmuir circulations. *Journal of Fluid Mechanics*, **73** (3), 401–426, [doi](#).
- D'Alessio, S. J. D., K. Abdella, and N. A. McFarlane, 1998: A new second-order turbulence closure scheme for modeling the oceanic mixed layer. *Journal of Physical Oceanography*, **28** (8), 1624–1641, [doi](#).
- Daley, R. and E. Barker, 2001: Defense Technical Information Center, 163 pp., [doi](#).
- Danabasoglu, G., R. Ferrari, and J. C. McWilliams, 2008: Sensitivity of an ocean general circulation model to a parameterization of near-surface eddy fluxes. *Journal of Climate*, **21** (6), 1192–1208, [doi](#).
- Davies, H. C., 1976: A lateral boundary formulation for multi-level prediction models. *Quarterly Journal of the Royal Meteorological Society*, **102** (432), 405–418, [doi](#).
- de Boyer Montégut, C., G. Madec, A. S. Fischer, A. Lazar, and D. Iudicone, 2004: Mixed layer depth over the global ocean: An examination of profile data and a profile-based climatology. *Journal of Geophysical Research*, **109** (C12), C12003, [doi](#).
- de Lavergne, C., G. Madec, J. Le Sommer, A. J. G. Nurser, and A. C. N. Garabato, 2016: The impact of a variable mixing efficiency on the abyssal overturning. *Journal of Physical Oceanography*, **46** (2), 663–681, [doi](#).
- Debreu, L., C. Vouland, and E. Blayo, 2008: AGRIF: Adaptive grid refinement in fortran. *Computers & Geosciences*, **34** (1), 8–13, [doi](#).
- Delécluse, P. and G. Madec, 1999: Ocean modelling and the role of the ocean in the climate system. *Modeling the Earth's Climate and its Variability*, Holland, W. R., S. Joussaume, and F. David, Eds., North Holland, Les Houches, Vol. 67, 237–313.
- Demange, J., 2014: Advection and gravity waves propagation numerical schemes for oceanic circulation models. Ph.D. thesis, Grenoble University, France, [URL](#).
- Dobricic, S., N. Pinardi, M. Adani, M. Tonani, C. Fratianni, A. Bonazzi, and V. Fernandez, 2007: Daily oceanographic analyses by mediterranean forecasting system at the basin scale. *Ocean Science*, **3** (1), 149–157, [doi](#).
- Drijfhout, S. S., 1994: Heat transport by mesoscale eddies in an ocean circulation model. *Journal of Physical Oceanography*, **24** (2), 353–369, [doi](#).
- Dukowicz, J. K. and R. D. Smith, 1994: Implicit free-surface method for the bryan-cox-semtner ocean model. *Journal of Geophysical Research*, **99** (C4), 7991–8014, [doi](#).
- Edson, J. B., V. Jampana, R. A. Weller, S. P. Bigorre, A. J. Plueddemann, C. W. Fairall, S. D. Miller, L. Mahrt, D. Vickers, and H. Hersbach, 2013: On the exchange of momentum over the open ocean. *Journal of Physical Oceanography*, **43** (8), 1589–1610, [doi](#).
- Eiseman, P. R. and A. P. Stone, 1980: Conservation laws of fluid dynamics - A survey. *SIAM Review*, **22** (1), 12–27, [doi](#).
- Emile-Geay, J. and G. Madec, 2009: Geothermal heating, diapycnal mixing and the abyssal circulation. *Ocean Science*, **5** (2), 203–217, [doi](#).
- Engedahl, H., 1995: Use of the flow relaxation scheme in a three-dimensional baroclinic ocean model with realistic topography. *Tellus A*, **47** (3), 365–382, [doi](#).
- Fairall, C. W., E. F. Bradley, J. E. Hare, A. A. Grachev, and J. B. Edson, 2003: Bulk parameterization of air-sea fluxes: Updates and verification for the coare algorithm. *Journal of Climate*, **16** (4), 571–591, [doi](#).
- Farge Coulombier, M., 1987: Dynamique non-linéaire des ondes et des tourbillons dans les équations de saint-venant. Ph.D. thesis, Paris VI University, France, [URL](#).
- Farrow, D. E. and D. P. Stevens, 1995: A new tracer advection scheme for bryan and cox type ocean general circulation models. *Journal of Physical Oceanography*, **25** (7), 1731–1741, [doi](#).
- Flather, R. A., 1994: A storm surge prediction model for the northern bay of bengal with application to the cyclone disaster in april 1991. *Journal of Physical Oceanography*, **24** (1), 172–190, [doi](#).
- Fofonoff, N. P. and R. C. Millard, 1983: Technical Paper in Marine Science, Vol. 44. UNESCO, 53 pp., [URL](#).
- Fox-Kemper, B., R. Ferrari, and B. Hallberg, 2008: Parameterization of mixed layer eddies. part i: Theory and diagnosis. *Journal of Physical Oceanography*, **38** (6), 1145–1165, [doi](#).

- Galperin, B., L. H. Kantha, S. Hassid, and A. Rosati, 1988: A quasi-equilibrium turbulent energy model for geophysical flows. *Journal of the Atmospheric Sciences*, **45** (1), 55–62, [doi](#).
- Gargett, A. E., 1984: Vertical eddy diffusivity in the ocean interior. *Journal of Marine Research*, **42** (2), 359–393, [doi](#).
- Gaspar, P., Y. Grégoris, and J.-M. Lefevre, 1990: A simple eddy kinetic energy model for simulations of the oceanic vertical mixing: Tests at station papa and long-term upper ocean study site. *Journal of Geophysical Research*, **95** (C9), 16 179, [doi](#).
- Gent, P. R. and J. C. McWilliams, 1990: Isopycnal mixing in ocean circulation models. *Journal of Physical Oceanography*, **20** (1), 150–155, [doi](#).
- Gerdes, R., 1993a: A primitive equation ocean circulation model using a general vertical coordinate transformation: 1. description and testing of the model. *Journal of Geophysical Research*, **98** (C8), 14 683–14 701, [doi](#).
- , 1993b: A primitive equation ocean circulation model using a general vertical coordinate transformation: 2. application to an overflow problem. *Journal of Geophysical Research*, **98** (C8), 14 703–14 726, [doi](#).
- Gerdes, R., C. Köberle, and J. Willebrand, 1991: The influence of numerical advection schemes on the results of ocean general circulation models. *Climate Dynamics*, **5** (4), 211–226, [doi](#).
- Gibson, J. K., 1986: Standards for software development and maintenance. Tech. Rep. 120, ECMWF Operations Department; Reading, United Kingdom, [doi](#).
- Gill, A. E., 1982: International Geophysics, Vol. 30. Elsevier, 662 pp., [doi](#).
- Goff, J. A., 2010: Global prediction of abyssal hill root-mean-square heights from small-scale altimetric gravity variability. *Journal of Geophysical Research*, **115** (B12), B12 104, [doi](#).
- Graham, F. S. and T. J. McDougall, 2013: Quantifying the non-conservative production of conservative temperature, potential temperature, and entropy. *Journal of Physical Oceanography*, **43** (5), 838–862, [doi](#).
- Greatbatch, R. J., 1994: A note on the representation of steric sea level in models that conserve volume rather than mass. *Journal of Geophysical Research*, **99** (C6), 12 767–12 771, [doi](#).
- Griffies, S. M., 1998: The gent-mcwilliams skew-flux. *Journal of Physical Oceanography*, **28** (5), 831–841, [doi](#).
- , 2004: Princeton University Press, 434 pp., [doi](#).
- Griffies, S. M., A. Gnanadesikan, R. C. Pacanowski, V. D. Larichev, J. K. Dukowicz, and R. D. Smith, 1998: Isonutral diffusion in a z-coordinate ocean model. *Journal of Physical Oceanography*, **28** (5), 805–830, [doi](#).
- Griffies, S. M., R. C. Pacanowski, M. Schmidt, and V. Balaji, 2001: Tracer conservation with an explicit free surface method for z-coordinate ocean models. *Monthly Weather Review*, **129** (5), 1081–1098, [doi](#).
- Guilyardi, E., G. Madec, and L. Terray, 2001: The role of lateral ocean physics in the upper ocean thermal balance of a coupled ocean-atmosphere GCM. *Climate Dynamics*, **17** (8), 589–599, [doi](#).
- Haney, R. L., 1991: On the pressure gradient force over steep topography in sigma coordinate ocean models. *Journal of Physical Oceanography*, **21** (4), 610–619, [doi](#).
- Hazeleger, W. and S. S. Drijfhout, 1998: Mode water variability in a model of the subtropical gyre: response to anomalous forcing. *Journal of Physical Oceanography*, **28** (2), 266–288, [doi](#).
- , 1999: Stochastically forced mode water variability. *Journal of Physical Oceanography*, **29** (8), 1772–1786, [doi](#).
- , 2000a: Eddy subduction in a model of the subtropical gyre. *Journal of Physical Oceanography*, **30** (4), 677–695, [doi](#).
- , 2000b: A model study on internally generated variability in subtropical mode water formation. *Journal of Geophysical Research*, **105** (C6), 13 965–13 979, [doi](#).
- He, Y. and C. H. Q. Ding, 2001: Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications. *The Journal of Supercomputing*, **18** (3), 259–277, [doi](#).
- Hirt, C. W., A. A. Amsden, and J. L. Cook, 1974: An arbitrary lagrangian-eulerian computing method for all flow speeds. *Journal of Computational Physics*, **14** (3), 227–253, [doi](#).
- Hofmeister, R., H. Burchard, and J.-M. Beckers, 2010: Non-uniform adaptive vertical grids for 3D numerical ocean models. *Ocean Modelling*, **33** (1-2), 70–86, [doi](#).
- Holland, D. M. and A. Jenkins, 1999: Modeling thermodynamic ice-ocean interactions at the base of an ice shelf. *Journal of Physical Oceanography*, **29** (8), 1787–1800, [doi](#).
- Hunter, J. R., 2006: Specification for test models of ice shelf cavities. Tech. Rep. 7, Antarctic Climate & Ecosystems Cooperative Research Centre, URL.
- IOC, S. and IAPSO, 2010: *The international thermodynamic equation of seawater - 2010: Calculation and use of thermodynamic properties*, Manuals and Guides, Vol. 56. UNESCO, 196 pp., URL.
- Jackson, P. R. and C. R. Rehmann, 2014: Experiments on differential scalar mixing in turbulence in a sheared, stratified flow. *Journal of Physical Oceanography*, **44** (10), 2661–2680, [doi](#).
- Janssen, P. A. E. M., Ø. Breivik, K. Mogensen, F. Vitart, M. Alonso-Balmaseda, J.-R. Bidlot, S. Keeley, M. Leutbecher, L. Magnusson, and F. Molteni, 2013: Air-sea interaction and surface waves. Tech. rep., ECMWF Research Department; Reading, United Kingdom, [doi](#).
- Jenkins, A., 1991: A one-dimensional model of ice shelf-ocean interaction. *Journal of Geophysical Research*, **96** (C11), 20 671–20 677, [doi](#).
- Jenkins, A., K. W. Nicholls, and H. F. J. Corr, 2010: Observation and parameterization of ablation at the base of ronne ice shelf, antarctica. *Journal of Physical Oceanography*, **40** (10), 2298–2312, [doi](#).
- Kantha, L. and S. Carniel, 2003: Comments on “A generic length-scale equation for geophysical turbulence models” by L. umlauf and H. burchard. *Journal of Marine Research*, **61** (5), 693–702, [doi](#).
- Kantha, L. H. and C. A. Clayson, 1994: An improved mixed layer model for geophysical applications. *Journal of Geophysical Research*, **99** (C12), 25 235–25 266, [doi](#).

- Kasahara, A., 1974: Various vertical coordinate systems used for numerical weather prediction. *Monthly Weather Review*, **102** (7), 509–522, [doi](#).
- Killworth, P. D., 1989: On the parameterization of deep convection in ocean models. *Parameterization of small-scale processes*, Muller, P. and D. Henderson, Eds., Hawaii Institute of Geophysics, 59–74, [URL](#).
- , 1992: An equivalent-barotropic mode in the fine resolution antarctic model. *Journal of Physical Oceanography*, **22** (11), 1379–1387, [doi](#).
- Killworth, P. D., D. J. Webb, D. Stainforth, and S. M. Paterson, 1991: The development of a free-surface bryan-cox-semtner ocean model. *Journal of Physical Oceanography*, **21** (9), 1333–1348, [doi](#).
- Kolmogorov, A. N., 1942: Equations of turbulent motion in an incompressible fluid. *Izvestiya Akademiyi Nauk SSSR*, **6**, 56–58.
- Kraus, E. and J. Turner, 1967: A one dimensional model of the seasonal thermocline II. The general theory and its consequences. *Tellus*, **19**, 98–106.
- Large, W. G., J. C. McWilliams, and S. C. Doney, 1994: Oceanic vertical mixing: A review and a model with a nonlocal boundary layer parameterization. *Reviews of Geophysics*, **(4)**, 363–403, [doi](#).
- Large, W. G. and S. G. Yeager, 2004: Diurnal to decadal global forcing for ocean and sea-ice models: the data sets and flux climatologies. Tech. rep., NCAR Climate and global dynamics division; Boulder, CO, United States, [doi](#).
- Lazar, A., 1997: La branche froide de la circulation thermohaline - sensibilité à la diffusion turbulente dans un modèle de circulation générale idéalisée. Ph.D. thesis, Université Pierre et Marie Curie, Paris, France, [URL](#).
- Lazar, A., G. Madec, and P. Delécluse, 1999: The deep interior downwelling, the veronis effect, and mesoscale tracer transport parameterizations in an OGCM. *Journal of Physical Oceanography*, **29** (11), 2945–2961, [doi](#).
- Le Sommer, J., T. Penduff, S. Theetten, G. Madec, and B. Barnier, 2009: How momentum advection schemes influence current-topography interactions at eddy permitting resolution. *Ocean Modelling*, **29** (1), 1–14, [doi](#).
- Leclair, M., 2010: Introduction d'une coordonnée verticale arbitrairement lagrangienne eulérienne dans le code d'océan NEMO. Ph.D. thesis, Université Pierre et Marie Curie, Paris, France, [URL](#).
- Leclair, M. and G. Madec, 2009: A conservative leapfrog time stepping method. *Ocean Modelling*, **30** (2-3), 88–94, [doi](#).
- , 2011: \tilde{z} -coordinate, an arbitrary lagrangian-eulerian coordinate separating high and low frequency motions. *Ocean Modelling*, **37** (3-4), 139–152, [doi](#).
- Lele, S. K., 1992: Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, **103** (1), 16–42, [doi](#).
- Lemarié, F., L. Debreu, G. Madec, J. Demange, J. M. Molines, and M. Honnorat, 2015: Stability constraints for oceanic numerical models: implications for the formulation of time and space discretizations. *Ocean Modelling*, **92**, 124–148, [doi](#).
- Lemarié, F., L. Debreu, A. F. Shchepetkin, and J. C. McWilliams, 2012: On the stability and accuracy of the harmonic and bi-harmonic isoneutral mixing operators in ocean models. *Ocean Modelling*, **52-53**, 9–35, [doi](#).
- Lengaigne, M., G. Madec, G. Alory, and C. Menkes, 2003: Impact of isopycnal mixing on the tropical ocean circulation. *Journal of Geophysical Research*, **108** (C11), 3345, [doi](#).
- Lengaigne, M., C. Menkes, O. Aumont, T. Gorgues, L. Bopp, J.-M. André, and G. Madec, 2007: Influence of the oceanic biology on the tropical pacific climate in a coupled general circulation model. *Climate Dynamics*, **28** (5), 503–516, [doi](#).
- Leonard, B. P., 1979: A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer Methods in Applied Mechanics and Engineering*, **19** (1), 59–98, [doi](#).
- , 1991: The ULTIMATE conservative difference scheme applied to unsteady one-dimensional advection. *Computer Methods in Applied Mechanics and Engineering*, **88** (1), 17–74, [doi](#).
- Lermusiaux, P. F. J., 2001: Evolving the subspace of the three-dimensional multiscale ocean variability: Massachusetts bay. *Journal of Marine Systems*, **29** (1-4), 385–422, [doi](#).
- Lievier, B., A.-M. Tréguier, G. Madec, and V. Garnier, 2007: Free surface and variable volume in the NEMO code. Tech. rep., European Union: Marine Environment and Security for the European Area (MERSEA) Integrated Project, [doi](#).
- Lévy, M., A. Estublier, and G. Madec, 2001: Choice of an advection scheme for biogeochemical models. *Geophysical Research Letters*, **28** (19), 3725–3728, [doi](#).
- Lévy, M., P. Klein, A.-M. Tréguier, D. Iovino, G. Madec, S. Masson, and K. Takahashi, 2010: Modifications of gyre circulation by sub-mesoscale physics. *Ocean Modelling*, **34** (1-2), 1–15, [doi](#).
- Li, M. and C. Garrett, 1993: Cell merging and the jet/downwelling ratio in langmuir circulation. *Journal of Marine Research*, **51** (4), 737–769, [doi](#).
- Losch, M., 2008: Modeling ice shelf cavities in a z coordinate ocean general circulation model. *Journal of Geophysical Research*, **113** (C8), [doi](#).
- Lott, F., G. Madec, and J. Verron, 1990: Topographic experiments in an ocean general circulation model. *Ocean Modelling*, **88**, 1–4.
- Lüpkes, C. and V. M. Gryanik, 2015: A stability-dependent parametrization of transfer coefficients formomentum and heat over polar sea ice to be used in climate models. *Journal of Geophysical Research*, **120** (2), 552–581, [doi](#).
- Lüpkes, C., V. M. Gryanik, J. Hartmann, and E. L. Andreas, 2012: A parametrization, based on sea ice morphology, of the neutral atmospheric drag coefficients for weather prediction and climate models. *Journal of Geophysical Research Atmospheres*, **117** (13), 1–18, [doi](#).
- Madec, G., M. Chartier, and M. Crépon, 1991a: The effect of thermohaline forcing variability on deep water formation in the western mediterranean sea: a high-resolution three-dimensional numerical study. *Dynamics of Atmospheres and Oceans*, **15** (3-5), 301–332, [doi](#).

- Madec, G. and M. Crépon, 1991: Thermohaline-driven deep water formation in the northwestern mediterranean sea. *Deep convection and deep water formation in the oceans, Proceedings of the international Monterey colloquium on deep convection and deep water formation in the oceans*, Chu, P. and J. Gascard, Eds., Elsevier, Elsevier Oceanography, Vol. 57, 241–265, [doi](#).
- Madec, G. and P. Delécluse, 1997: The OPA/ARPEGE and OPA/LMD global ocean-atmosphere coupled model. *International WOCE Newsletter*, **26**, 12–15, [URL](#).
- Madec, G., P. Delécluse, M. Crépon, and M. Chartier, 1991b: A three-dimensional numerical study of deep-water formation in the northwestern mediterranean sea. *Journal of Physical Oceanography*, **21** (9), 1349–1371, [doi](#).
- Madec, G., P. Delécluse, M. Crépon, and F. Lott, 1996: Large-scale preconditioning of deep-water formation in the northwestern mediterranean sea. *Journal of Physical Oceanography*, **26** (8), 1393–1408, [doi](#).
- Madec, G., P. Delécluse, M. Imbard, and C. Lévy, 1998: *OPA 8.1 Ocean General Circulation Model reference manual*, Vol. 11. Institut Pierre Simon Laplace (IPSL), 91 pp., [URL](#).
- Madec, G. and M. Imbard, 1996: A global ocean mesh to overcome the north pole singularity. *Climate Dynamics*, **12** (6), 381–388, [doi](#).
- Maltrud, M. E., R. D. Smith, A. J. Semtner, and R. C. Malone, 1998: Global eddy-resolving ocean simulations driven by 1985–1995 atmospheric winds. *Journal of Geophysical Research*, **103** (C13), 30 825–30 853, [doi](#).
- Marchesiello, P., J. C. McWilliams, and A. Shchepetkin, 2001: Open boundary conditions for long-term integration of regional oceanic models. *Ocean Modelling*, **3** (1-2), 1–20, [doi](#).
- Marsaleix, P., F. Auclair, J. W. Floor, M. J. Herrmann, C. Estournel, I. Pairaud, and C. Ulses, 2008: Energy conservation issues in sigma-coordinate free-surface ocean models. *Ocean Modelling*, **20** (1), 61–89, [doi](#).
- Marsh, R., V. O. Ivchenko, N. Skliris, S. Alderson, G. R. Bigg, G. Madec, A. T. Blaker, Y. Aksenov, B. Sinha, A. C. Coward, J. Le Sommer, N. Merino, and V. B. Zalesny, 2015: NEMO-ICB (v1.0): interactive icebergs in the NEMO ocean model globally configured at eddy-permitting resolution. *Geoscientific Model Development*, **8** (5), 1547–1562, [doi](#).
- Marti, O., G. Madec, and P. Delécluse, 1992: Comment on “Net diffusivity in ocean general circulation models with nonuniform grids” by F. L. yin and I. Y. fung. *Journal of Geophysical Research*, **97** (C8), 12 763–12 766, [doi](#).
- Martin, T. and A. Adcroft, 2010: Parameterizing the fresh-water flux from land ice to ocean with interactive icebergs in a coupled climate model. *Ocean Modelling*, **34** (3-4), 111–124, [doi](#).
- Mathiot, P., A. Jenkins, C. Harris, and G. Madec, 2017: Explicit representation and parametrised impacts of under ice shelf seas in the z^* coordinate ocean model NEMO 3.6. *Geoscientific Model Development*, **10** (7), 2849–2874, [doi](#).
- McDougall, T. J., 1987: Neutral surfaces. *Journal of Physical Oceanography*, **17** (11), 1950–1964, [doi](#).
- McDougall, T. J. and J. R. Taylor, 1984: Flux measurements across a finger interface at low values of the stability ratio. *Journal of Marine Research*, **42** (1), 1–14, [doi](#).
- McWilliams, J. C., P. P. Sullivan, and C.-H. Moeng, 1997: Langmuir turbulence in the ocean. *Journal of Fluid Mechanics*, **334**, 1–30, [doi](#).
- Mellor, G. and A. Blumberg, 2004: Wave breaking and ocean surface layer thermal response. *Journal of Physical Oceanography*, **34** (3), 693–698, [doi](#).
- Mellor, G. L. and T. Yamada, 1982: Development of a turbulence closure model for geophysical fluid problems. *Reviews of Geophysics*, **20** (4), 851–875, [doi](#).
- Merryfield, W. J., G. Holloway, and A. E. Gargett, 1999: A global ocean model with double-diffusive mixing. *Journal of Physical Oceanography*, **29** (6), 1124–1142, [doi](#).
- Mesinger, F. and A. Arakawa, 1976: GARP Publication, Vol. 1. [URL](#).
- Morel, A., 1988: Optical modeling of the upper ocean in relation to its biogenous matter content (case I waters). *Journal of Geophysical Research*, **93** (C9), 10 749–10 768, [doi](#).
- Morel, A. and J.-F. Berthon, 1989: Surface pigments, algal biomass profiles, and potential production of the euphotic layer: Relationships reinvestigated in view of remote-sensing applications. *Limnology and Oceanography*, **34** (8), 1545–1562, [doi](#).
- Morel, A. and S. Maritorena, 2001: Bio-optical properties of oceanic waters: A reappraisal. *Journal of Geophysical Research*, **106** (C4), 7163–7180, [doi](#).
- Murray, R. J., 1996: Explicit generation of orthogonal grids for ocean models. *Journal of Computational Physics*, **126** (2), 251–273, [doi](#).
- Oey, L.-Y., 2006: An OGCM with movable land-sea boundaries. *Ocean Modelling*, **13** (2), 176–195, [doi](#).
- Osborn, T. R., 1980: Estimates of the local rate of vertical diffusion from dissipation measurements. *Journal of Physical Oceanography*, **10** (1), 83–89, [doi](#).
- Pacanowski, R. C. and A. Gnanadesikan, 1998: Transient response in a z-level ocean model that resolves topography with partial-cells. *Monthly Weather Review*, **126** (12), 3248–3270, [doi](#).
- Pacanowski, R. C. and S. G. H. Philander, 1981: Parameterization of vertical mixing in numerical models of tropical oceans. *Journal of Physical Oceanography*, **11** (11), 1443–1451, [doi](#).
- Paulson, C. A. and J. J. Simpson, 1977: Irradiance measurements in the upper ocean. *Journal of Physical Oceanography*, **7** (6), 952–956, [doi](#).
- Penduff, T., J. Le Sommer, B. Barnier, A.-M. Tréguier, J.-M. Molines, and G. Madec, 2007: Influence of numerical schemes on current-topography interactions in 1/4° global ocean simulations. *Ocean Science*, **3** (4), 509–524, [doi](#).
- Qiao, F., Y. Yuan, T. Ezer, C. Xia, Y. Yang, X. Lu, and Z. Song, 2010: A three-dimensional surface waveocean circulation coupled model and its initial testing. *Ocean Dynamics*, **60** (5), 1339–1335, [doi](#).
- Redi, M. H., 1982: Oceanic isopycnal mixing by coordinate rotation. *Journal of Physical Oceanography*, **12** (10), 1154–1158, [doi](#).
- Reffray, G., R. Bourdalle-Badie, and C. Calone, 2015: Modelling turbulent vertical mixing sensitivity using a 1-d version of nemo. *Geoscientific Model Development*, **8** (1), 69–86, [doi](#).

- Richtmyer, R. D. and K. W. Morton, 1967: 2d ed., Interscience tracts in pure and applied mathematics, Interscience, 405 pp., [URL](#).
- Robert, A. J., 1966: The integration of a low order spectral form of the primitive meteorological equations. *Journal of the Meteorological Society of Japan*, **44** (5), 237–245, [doi](#).
- Rodgers, K. B., O. Aumont, S. E. M. Fletcher, Y. Plancherel, L. Bopp, C. de Boyer Montégut, D. Iudicone, R. F. Keeling, G. Madec, and R. Wanninkhof, 2014: Strong sensitivity of southern ocean carbon uptake and nutrient cycling to wind stirring. *Biogeosciences*, **11** (15), 4077–4098, [doi](#).
- Rodi, W., 1987: Examples of calculation methods for flow and mixing in stratified fluids. *Journal of Geophysical Research*, **92** (C5), 5305–5328, [doi](#).
- Roquet, F., G. Madec, L. Brodeau, and J. Nycander, 2015a: Defining a simplified yet “realistic” equation of state for seawater. *Journal of Physical Oceanography*, **45** (10), 2564–2579, [doi](#).
- Roquet, F., G. Madec, T. J. McDougall, and P. M. Barker, 2015b: Accurate polynomial expressions for the density and specific volume of seawater using the TEOS-10 standard. *Ocean Modelling*, **90**, 29–43, [doi](#).
- Roullet, G. and G. Madec, 2000: Salt conservation, free surface, and varying levels: A new formulation for ocean general circulation models. *Journal of Geophysical Research*, **105** (C10), 23 927–23 942, [doi](#).
- Sarmiento, J. L. and K. Bryan, 1982: An ocean transport model for the north atlantic. *Journal of Geophysical Research*, **87** (C1), 394–409, [doi](#).
- Shchepetkin, A. F., 2015: An adaptive, courant-number-dependent implicit scheme for vertical advection in oceanic modeling. *Ocean Modelling*, **91**, 38–69, [doi](#).
- Shchepetkin, A. F. and J. C. McWilliams, 2005: The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Modelling*, **9** (4), 347–404, [doi](#).
- , 2009: *Computational kernel algorithms for fine-scale, multiprocess, longtime oceanic simulations*, Handbook of Numerical Analysis, Vol. 14, chap. 2, 121–183. Elsevier, [doi](#).
- Siddorn, J. R. and R. Furner, 2013: An analytical stretching function that combines the best attributes of geopotential and terrain-following vertical coordinates. *Ocean Modelling*, **66**, 1–13, [doi](#).
- Smagorinsky, J., 1963: General circulation experiments with the primitive equations: I. the basic experiment. *Monthly Weather Review*, **91** (3), 99–164, [doi](#).
- Song, Y. and D. Haidvogel, 1994: A semi-implicit ocean circulation model using a generalized topography-following coordinate system. *Journal of Computational Physics*, **115** (1), 228–244, [doi](#).
- Song, Y. T., 1998: A general pressure gradient formulation for ocean models. part I: Scheme design and diagnostic analysis. *Monthly Weather Review*, **126** (12), 3213–3230, [doi](#).
- St Laurent, L. C., H. L. Simmons, and S. R. Jayne, 2002: Estimating tidally driven mixing in the deep ocean. *Geophysical Research Letters*, **29** (23), 2101–2104, [doi](#).
- Stacey, M. W., 1999: Simulations of the wind-forced near-surface circulation in knight inlet: A parameterization of the roughness length. *Journal of Physical Oceanography*, **29** (6), 1363–1367, [doi](#).
- Steele, M., R. Morley, and W. Ermold, 2001: PHC: A global ocean hydrography with a high-quality arctic ocean. *Journal of Climate*, **14** (9), 2079–2087, [doi](#).
- Stein, C. A. and S. Stein, 1992: A model for the global variation in oceanic depth and heat flow with lithospheric age. *Nature*, **359** (6391), 123–129, [doi](#).
- Stokes, G. G., 2009: *On the theory of oscillatory waves*, Cambridge Library Collection - Mathematics, Vol. 1, chap. 12, 197–229. Cambridge University Press, [doi](#).
- Talagrand, O., 1972: On the damping of high-frequency motions in four-dimensional assimilation of meteorological data. *Journal of the Atmospheric Sciences*, **29** (8), 1571–1574, [doi](#).
- Tréguier, A. M., 1992: Kinetic energy analysis of an eddy resolving, primitive equation model of the north atlantic. *Journal of Geophysical Research*, **97** (C1), 687–701, [doi](#).
- Tréguier, A.-M., J. K. Dukowicz, and K. Bryan, 1996: Properties of nonuniform grids used in ocean general circulation models. *Journal of Geophysical Research*, **101** (C9), 20 877–20 881, [doi](#).
- Tréguier, A. M., I. M. Held, and V. D. Larichev, 1997: Parameterization of quasigeostrophic eddies in primitive equation ocean models. *Journal of Physical Oceanography*, **27** (4), 567–580, [doi](#).
- Umlauf, L. and H. Burchard, 2003: A generic length-scale equation for geophysical turbulence models. *Journal of Marine Research*, **61** (2), 235–265, [doi](#).
- , 2005: Second-order turbulence closure models for geophysical boundary layers. A review of recent work. *Continental Shelf Research*, **25** (7-8), 795–827, [doi](#).
- Vallis, G. K., 2006: Cambridge University Press, [doi](#).
- Warner, J. C., Z. Defne, K. Haas, and H. G. Arango, 2013: A wetting and drying scheme for ROMS. *Computers & Geosciences*, **58**, 54–61, [doi](#).
- Warner, J. C., C. R. Sherwood, H. G. Arango, and R. P. Signell, 2005: Performance of four turbulence closure models implemented using a generic length scale method. *Ocean Modelling*, **8** (1-2), 81–113, [doi](#).
- Weatherly, G. L., 1984: An estimate of bottom frictional dissipation by gulf stream fluctuations. *Journal of Marine Research*, **42** (2), 289–301, [doi](#).
- Weaver, A. J. and M. Eby, 1997: On the numerical implementation of advection schemes for use in conjunction with various mixing parameterizations in the GFDL ocean model. *Journal of Physical Oceanography*, **27** (2), 369–377, [doi](#).
- Webb, D. J., B. A. de Cuevas, and C. S. Richmond, 1998: Improved advection schemes for ocean models. *Journal of Atmospheric and Oceanic Technology*, **15** (5), 1171–1187, [doi](#).
- White, A. A., B. J. Hoskins, I. Roulstone, and A. Staniforth, 2005: Consistent approximate models of the global atmosphere: shallow, deep, hydrostatic, quasi-hydrostatic and non-hydrostatic. *Quarterly Journal of the Royal Meteorological Society*, **131**, 2081–2107, [doi](#).

- White, L., A. Adcroft, and R. Hallberg, 2009: High-order regridding-remapping schemes for continuous isopycnal and generalized coordinates in ocean models. *Journal of Computational Physics*, **228** (23), 8665–8692, [doi](#).
- Wilcox, D. C., 1988: Reassessment of the scale-determining equation for advanced turbulence models. *AIAA Journal*, **26** (11), 1299–1310, [doi](#).
- Willebrand, J., B. Barnier, C. Böning, C. Dieterich, P. D. Killworth, C. L. Provost, Y. Jia, J.-M. Molines, and A. L. New, 2001: Circulation characteristics in three eddy-permitting models of the north atlantic. *Progress in Oceanography*, **48** (2-3), 123–161, [doi](#).
- Zalesak, S. T., 1979: Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, **31** (3), 335–362, [doi](#).
- Zhang, R.-H. and M. Endoh, 1992: A free surface general circulation model for the tropical pacific ocean. *Journal of Geophysical Research*, **97** (C7), 11 237–11 255, [doi](#).

Namelist blocks

Symbols

&nam_asminc..... 179, 180
&nam_diadct 154
&nam_diaharm..... 153
&nam_tide..... 77, 103

B

&nambbc..... 39
&nambb1..... 39
&nambdy..... 97
&nambdy_dta..... 97, 100
&nambdy_index..... 98
&nambdy_tide..... 103
&namberg..... 83

C

&namcfg. 24, 27, 93, 187, 192, 195,
243
&namctl..... 158, 189

D

&namdom.. 26, 27, 43, 65, 241, 242,
245
&namdrg..... 127, 130
&namdyn_adv..... 49, 52
&namdyn_hpg..... 54
&namdyn_ldf..... 57, 106
&namdyn_spg..... 56
&namdyn_vor..... 49

E

&nameos..... 44

F

&namflo..... 152

L

&namlbc..... 92

M

&nammpp..... 96, 188

N

&namnc4..... 149, 150

O

&namobs..... 160, 161, 172

P

&namptr..... 157

R

&namrun..... 18, 149

S

&namsao..... 172
&namsbc..... 67, 77, 79, 85–88

&namsbc_apr..... 68, 77
&namsbc_blk..... 75
&namsbc_cpl..... 77, 85
&namsbc_flx..... 73
&namsbc_rnf..... 79
&namsbc_sas..... 73
&namsbc_ssr..... 88
&namsbc_wave..... 85
&namsto..... 183

T

&namtra_adv..... 31
&namtra_dmp..... 42
&namtra_eiv..... 112
&namtra_ldf..... 34, 106, 227
&namtra_mle..... 112
&namtra_qsr..... 38, 68
&namtrd..... 150, 151
&namtsd..... 28, 43

U

&namusr_def..... 195

Z

&namzdf 36, 60, 114, 116, 125, 126
&namzdf_gls..... 120
&namzdf_iwm..... 131
&namzdf_ric..... 115
&namzdf_tke..... 118
&namzgr_sco..... 246

A

key_agrif..... 195
 key_asminc..... 179

C

key_cld..... 192
 key_cice..... 89

D

key_diadct..... 153
 key_diaharm..... 153
 key_diahth..... 157

F

key_floats..... 151

I

key_iomput..... 137, 150–152

M

key_mpi2..... 96

key_mpp_mpi..... 24, 85, 96, 137, 188

N

key_nemocice_decomp..... 90
 key_netcdf4..... 149, 150
 key_nosignedzero..... 188

O

key_oa3mct_v1v2..... 76
 key_oasis3..... 76

S

key_si3..... 89

T

key_top..... 77
 key_trdmx1_trc..... 151
 key_trdtrc..... 151

V

key_vectopt_loop..... 189

<p>C</p> <p><i>closea.F90</i> 185, 186, 188</p>	<p>L</p> <p><i>lbclnk.F90</i> 93, 95, 96</p> <p><i>lbcnfd.F90</i> 93</p> <p><i>ldfc1d_c2d.F90</i> 110</p> <p><i>ldfdyn.F90</i> 109</p> <p><i>ldfslp.F90</i> 34, 106</p> <p><i>ldftra.F90</i> 109</p> <p><i>lib_fortran.F90</i> 188</p> <p><i>lib_mpp.F90</i> 95, 96</p>	<p>T</p> <p><i>traadv.F90</i> 30</p> <p><i>traadv_cen.F90</i> 32</p> <p><i>traadv_fct.F90</i> 32</p> <p><i>traadv_mus.F90</i> 33</p> <p><i>traadv_qck.F90</i> 34</p> <p><i>traadv_ubs.F90</i> 33</p> <p><i>trabbc.F90</i> 39</p> <p><i>trabbl.F90</i> 39</p> <p><i>tradmp.F90</i> 42</p> <p><i>traldf.F90</i> 34, 45</p> <p><i>traldf_iso.F90</i> 35, 36</p> <p><i>traldf_lap_blp.F90</i> 35</p> <p><i>traldf_triad.F90</i> 35</p> <p><i>tranpc.F90</i> 30</p> <p><i>tranxt.F90</i> 43</p> <p><i>traqsr.F90</i> 38, 39, 69</p> <p><i>trasbc.F90</i> 37, 68</p> <p><i>trazdf.F90</i> 15, 36, 43, 114, 133</p> <p><i>trc_oce.F90</i> 39</p> <p><i>trddyn.F90</i> 151</p> <p><i>trdtra.F90</i> 151</p>
<p>D</p> <p><i>daymod.F90</i> 73</p> <p><i>diaar5.F90</i> 156, 157</p> <p><i>diahth.F90</i> 157</p> <p><i>diaptr.F90</i> 157</p> <p><i>diawri.F90</i> 73</p> <p><i>divcur.F90</i> 48</p> <p><i>divhor.F90</i> 79, 81</p> <p><i>domain.F90</i> 96</p> <p><i>domhgr.F90</i> 25, 241</p> <p><i>domzgr.F90</i> 25, 242</p> <p><i>dtastd.F90</i> 97</p> <p><i>dtatsd.F90</i> 28</p> <p><i>dynadv.F90</i> 53</p> <p><i>dynadv_ubs.F90</i> 53</p> <p><i>dyncor_c1d.F90</i> 192</p> <p><i>dynhpg.F90</i> 45, 54, 63</p> <p><i>dynkeg.F90</i> 52</p> <p><i>dynldf.F90</i> 48, 57</p> <p><i>dynnxt.F90</i> 56, 65</p> <p><i>dynsp_ft.F90</i> 65</p> <p><i>dynspg.F90</i> 56, 77</p> <p><i>dynspg_exp.F90</i> 56</p> <p><i>dynspg_ft.F90</i> 56</p> <p><i>dynvor.F90</i> 49, 50, 52</p> <p><i>dynzad.F90</i> 52</p> <p><i>dynzdf.F90</i> 15, 48, 60, 68, 114, 128–131, 133</p>	<p>N</p> <p><i>nemo.F90</i> 172</p> <p><i>nemogcm.F90</i> 73, 160, 172</p>	<p>U</p> <p><i>userdef_istate.F90</i> 28</p> <p><i>usrdef_fmask.F90</i> 185</p> <p><i>usrdef_hgr.F90</i> .. 24, 25, 27, 185, 241</p> <p><i>usrdef_zgr.F90</i> 24, 27, 65</p>
<p>E</p> <p><i>eosbn2.F90</i> 30, 43, 45, 127</p>	<p>P</p> <p><i>par_oce.F90</i> 96</p> <p><i>phycst.F90</i> 30, 44</p>	<p>W</p> <p><i>wet_dry.F90</i> 62</p>
<p>F</p> <p><i>fldread.F90</i> 43, 69</p>	<p>S</p> <p><i>sbcapr.F90</i> 77</p> <p><i>sbcblk.F90</i> 73</p> <p><i>sbcblk_algo_coare.F90</i> 75</p> <p><i>sbcblk_algo_coare3p5.F90</i> 75</p> <p><i>sbcblk_algo_ecmwf.F90</i> 75</p> <p><i>sbcblk_algo_ncar.F90</i> 75</p> <p><i>sbcctl.F90</i> 76</p> <p><i>sbcdcy.F90</i> 87</p> <p><i>sbcftx.F90</i> 73</p> <p><i>sbcfwb.F90</i> 90</p> <p><i>sbcice_cice.F90</i> 89</p> <p><i>sbcice_if.F90</i> 89</p> <p><i>sbcisf.F90</i> 79</p> <p><i>sbcmod.F90</i> 37, 69, 73</p> <p><i>sbcnrf.F90</i> 78</p> <p><i>sbcjas.F90</i> 73</p> <p><i>sbcjssr.F90</i> 77, 88</p> <p><i>sbc tide.F90</i> 77</p> <p><i>sbcwave.F90</i> 85</p> <p><i>sshwzv.F90</i> 48</p> <p><i>step.F90</i> 73, 160</p> <p><i>step_c1d.F90</i> 192</p> <p><i>stopar.F90</i> 183</p> <p><i>stopts.F90</i> 183</p> <p><i>storg.F90</i> 183</p> <p><i>stpctl.F90</i> 73</p>	<p>Z</p> <p><i>zdfddm.F90</i> 126</p> <p><i>zdfdr.F90</i> 127–130</p> <p><i>zdfgls.F90</i> 114</p> <p><i>zdfmxi.F90</i> 121</p> <p><i>zdfosm.F90</i> 114</p> <p><i>zdfphy.F90</i> 114</p> <p><i>zdftrc.F90</i> 114</p> <p><i>zdfike.F90</i> 114</p> <p><i>zpsdhe.F90</i> 54</p> <p><i>zpsdhe.F90</i> 35, 45, 46</p>
<p>G</p> <p><i>geo2ocean.F90</i> 88</p>	<p>I</p> <p><i>iom.F90</i> 73, 138–140, 143</p> <p><i>istate.F90</i> 28</p>	

Namelist parameters

Symbols

) 187
 121
 ; 121
 \check@icr 121

A

a 242

C

clim 70
 clname 77, 103
 cn_cfg 24
 cn_dir 69, 75
 cn_domcfg 24–26, 185
 cn_domcfg_out 27
 cn_dyn3d 101
 cn_gridsearch 160
 cn_mask_file 102
 cn_profbbfiles 160
 cn_resto 43
 cn_storst_in 183
 cn_storst_out 183
 cn_tra 98, 101
 constant value 75

F

filtide 103, 104

I

iom_get 187

J

jperio 245
 jphgr_mesh 241, 242
 jpnfl 152
 jpnflnewflo 152
 jpni 96, 97, 188
 jpnj 96, 97

L

ldbletanh 243
 ln_apr_dyn 61, 68, 77
 ln_apr_obc 77

ln_ariane 152
 ln_asmdin 179
 ln_asmiau 179
 ln_bdy 98, 196
 ln_bdytide_2ddta 103
 ln_bdytide_conj 104
 ln_bench 195
 ln_blk 67, 87
 ln_boost 129
 ln_botmix_triad. 36, 227, 233,
 237

ln_bt_av 57, 58
 ln_bt_fw 57, 58
 ln_bt_nn_auto 57
 ln_Cd_L12 73, 75
 ln_Cd_L15 73, 76
 ln_cdgw 68, 86
 ln_cfmeta 149
 ln_closea 186
 ln_COARE_3p0 73, 75
 ln_COARE_3p5 73, 75
 ln_convmix 121
 ln_coords_file 98
 ln_cpl 67, 85
 ln_ctl 85, 189
 ln_dia_osm 121
 ln_diaobs 160
 ln_diaptr 157
 ln_dm2dc 68, 87
 ln_drgimp 129, 130
 ln_dyn3d_dmp 101
 ln_dyn_mxl 151
 ln_dyn_trd 151
 ln_dynadv_cen2 53
 ln_dynadv_ubs 53, 106
 ln_dynhpg_djc 55
 ln_dynhpg_imp 55
 ln_dynhpg_isf 55
 ln_dynhpg_prj 55
 ln_dynhpg_sco 55
 ln_dynhpg_vec 65
 ln_dynhpg_zco 54
 ln_dynhpg_zps 54
 ln_dynldf_bilap 53, 60
 ln_dynldf_blp 106
 ln_dynldf_hor 59, 106
 ln_dynldf_iso 59
 ln_dynldf_lap 53, 59, 106

ln_dynldf_OFF 106
 ln_dynspg_exp 16, 56
 ln_dynspg_ts 16, 56
 ln_dynvor_con 50
 ln_dynvor_eeen 50, 213, 219
 ln_dynvor_ene 50, 213
 ln_dynvor_ens 50, 218
 ln_dynvor_mix 50
 ln_dynzad_zts 52
 ln_ECMWF 73, 75
 ln_eos80 44
 ln_flo_ascii 152
 ln_flork4 152
 ln_flx 67, 73, 87, 88
 ln_full_vel 100
 ln_glo_trd 151
 ln_grid_global 160
 ln_grid_search_lookup.. 160
 ln_hsb 83
 ln_ice_embd 68
 ln_icebergs 83
 ln_iscpl 82
 ln_isf 68
 ln_isfcav 26, 45, 55, 79, 81, 129,
 243, 245
 ln_KE_trd 151
 ln_kpprimix 121
 ln_lc 118
 ln_ldfeiv 111
 ln_length_lim 120
 ln_lin 128
 ln_linssh 26, 30, 31, 37, 151, 156,
 248
 ln_list 97
 ln_loglayer 129
 ln_meshmask 27
 ln_mevar 131
 ln_mixcpl 67, 73
 ln_mldw 115
 ln_mle 112
 ln_mskland 97
 ln_mus_ups 33
 ln_mxl0 118
 ln_nc4zip 149
 ln_NCAR 73, 75
 ln_nnogather 188
 ln_non_lin 129
 ln_OFF 128

ln_qsr_2bd..... 38
 ln_qsr_bio..... 39
 ln_qsr_rgb..... 39
 ln_read_cfg..... 24, 195
 ln_read_load..... 78
 ln_ref_apr..... 77
 ln_rnf..... 68
 ln_rnf_depth..... 79
 ln_rnf_sal..... 79
 ln_rnf_temp..... 79
 ln_rstflo..... 152
 ln_rstseed..... 183
 ln_rststo..... 183
 ln_s3d..... 160
 ln_s_SF12..... 246, 247
 ln_s_SH94..... 246
 ln_scal_load..... 78
 ln_sco..... 26, 35, 106, 246
 ln_sdw..... 68, 86
 ln_seos..... 43, 44, 107
 ln_sigcrit..... 248
 ln_ssr..... 68
 ln_sstnight..... 160
 ln_stcor..... 68, 87
 ln_sto_eos..... 183
 ln_subbas..... 157
 ln_t3d..... 160
 ln_taudif..... 75
 ln_tauw..... 68, 87
 ln_tauwoc..... 68, 87
 ln_teos10..... 43, 44
 ln_teos80..... 43
 ln_tide..... 61, 77, 103
 ln_tide_pot..... 61, 77
 ln_tide_ramp..... 78
 ln_tra_dmp..... 101
 ln_tra_mxl..... 30, 151
 ln_tra_trd..... 30, 151
 ln_traadv_cen..... 32
 ln_traadv_fct..... 32
 ln_traadv_mus..... 33
 ln_traadv_OFF..... 30
 ln_traadv_qck..... 34
 ln_traadv_tvd_zts..... 52
 ln_traadv_ubs..... 33
 ln_trabbc..... 39, 114
 ln_trabbl..... 39, 41, 233
 ln_tradmp..... 42
 ln_traldf_blp..... 34, 35, 106
 ln_traldf_eiv..... 227
 ln_traldf_gdia..... 238
 ln_traldf_hor..... 35, 36, 106
 ln_traldf_iso..... 35, 36, 227
 ln_traldf_lap..... 34–36, 106
 ln_traldf_lev..... 35
 ln_traldf_msc..... 34, 36, 227
 ln_traldf_OFF..... 34, 35, 106
 ln_traldf_triad..... 35, 36,
 106–108, 111, 112, 227,
 236
 ln_tranpc..... 125
 ln_traqsr..... 38, 68, 69
 ln_triad_iso..... 36, 227, 234, 235,
 238
 ln_tsd_dmp..... 42
 ln_tsd_init..... 28, 42

ln_tsdiff..... 131
 ln_use_jattr..... 187
 ln_usr..... 68
 ln_vol..... 102
 ln_vor_trd..... 151
 ln_wave..... 68, 85
 ln_wd_dl_bc..... 62
 ln_wd_dl_ramp..... 62
 ln_write_cfg..... 27
 ln_xios_read..... 138
 ln_zad_Aimp..... 132
 ln_zco..... 26, 35, 36, 245
 ln_zdfcst..... 114
 ln_zdfddm..... 37, 126
 ln_zdfevd..... 125, 126
 ln_zdfexp..... 60
 ln_zdfgls..... 119, 120, 126, 196
 ln_zdfiwm..... 131
 ln_zdfnpc..... 125, 126
 ln_zdfosm..... 120, 126
 ln_zdftric..... 114, 115
 ln_zdfswm..... 132
 ln_zdftke..... 115, 126
 ln_zps..... 26, 35, 36, 45, 46, 245

N

nb_bdy..... 98
 niaufn..... 179
 nit000..... 153
 nit000_han..... 153
 nitend..... 153
 nitend_han..... 153
 nn_2dint..... 166, 167
 nn_aei_ijk_t..... 112
 nn_ahm_ijk_t..... 109–111
 nn_aht_ijk_t..... 109–111
 nn_atfp..... 65
 nn_ave..... 121
 nn_baro..... 57, 58
 nn_bathy..... 245
 nn_bbl_adv..... 41, 42
 nn_bbl_ldf..... 41, 233
 nn_bc_bot..... 119
 nn_bc_surf..... 119
 nn_bdy_jpk..... 100
 nn_btflt..... 58
 nn_cen_h..... 32
 nn_cen_v..... 32
 nn_cfg..... 24
 nn_chldta..... 39
 nn_clo..... 119, 120
 nn_clos..... 120
 nn_dct..... 154
 nn_dctwri..... 154
 nn_debug..... 154
 nn_diacfl..... 158
 nn_divdmp..... 180
 nn_drown..... 83
 nn_dtrc..... 144
 nn_dyn2d_dta..... 100
 nn_dynhpg_rst..... 56
 nn_ediff..... 116
 nn_ediss..... 116
 nn_eeen_e3f..... 51
 nn_eosflt..... 183
 nn_eosord..... 183

nn_etau..... 118, 119
 nn_euler..... 18
 nn_evdm..... 126
 nn_fct_h..... 32
 nn_fct_v..... 32
 nn_fsbc..... 68–70, 73, 90, 140, 144
 nn_fwb..... 68, 90
 nn_gammablck..... 81
 nn_geoflx..... 39
 nn_GYRE..... 195
 nn_ice..... 68, 88
 nn_ice_embd..... 61
 nn_isf..... 79, 81, 82
 nn_isfblk..... 79
 nn_lsm..... 68
 nn_msh..... 97
 nn_mxl..... 116, 117
 nn_nit000..... 83
 nn_npc..... 125
 nn_osm_wave..... 120, 121
 nn_pdl..... 116
 nn_profdavtypes..... 160
 nn_ric..... 115
 nn_rimwidth..... 100–102
 nn_sample_rate..... 85
 nn_sdrift..... 68, 86
 nn_stab_func..... 119
 nn_sto_eos..... 183
 nn_stockfl..... 152
 nn_test_box..... 83
 nn_test_icebergs..... 83
 nn_tra_dta..... 98, 100
 nn_trd..... 151
 nn_ubs_v..... 33
 nn_verbose_level..... 85
 nn_verbose_write..... 85
 nn_volctl..... 103
 nn_writefl..... 152
 nn_wxios..... 138
 nn_zdfexp..... 60
 nn_zdmp..... 43
 nn_zpyc..... 131
 nstep_han..... 153
 nz_rnf..... 79

O

open_ocean_jstart..... 187

P

ppa0..... 243
 ppa1..... 243
 ppacr..... 243
 ppdzmin..... 243
 ppe1_deg..... 241
 ppe1_m..... 242
 ppe2_deg..... 241
 ppe2_m..... 242
 pp glam0..... 242
 ppgphi0..... 242
 pphmax..... 243, 245
 ppkth..... 243
 ppsur..... 243

R

FORTRAN subroutines

<p style="text-align: center;">D</p> <p>dom_glo..... 96</p> <p style="text-align: center;">F</p> <p>fld_read..... 69, 71</p> <p style="text-align: center;">H</p> <p>hgr_read..... 241</p> <p style="text-align: center;">I</p> <p>iom_put..... 137, 150</p> <p>iom_rstput..... 138</p> <p>iom_set_rst_vars,..... 138</p> <p>iom_set_rstw_active..... 138</p> <p>iom_set_rstw_core..... 138</p> <p style="text-align: center;">L</p> <p>lbc_lnk..... 96</p> <p>ldf_eiv_trp..... 112</p> <p>ldf_mle_trp..... 112</p> <p>ldf_slp_init..... 106</p> <p style="text-align: center;">R</p>	<p>rst_write..... 138</p> <p style="text-align: center;">S</p> <p>sbc_blk_init..... 69</p> <p>sbc_isf_div..... 81</p> <p>sbc_rnf_div..... 79</p> <p>sbcbk_algo_ncar..... 86</p> <p>sto_par..... 183</p> <p>sto_par_init..... 183</p> <p>sto_rst_write..... 183</p> <p>stp..... 192</p> <p>stp_cld..... 192</p> <p>stp_ctl..... 73</p> <p style="text-align: center;">T</p> <p>tra_adv..... 112</p> <p>tra_dmp_init..... 43</p> <p>tra_ldf_blp..... 35</p> <p>tra_ldf_lap..... 35</p> <p>trc_oce_rgb..... 39</p> <p style="text-align: center;">U</p> <p>usr_def_istate..... 28</p>
---	--