



# NEMO ocean engine

Version 5.0 - December , 2024

DOI [10.5281/zenodo.1464186](https://doi.org/10.5281/zenodo.1464186)

Authors of the version-5.0 release:

Gurvan Madec

Mike Bell

Adam Blaker

Clément Bricaud

Diego Bruciaferri

Daley Calvert

Davi Carneiro

Jérôme Chanut

Andrew Coward

Casimir de Lavergne

Italo Epicoco

Christian Éthé

Emma Fiedler

David Ford

Jonas Ganderton

James Harle

Katherine Hutchinson

Doroteaciro Iovino

Robert King

Dan Lea

Eric Maisonnave

Julian Mak

Juan Manuel Castillo Sanchez

Matt Martin

Diana Martins

Sébastien Masson

Pierre Mathiot

Francesca Mele

Aimie Moulin

Simon Müller

George Nurser

Mathieu Peltier

Renaud Person

Clément Rousset

David Schroeder

Guillaume Samson

Sibylle Téchéné

List of historic co-authors can be found on the next page.

## Abstract

“Nucleus for European Modelling of the Ocean” as *NEMO* is a state-of-the-art modelling framework of ocean-related engines for research activities and forecasting services in oceanography and climatology, developed in a sustainable way since 2008 by a European consortium of 5 institutes (CMCC | CNRS | Mercator Océan | Met Office | NOC). It is intended to be a flexible tool for studying the physical and biogeochemical phenomena in the ocean circulation, as well as its interactions with the components of the Earth climate system, over a wide range of space and time scales.

Concerning the physics, the fundamental engine for the “blue ocean” solves the primitive equations of the ocean {thermo}dynamics. It can be supplemented by the “white ocean” for sea-ice {thermo}dynamics, brine inclusions and subgrid-scale thickness variations ( $SI^3$ ), and also by the “green ocean” for {on,off}line oceanic tracers transport and biogeochemical processes (*TOP-PISCES*). External alternative models can be used instead of the core engines (e.g. *BFM*). Regarding the numerics, main features include versatile data assimilation interface, agile diagnostics generation thanks to *XIOS* software, ocean-atmosphere coupling via the *OASIS* library, and seamless embedded zooms with the *AGRIF* 2-way nesting package.

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, along with various HPC functionalities to improve performances.



## 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 their 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 using the [Discourse platform](#) or by opening an issue on the [NEMO Forge](#).

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
-  the [development platform](#) of the model with the code repository for the shared reference and the last stable release (suggested download for users)
-  the [Zenodo archive](#) delivering the publications issued by the consortium and the [NEMO Open Collection](#) of reports, datasets and demonstrators.
-  The [newsletter](#) for announcements and calls from the project

## List of additional co-authors who made notable contributions to previous versions of this document

 Rachid Benshila  
 Romain Boudrallé-Badie  
 Miguel Castrillo  
 Emanuela Clementi  
 Srđan Dobricic  
 Rachel Furner  
Tim Graham  
 Claire Levy  
 Tomas Lovato  
Nicolas Martin  
 Silvia Mocavero  
 Paolo Oddo  
 Stella Paronuzzi  
Julien Paul  
 Stefanie Rynders  
Dave Storkey  
Andrea Storto  
Chris Wilson  
 Martin Vancoppenolle

## Citation

Please reference this document as follows:

Madec, G. and the NEMO System Team, 2024. NEMO Ocean Engine Reference Manual, Zenodo [doi:10.5281/zenodo.1464816](https://doi.org/10.5281/zenodo.1464816)

<b>1. Model Basics</b>	<b>3</b>
1.1. Primitive equations	4
1.1.1. Vector invariant formulation	4
1.1.2. Boundary conditions	5
1.2. Horizontal pressure gradient	6
1.2.1. Pressure formulation	6
1.2.2. Free surface formulation	6
1.3. Curvilinear $z$ -coordinate system	7
1.3.1. Tensorial formalism	7
1.3.2. Continuous model equations	8
1.4. Curvilinear generalised vertical coordinate system	9
1.4.1. $S$ -coordinate formulation	10
1.4.2. Curvilinear $z^*$ -coordinate system	11
1.4.3. Curvilinear terrain-following $s$ -coordinate	12
1.4.4. Curvilinear $\tilde{z}$ -coordinate	13
1.5. Subgrid scale physics	13
1.5.1. Vertical subgrid scale physics	14
1.5.2. Formulation of the lateral diffusive and viscous operators	14
<b>2. Time Domain</b>	<b>17</b>
2.1. Time stepping schemes: MLF and RK3	18
2.2. MLF scheme	18
2.2.1. Non-diffusive part	18
2.2.2. Diffusive part — Forward or backward scheme	19
2.2.3. Modified LeapFrog – Robert Asselin filter scheme (LF-RA)	19
2.3. RK3 scheme	20
2.3.1. Non-diffusive part	20
2.3.2. Diffusive part — Forward or backward scheme	21
2.4. Surface pressure gradient	21
2.5. Start/Restart strategy	21
2.6. Initial state ( <i>istate.F90</i> and <i>dtatsd.F90</i> )	23
2.7. Adaptive-implicit vertical advection ( <i>ln_zad_Aimp</i> )	24
2.7.1. Adaptive-implicit vertical advection in the OVERFLOW test-case	25
<b>3. Space Domain (DOM)</b>	<b>28</b>
3.1. Fundamentals of the discretisation	29
3.1.1. Arrangement of variables	29
3.1.2. Discrete operators	30
3.1.3. Numerical indexing	31
3.2. Spatial domain configuration	32
3.2.1. Horizontal grid mesh ( <i>domhgr.F90</i> )	33
3.2.2. Vertical grid ( <i>domzgr.F90</i> )	33
3.2.3. Closed seas	36
3.2.4. Output grid files	37

<b>4. Sea surface height and 2D external mode (D2D)</b>	<b>38</b>
4.1. Sea surface height ( <i>sshwzv.F90/stp2d.F90</i> ) . . . . .	39
4.1.1. Explicit free surface ( <i>ln_dynspg_exp, sshwzv.F90</i> ) . . . . .	39
4.1.2. Split-explicit free surface ( <i>ln_dynspg_ts</i> ) . . . . .	39
4.1.3. Split-explicit time-stepping ( <i>dynspg_ts.F90</i> ) . . . . .	40
4.1.4. External forcings . . . . .	41
<b>5. Ocean Dynamics (DYN)</b>	<b>43</b>
5.1. Continuity equation ( <i>sshwzv.F90, w</i> ) . . . . .	44
5.1.1. Horizontal divergence ( <i>divhor.F90, <math>\chi</math></i> ) . . . . .	44
5.1.2. Vertical velocity ( <i>sshwzv.F90</i> ) . . . . .	45
5.2. Coriolis and advection: vector invariant form . . . . .	45
5.2.1. Vorticity term ( <i>dynvor.F90</i> ) . . . . .	45
5.2.2. Kinetic energy gradient term ( <i>dynkeg.F90</i> ) . . . . .	48
5.2.3. Vertical advection term ( <i>dynzad.F90</i> ) . . . . .	48
5.3. Coriolis and advection: flux form . . . . .	48
5.3.1. Coriolis plus curvature metric terms ( <i>dynvor.F90</i> ) . . . . .	48
5.3.2. Flux form advection term ( <i>dynadv.F90</i> ) . . . . .	49
5.4. Hydrostatic pressure gradient ( <i>dynhpg.F90</i> ) . . . . .	50
5.4.1. Full step <i>Z</i> -coordinate ( <i>ln_dynhpg_zco</i> ) . . . . .	50
5.4.2. Generalised <i>S</i> -coordinates . . . . .	51
5.5. Surface pressure gradient ( <i>dynspg.F90/stp2d.F90</i> ) . . . . .	52
5.6. Lateral diffusion term and operators ( <i>dynldf.F90</i> ) . . . . .	52
5.6.1. Iso-level laplacian ( <i>ln_dynldf_lap</i> ) . . . . .	53
5.6.2. Rotated laplacian ( <i>ln_dynldf_iso</i> ) . . . . .	53
5.6.3. Iso-level bilaplacian ( <i>ln_dynldf_bilap</i> ) . . . . .	54
5.7. Vertical diffusion term ( <i>dynzdf.F90</i> ) . . . . .	54
5.8. Wetting and drying . . . . .	55
5.8.1. Directional limiter ( <i>wet_dry.F90</i> ) . . . . .	56
5.8.2. The WAD test cases ( <i>usrdef_zgr.F90</i> ) . . . . .	57
5.9. Time evolution term - leapfrog ( <i>dynatf.F90</i> ) . . . . .	57
<b>6. Ocean Tracers (TRA)</b>	<b>58</b>
6.1. Tracer advection ( <i>traadv.F90</i> ) . . . . .	59
6.1.1. CEN: Centred scheme ( <i>ln_traadv_cen</i> ) . . . . .	61
6.1.2. FCT: Flux Corrected Transport scheme ( <i>ln_traadv_fct</i> ) . . . . .	61
6.1.3. MUSCL: Monotone Upstream Scheme for Conservative Laws ( <i>ln_traadv_mus</i> ) . . . . .	62
6.1.4. UBS a.k.a. UP3: Upstream-Biased Scheme ( <i>ln_traadv_ubs</i> ) . . . . .	62
6.1.5. QCK: QuiCKest scheme ( <i>ln_traadv_qck</i> ) . . . . .	63
6.2. Tracer lateral diffusion ( <i>traldf.F90</i> ) . . . . .	63
6.2.1. Type of operator ( <i>ln_traldf_{OFF,lap,blp}</i> ) . . . . .	64
6.2.2. Action direction ( <i>ln_traldf_{lev,hor,iso,triad}</i> ) . . . . .	64
6.2.3. Iso-level (bi-)laplacian operator ( <i>ln_traldf_iso</i> ) . . . . .	65
6.2.4. Standard and triad (bi-)laplacian operator . . . . .	65
6.3. Tracer vertical diffusion ( <i>trazdf.F90</i> ) . . . . .	66
6.4. External forcing . . . . .	66
6.4.1. Surface boundary condition ( <i>trasbc.F90</i> ) . . . . .	67
6.4.2. Solar radiation penetration ( <i>traqsr.F90</i> ) . . . . .	68
6.4.3. Bottom boundary condition ( <i>trabbc.F90</i> ) - <i>ln_trabbc</i> ) . . . . .	69
6.5. Bottom boundary layer ( <i>trabbl.F90</i> - <i>ln_trabbl</i> ) . . . . .	71
6.5.1. Diffusive bottom boundary layer ( <i>nn_bbl_ldf=1</i> ) . . . . .	71
6.5.2. Advective bottom boundary layer ( <i>nn_bbl_adv=1,2</i> ) . . . . .	72
6.6. Tracer damping ( <i>tradmp.F90</i> ) . . . . .	73
6.7. Tracer time evolution ( <i>traatf.F90</i> ) . . . . .	73
6.8. Equation of state ( <i>eosbn2.F90</i> ) . . . . .	74
6.8.1. Equation of seawater ( <i>ln_{teos10,eos80,seos}</i> ) . . . . .	74
6.8.2. Brunt-Väisälä frequency . . . . .	75
6.8.3. Freezing point of seawater . . . . .	76
6.9. Wave induced transport . . . . .	76
6.10. Internal wave filtering . . . . .	76



<b>7. Surface Boundary Condition (SBC, SAS, TDE)</b>	<b>77</b>
7.1. Surface boundary condition for the ocean . . . . .	80
7.2. Input data generic interface . . . . .	81
7.2.1. Input data specification ( <i>fldread.F90</i> ) . . . . .	81
7.2.2. Interpolation on-the-fly . . . . .	82
7.2.3. Standalone surface boundary condition scheme (SAS) . . . . .	84
7.3. Flux formulation ( <i>sbcflx.F90</i> ) . . . . .	85
7.4. Bulk formulation ( <i>sbcblk.F90</i> ) . . . . .	86
7.4.1. Bulk formulae . . . . .	86
7.4.2. Bulk parametrizations . . . . .	88
7.4.3. Cool-skin and warm-layer parameterizations ( <i>ln_skin_cs</i> & <i>ln_skin_wl</i> ) . . . . .	88
7.4.4. Appropriate use of each bulk parametrization . . . . .	88
7.4.5. Ice-Atmosphere Bulk formulae . . . . .	90
7.4.6. Prescribed near-surface atmospheric state . . . . .	91
7.5. Atmospheric Boundary Layer (ABL) model ( <i>sbcabl.F90</i> ) . . . . .	91
7.5.1. ABL1D pre-processing . . . . .	91
7.5.2. ABL1D namelist . . . . .	92
7.6. Coupled formulation ( <i>sbcctl.F90</i> ) . . . . .	94
7.7. Atmospheric pressure ( <i>sbcapr.F90</i> ) . . . . .	94
7.8. Surface tides (TDE) . . . . .	96
7.8.1. Tidal constituents . . . . .	96
7.8.2. Surface tidal forcing . . . . .	96
7.9. River runoffs ( <i>sbcrrf.F90</i> ) . . . . .	97
7.10. Interactions with waves ( <i>sbcwave.F90</i> ) . . . . .	99
7.10.1. Neutral drag coefficient from wave model ( <i>ln_cdgw</i> ) . . . . .	100
7.10.2. Charnok coefficient from wave model ( <i>ln_charn</i> ) . . . . .	100
7.10.3. 3D Stokes Drift ( <i>ln_sdw</i> ) . . . . .	100
7.10.4. Stokes-Coriolis term ( <i>ln_stcor</i> ) . . . . .	101
7.10.5. Vortex-force term ( <i>ln_vortex_force</i> ) . . . . .	101
7.10.6. Wave-induced pressure term ( <i>ln_bern_srfc</i> ) . . . . .	101
7.10.7. Wave modified stress ( <i>ln_tauoc</i> & <i>ln_taw</i> ) . . . . .	102
7.10.8. Waves impact vertical mixing ( <i>ln_phioc</i> & <i>ln_stshear</i> ) . . . . .	102
7.11. Miscellaneous options . . . . .	102
7.11.1. Diurnal cycle ( <i>sbcscy.F90</i> ) . . . . .	102
7.11.2. Rotation of vector pairs onto the model grid directions . . . . .	103
7.11.3. Surface restoring to observed SST and/or SSS ( <i>sbcssr.F90</i> ) . . . . .	103
7.11.4. Handling of ice-covered area ( <i>sbcice_...</i> ) . . . . .	105
7.11.5. Freshwater budget control ( <i>sbcfwb.F90</i> ) . . . . .	105
<b>8. Land Ice Ocean interactions (ISF and ICB)</b>	<b>107</b>
8.1. Ice Shelf (ISF) . . . . .	108
8.1.1. Ocean/Ice shelf fluxes in opened cavities ( <i>isfcav.F90</i> ) . . . . .	108
8.1.2. Ocean/Ice shelf fluxes in parametrised cavities ( <i>isfpar.F90</i> ) . . . . .	112
8.1.3. Available outputs ( <i>isfdiags.F90</i> ) . . . . .	112
8.1.4. Ice sheet coupling ( <i>isfcpl.F90</i> ) . . . . .	113
8.1.5. Ice shelf load ( <i>isfload.F90</i> ) . . . . .	114
8.1.6. Under ice shelf cavity geometry ( <i>domisf.F90</i> ) . . . . .	114
8.2. Icebergs (ICB) . . . . .	115
8.2.1. Iceberg initialisation ( <i>icbclv.F90</i> ) . . . . .	115
8.2.2. Iceberg dynamics ( <i>icbdyn.F90</i> ) . . . . .	117
8.2.3. Iceberg thermodynamics ( <i>icbthm.F90</i> ) . . . . .	117
8.2.4. Iceberg mpp ( <i>icblbc.F90</i> ) . . . . .	118
8.2.5. Iceberg outputs ( <i>icbdia.F90</i> ) . . . . .	118
8.3. Land Ice Runoff . . . . .	119
<b>9. Lateral Boundary Condition (LBC)</b>	<b>121</b>
9.1. Boundary condition at the continental interface ( <i>rn_shlat</i> ) . . . . .	122
9.2. Model-domain boundary condition . . . . .	123
9.2.1. Closed, cyclic ( <i>l_Iperio,l_Jperio</i> ) . . . . .	124
9.2.2. North-fold ( <i>l_NFold = .true., c_NFtype = 'T' or c_NFtype = 'F'</i> ) . . . . .	124
9.3. Exchange with neighbouring processes ( <i>lbclnk.F90, lib_mpp.F90</i> ) . . . . .	124

9.4.	Unstructured open boundary conditions (BDY)	127
9.4.1.	Namelists	128
9.4.2.	Flow relaxation scheme	130
9.4.3.	Flather radiation scheme	131
9.4.4.	Orlanski radiation scheme	131
9.4.5.	Relaxation at the boundary	131
9.4.6.	Boundary geometry	132
9.4.7.	Input boundary data files	132
9.4.8.	Volume correction	133
9.4.9.	Tidal harmonic forcing	133
<b>10.</b>	<b>Lateral Ocean Physics (LDF)</b>	<b>135</b>
10.1.	Lateral mixing operators	136
10.2.	Direction of lateral mixing ( <i>ldfslp.F90</i> )	136
10.2.1.	Slopes for geopotential mixing in the <i>s</i> -coordinate	136
10.2.2.	Slopes for tracer iso-neutral mixing	137
10.2.3.	Slopes for momentum iso-neutral mixing	138
10.3.	Lateral mixing coefficient ( <code>nn_aht_ijk_t</code> & <code>nn_ahm_ijk_t</code> )	139
10.3.1.	Mixing coefficients read from file (=20, -30)	139
10.3.2.	Constant mixing coefficients (=0)	140
10.3.3.	Vertically varying mixing coefficients (=10)	140
10.3.4.	Mesh size dependent mixing coefficients (=20)	140
10.3.5.	Mesh size and depth dependent mixing coefficients (=30)	140
10.3.6.	Eddy parameterization ( <code>nn_aht_ijk_t=21</code> )	141
10.3.7.	Velocity dependent mixing coefficients (=31)	141
10.3.8.	Deformation rate dependent viscosities ( <code>nn_ahm_ijk_t=32</code> )	141
10.3.9.	About space and time varying mixing coefficients	142
10.4.	Eddy induced velocity ( <code>ln_ldfeiv</code> )	142
10.5.	Mixed layer eddies ( <code>ln_mle</code> )	143
<b>11.</b>	<b>Vertical Ocean Physics (ZDF)</b>	<b>145</b>
11.1.	Vertical mixing	146
11.1.1.	Background values	146
11.1.2.	Constant ( <code>ln_zdfcst</code> )	147
11.1.3.	Richardson number dependent ( <code>ln_zdfric</code> )	147
11.1.4.	TKE turbulent closure scheme ( <code>ln_zdf_tke</code> )	148
11.1.5.	GLS: Generic Length Scale ( <code>ln_zdf_gls</code> )	152
11.1.6.	OSMOSIS boundary layer scheme ( <code>ln_zdf_oscsm = .true.</code> )	154
11.1.7.	Discrete energy conservation for TKE and GLS schemes	157
11.2.	Convection	159
11.2.1.	Non-penetrative convective adjustment ( <code>ln_tranpc</code> )	159
11.2.2.	Enhanced vertical diffusion ( <code>ln_zdfevd</code> )	160
11.2.3.	Mass Flux Convection ( <code>ln_zdfmfc</code> )	160
11.2.4.	Handling convection with turbulent closure schemes ( <code>ln_zdf_{tke,gl,osm}</code> )	161
11.3.	Double diffusion mixing ( <code>ln_zdfddm</code> )	161
11.4.	Bottom and top friction ( <i>zdfdr.F90</i> )	162
11.4.1.	Free-slip boundary conditions ( <code>ln_drg_OFF</code> )	163
11.4.2.	Linear top/bottom friction ( <code>ln_lin</code> )	163
11.4.3.	Non-linear top/bottom friction ( <code>ln_non_lin</code> )	164
11.4.4.	Log-layer top/bottom friction ( <code>ln_loglayer</code> )	164
11.4.5.	Explicit top/bottom friction ( <code>ln_drgimp=.false.</code> )	164
11.4.6.	Implicit top/bottom friction ( <code>ln_drgimp=.true.</code> )	165
11.4.7.	Bottom friction with split-explicit free surface	165
11.5.	Internal wave-driven mixing ( <code>ln_zdfiwm</code> )	166
11.6.	Surface wave-induced mixing ( <code>ln_zdfswm</code> )	168
<b>12.</b>	<b>Output and Diagnostics (IOM, DIA, TRD)</b>	<b>169</b>
12.1.	Model outputs	170
12.2.	Standard model diagnostic output with XIOS ( <code>iom_put</code> , <code>key_xios</code> )	170
12.2.1.	Main XIOS configuration file ( <i>iodef.xml</i> )	171
12.2.2.	Practical issues	172

12.2.3. XML fundamentals . . . . .	173
12.2.4. Detailed functionalities . . . . .	176
12.2.5. CF metadata standard compliance . . . . .	181
12.2.6. Enabling NetCDF4 compression with XIOS . . . . .	182
12.3. Reading and writing restart files . . . . .	182
12.4. NetCDF4 support (legacy output file method) . . . . .	183
12.5. Tracer/Dynamics trends ( <code>namtrd</code> , <code>namtrc_trd</code> ) . . . . .	183
12.6. Transports across sections . . . . .	185
12.7. Diagnosing the steric effect on sea surface height . . . . .	187
12.8. Tidal harmonic and generic multiple-linear-regression analysis ( <code>diamlr.F90</code> ) . . . . .	189
12.8.1. Configuration of the multiple-linear-regression analysis . . . . .	189
12.8.2. The intermediate output and its post-processing . . . . .	190
12.9. Other diagnostics . . . . .	190
12.9.1. Depth of various quantities ( <code>diahth.F90</code> ) . . . . .	190
12.9.2. CMIP-specific and poleward transport diagnostics ( <code>diar5.F90</code> , <code>diaptr.F90</code> ) . . . . .	191
12.9.3. De-tided diagnostic output from tidal models ( <code>dia25h.F90</code> , <code>diadetide.F90</code> ) . . . . .	191
12.9.4. Courant numbers . . . . .	192
<b>13. Observation and Model Comparison (OBS)</b>	<b>193</b>
13.1. Running the observation operator code example . . . . .	194
13.2. Technical details and full namelist options . . . . .	195
13.3. Example feedback type observation file headers . . . . .	197
13.3.1. Temperature and salinity profile feedback file . . . . .	197
13.3.2. Sea level anomaly feedback file . . . . .	199
13.3.3. Sea surface temperature feedback file . . . . .	201
13.4. Adding code for a new variable . . . . .	202
13.5. Theoretical details . . . . .	203
13.5.1. Horizontal interpolation and averaging methods . . . . .	203
13.5.2. Grid search . . . . .	204
13.5.3. Parallel aspects of horizontal interpolation . . . . .	206
13.5.4. Vertical interpolation operator . . . . .	206
13.6. Standalone observation operator (SAO) . . . . .	209
13.6.1. Concept . . . . .	209
13.6.2. Using the standalone observation operator . . . . .	209
13.6.3. Configuring the standalone observation operator . . . . .	209
13.7. Observation utilities . . . . .	210
13.7.1. Obstools . . . . .	210
13.7.2. Building the obstools . . . . .	212
13.7.3. Dataplot . . . . .	212
<b>14. Apply Assimilation Increments (ASM)</b>	<b>215</b>
14.1. Direct initialization . . . . .	216
14.2. Incremental analysis updates . . . . .	216
14.3. Divergence damping initialisation . . . . .	217
14.4. Implementation details . . . . .	217
<b>15. Stochastic Parametrization of EOS (STO)</b>	<b>219</b>
15.1. Stochastic processes . . . . .	220
15.2. Implementation details . . . . .	221
<b>16. Miscellaneous Topics</b>	<b>223</b>
16.1. Representation of unresolved straits . . . . .	224
16.1.1. Hand made geometry changes . . . . .	224
16.2. Closed seas ( <code>closea.F90</code> ) . . . . .	224
16.3. Accuracy and reproducibility ( <code>lib_fortran.F90</code> ) . . . . .	226
16.3.1. Issues with intrinsic SIGN function ( <code>key_nosignedzero</code> ) . . . . .	226
16.3.2. MPP reproducibility . . . . .	227
16.4. Model optimisation, control print and benchmark . . . . .	227
16.4.1. Status and debugging information output . . . . .	227
16.4.2. Control print suboptions . . . . .	228

<b>17. Configurations</b>	<b>230</b>
17.1. Introduction	231
17.2. List of NEMO Idealised Configurations and Test Cases	231
17.3. List of NEMO Realistic Regional Configurations	232
17.4. Global configurations: the ORCA family	232
17.4.1. ORCA tripolar grid	232
17.4.2. ORCA pre-defined resolution	234
17.5. A special mention of the double gyre basin: GYRE family	234
17.6. Other well documented configurations	235
<b>A. Curvilinear <math>s</math>-Coordinate Equations</b>	<b>236</b>
A.1. Chain rule for $s$ -coordinates	237
A.2. Continuity equation in $s$ -coordinates	237
A.3. Momentum equation in $s$ -coordinate	238
A.4. Tracer equation	241
<b>B. Diffusive Operators</b>	<b>243</b>
B.1. Horizontal/Vertical $2^{nd}$ order tracer diffusive operators	244
B.2. Iso/Diapycnal $2^{nd}$ order tracer diffusive operators	245
B.3. Lateral/Vertical momentum diffusive operators	247
<b>C. Discrete Invariants of the Equations</b>	<b>248</b>
C.1. Introduction / Notations	249
C.2. Continuous conservation	250
C.3. Discrete total energy conservation: vector invariant form	252
C.3.1. Total energy conservation	252
C.3.2. Vorticity term (coriolis + vorticity part of the advection)	252
C.3.3. Pressure gradient term	255
C.4. Discrete total energy conservation: flux form	256
C.4.1. Total energy conservation	256
C.4.2. Coriolis and advection terms: flux form	256
C.5. Discrete enstrophy conservation	257
C.6. Conservation properties on tracers	259
C.6.1. Advection term	259
C.7. Conservation properties on lateral momentum physics	259
C.7.1. Conservation of potential vorticity	260
C.7.2. Dissipation of horizontal kinetic energy	260
C.7.3. Dissipation of enstrophy	261
C.7.4. Conservation of horizontal divergence	261
C.7.5. Dissipation of horizontal divergence variance	261
C.8. Conservation properties on vertical momentum physics	261
C.9. Conservation properties on tracer physics	263
C.9.1. Conservation of tracers	263
C.9.2. Dissipation of tracer variance	264
<b>D. Iso-Neutral Diffusion and Eddy Advection using Triads</b>	<b>265</b>
D.1. Choice of <code>namtra_ldf</code> namelist parameters	266
D.2. Triad formulation of iso-neutral diffusion	266
D.2.1. Iso-neutral diffusion operator	266
D.2.2. Standard discretization	267
D.2.3. Expression of the skew-flux in terms of triad slopes	267
D.2.4. Full triad fluxes	269
D.2.5. Ensuring the scheme does not increase tracer variance	269
D.2.6. Triad volumes in Griffes's scheme and in <i>NEMO</i>	270
D.2.7. Summary of the scheme	271
D.2.8. Treatment of the triads at the boundaries	272
D.2.9. Limiting of the slopes within the interior	272
D.2.10. Tapering within the surface mixed layer	272
D.3. Eddy induced advection formulated as a skew flux	275
D.3.1. Continuous skew flux formulation	275
D.3.2. Discrete skew flux formulation	276

D.3.3. Treatment of the triads at the boundaries . . . . .	277
D.3.4. Limiting of the slopes within the interior . . . . .	277
D.3.5. Tapering within the surface mixed layer . . . . .	277
D.3.6. Streamfunction diagnostics . . . . .	277
<b>E. North Pole Folding</b> . . . . .	<b>280</b>
E.1. North Pole Folding around a T-Point . . . . .	281
E.2. North Pole Folding around a F-Point . . . . .	285
<b>F. A brief guide to the DOMAINcfg tool</b> . . . . .	<b>289</b>
F.1. Choice of horizontal grid . . . . .	290
F.2. Vertical grid . . . . .	291
F.2.1. Vertical reference coordinate . . . . .	291
F.2.2. Model bathymetry . . . . .	293
F.2.3. Choice of vertical grid . . . . .	294
F.3. Ice shelf cavity definition . . . . .	297
F.4. Closed sea mask definition ( <i>domclo.F90</i> ) . . . . .	297
<b>G. Coding Rules</b> . . . . .	<b>299</b>
G.1. Introduction . . . . .	300
G.2. Overview and general conventions . . . . .	300
G.3. Architecture . . . . .	301
G.4. Style rules . . . . .	301
G.4.1. Argument list format . . . . .	301
G.4.2. Array syntax . . . . .	301
G.4.3. Case . . . . .	301
G.4.4. Comments . . . . .	302
G.4.5. Continuation lines . . . . .	302
G.4.6. Declaration of arguments and local variables . . . . .	302
G.4.7. F90 Standard . . . . .	302
G.4.8. Free-Form Source . . . . .	302
G.4.9. Indentation . . . . .	303
G.4.10. Loops . . . . .	303
G.4.11. Naming Conventions: files . . . . .	303
G.4.12. Naming Conventions: modules . . . . .	303
G.4.13. Naming Conventions: variables . . . . .	303
G.4.14. Operators . . . . .	303
G.4.15. Pre processor . . . . .	303
G.5. DO LOOP macros . . . . .	304
G.6. Content rules . . . . .	305
G.6.1. Configurations . . . . .	305
G.6.2. Constants . . . . .	305
G.6.3. Declaration for variables and constants . . . . .	306
G.6.4. Headers . . . . .	306
G.6.5. Interface blocks . . . . .	306
G.6.6. I/O Error Conditions . . . . .	306
G.6.7. PRINT - ASCII output files . . . . .	307
G.6.8. Precision . . . . .	307
G.6.9. Structures . . . . .	308
G.7. Packages coding rules . . . . .	308
G.7.1. Bounds checking . . . . .	308
G.7.2. Communication . . . . .	308
G.7.3. Error conditions . . . . .	308
G.7.4. Memory management . . . . .	308
G.7.5. Optimisation . . . . .	309
G.7.6. Package attribute: PRIVATE, PUBLIC, USE, ONLY . . . . .	309
G.7.7. Parallelism using MPI . . . . .	309
G.8. Features to be avoided . . . . .	309

<b>Indices</b>	<b>x</b>
Namelist blocks . . . . .	x
CPP keys . . . . .	xi
FORTTRAN modules . . . . .	xii
Namelist parameters . . . . .	xiii
FORTTRAN subroutines . . . . .	xvii

## List of Figures

1.1. Ocean boundary conditions . . . . .	5
1.2. Geographical and curvilinear coordinate systems . . . . .	7
1.3. Curvilinear $z$ -coordinate systems ( $\{\text{non-}\}$ linear free-surface cases and re-scaled $z^*$ ) . . . . .	11
2.1. Forcing integration methods for modified leapfrog (top and bottom) . . . . .	20
2.2. Leapfrog time stepping sequence with split-explicit free surface . . . . .	22
2.3. Partitioning coefficient used to partition vertical velocities into parts . . . . .	25
2.4. OVERFLOW: time-series of temperature vertical cross-sections . . . . .	25
2.5. OVERFLOW: sample temperature vertical cross-sections from mid- and end-run . . . . .	26
2.6. OVERFLOW: maximum partitioning coefficient during a series of test runs . . . . .	27
2.7. OVERFLOW: maximum partitioning coefficient for the case overlaid . . . . .	27
3.1. Arrangement of variables in the unit cell of space domain . . . . .	29
3.2. Comparison of grid-point position, vertical grid-size and scale factors . . . . .	30
3.3. Horizontal integer indexing . . . . .	32
3.4. Vertical integer indexing . . . . .	32
3.5. Ocean bottom regarding coordinate systems ( $z$ , $s$ and hybrid $s - z$ ) . . . . .	35
3.6. Overview of the substitution mechanism with respect to vertical keys . . . . .	35
3.7. Ocean bottom regarding $zps$ -coordinate systems . . . . .	36
4.1. Split-explicit time stepping scheme for the external and internal modes . . . . .	41
5.1. Triads used in the energy and enstrophy conserving scheme (EEN) . . . . .	47
6.1. Ways to evaluate the tracer value and the amount of tracer exchanged . . . . .	60
6.2. Combination of direction of action of diffusive operators and of the choice of vertical coordinate . . . . .	65
6.3. Penetration profile of the downward solar irradiance calculated by four models . . . . .	70
6.4. Geothermal heat flux . . . . .	70
6.5. Advective/diffusive bottom boundary layer . . . . .	72
7.1. Reconstruction of the diurnal cycle variation of short wave flux . . . . .	103
7.2. Reconstruction of the diurnal cycle variation of short wave flux on an ORCA2 grid . . . . .	104
8.1. Ice shelf location and fresh water flux definition . . . . .	110
9.1. Lateral boundary at $T$ -level . . . . .	122
9.2. Lateral boundary conditions . . . . .	123
9.3. Setting of east-west cyclic boundary conditions for <code>nn_hls=2</code> . The original global domain (without halos) is delimited by the bold lines. . . . .	124
9.4. Horizontal domain decomposition in $3 \times 3$ subdomains. The thick line on the left panel delimits the original domain (without halos). Subdomains numbering starts at 0 from the bottom-left subdomain. Communications of the subdomain 4 with its neighbours are represented by the blue arrows. . . . .	125
9.5. Atlantic domain defined for the CLIPPER projet . . . . .	126
9.6. Positioning of a sub-domain when massively parallel processing is used . . . . .	127



9.7. Geometry of unstructured open boundaries . . . . .	132
9.8. Header for a <i>coordinates.bdy.nc</i> file . . . . .	133
10.1. Averaging procedure for isopycnal slope computation . . . . .	138
10.2. Vertical profile of the slope used for lateral mixing in the mixed layer . . . . .	139
11.1. Mixing length computation . . . . .	149
11.2. The structure of the entraining boundary layer. (a) Mean buoyancy profile. (b) Profile of the buoyancy flux. . . . .	156
11.3. Subgrid kinetic energy integration in GLS and TKE schemes . . . . .	158
11.4. Unstable density profile treated by the non penetrative convective adjustment algorithm . . . . .	159
11.5. Diapycnal diffusivities for temperature and salt in regions of salt fingering and diffusive convection	161
11.6. Internal tide energy dissipation . . . . .	167
12.1. Sub-basin decomposition used to compute transports and the meridional stream-function . . . . .	191
13.1. Observational weights with a rectangular footprint . . . . .	205
13.2. Observational weights with a radial footprint . . . . .	205
13.3. Observations with the geographical distribution . . . . .	207
13.4. Observations with the round-robin distribution . . . . .	208
13.5. Main window of dataplot . . . . .	213
13.6. Profile plot from dataplot . . . . .	214
16.1. Two methods to defined the Gibraltar strait . . . . .	225
16.2. Mask fields for the <i>closea.F90</i> module . . . . .	226
17.1. ORCA mesh conception . . . . .	232
17.2. Horizontal scale factors and ratio of anisotropy for ORCA 0.5° mesh . . . . .	233
17.3. Snapshot of relative vorticity at the surface of the model domain in GYRE R9, R27 and R54 . . . . .	235
D.1. Triads arrangement and tracer gradients to give lateral and vertical tracer fluxes . . . . .	268
D.2. Triad notation for quarter cells . . . . .	268
D.3. Boundary triads . . . . .	273
D.4. Definition of mixed-layer depth and calculation of linearly tapered triads . . . . .	279
E.1. North fold boundary for the T grid, with a <i>T</i> -point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk routine (found in modules] lbclnk@lbclnk.F90 lbclnk.F90 module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by othe inner domain values when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk . Cells with the same color are at the same geographical location. Color shading shows the change in gradient on each side of the pivot-points.	281
E.2. North fold boundary for the U grid, with a <i>T</i> -point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk routine (found in modules] lbclnk@lbclnk.F90 lbclnk.F90 module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by othe inner domain values when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk . Cells with the same color are at the same geographical location. Signs + or - are illustrating the change of signe on each side of the pivotal point. . . . .	282
E.3. North fold boundary for the V grid, with a <i>T</i> -point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk routine (found in modules] lbclnk@lbclnk.F90 lbclnk.F90 module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by othe inner domain values when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk . Cells with the same color are at the same geographical location. Signs + or - are illustrating the change of signe on each side of the pivotal point. . . . .	283

E.4.	North fold boundary for the F grid, with a $T$ -point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk routine (found in modules] lbclnk@lbclnk.F90 lbclnk.F90 module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by othe inner domain values when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk . Cells with the same color are at the same geographical location. . . . .	284
E.5.	North fold boundary for the T grid, with a $F$ -point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk routine (found in modules] lbclnk@lbclnk.F90 lbclnk.F90 module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by othe inner domain values when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk . Cells with the same color are at the same geographical location. Color shading shows the change in gradient on each side of the pivot-points.	285
E.6.	North fold boundary for the U grid, with a $F$ -point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk routine (found in modules] lbclnk@lbclnk.F90 lbclnk.F90 module). Cells with the same color are at the same geographical location. Signs + or - are illustrating the change of signe on each side of the pivotal point. . . . .	286
E.7.	North fold boundary for the V grid, with a $F$ -point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk routine (found in modules] lbclnk@lbclnk.F90 lbclnk.F90 module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by othe inner domain values when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk . Cells with the same color are at the same geographical location. Signs + or - are illustrating the change of signe on each side of the pivotal point. . . . .	287
E.8.	North fold boundary for the F grid, with a $F$ -point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk routine (found in modules] lbclnk@lbclnk.F90 lbclnk.F90 module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by othe inner domain values when calling the subroutines]lbc_lnk@lbc_lnk lbc_lnk . Cells with the same color are at the same geographical location. . . . .	288
F.1.	DOMAINcfg: default vertical mesh for ORCA2 . . . . .	291
F.2.	DOMAINcfg: examples of the stretching function applied to a seamount . . . . .	296
F.3.	DOMAINcfg: comparison of $s$ - and $z$ -coordinate . . . . .	296

## List of Namelists

2.1. &namrun . . . . .	21
2.2. &namtsd . . . . .	23
3.1. &namcfg . . . . .	33
3.2. &namdom . . . . .	37
4.1. &namdyn_spg . . . . .	39
5.1. &namdyn_adv . . . . .	45
5.2. &namdyn_vor . . . . .	46
5.3. &namdyn_hpg . . . . .	50
5.4. &namdyn_ldf . . . . .	52
5.5. &namwad . . . . .	55
6.1. &namtra_adv . . . . .	59
6.2. &namtra_ldf . . . . .	64
6.3. &namtra_qsr . . . . .	68
6.4. &nambbc . . . . .	70
6.5. &namtbl . . . . .	71
6.6. &namtra_dmp . . . . .	73
6.7. &nameos . . . . .	74
7.1. &namsbc . . . . .	79
7.2. &namsbc_sas . . . . .	84
7.3. &namsbc_flx . . . . .	86
7.4. &namsbc_blk . . . . .	87
7.5. &namsbc_abl . . . . .	93
7.6. &namsbc_cpl . . . . .	95
7.7. &namsbc_apr . . . . .	95
7.8. &nam_tide . . . . .	96
7.9. &namsbc_rnf . . . . .	97
7.10. &namsbc_wave . . . . .	99
7.11. &namsbc_ssr . . . . .	105
7.12. &namsbc_fwb . . . . .	106
8.1. &namisf . . . . .	109
8.2. &namberg . . . . .	116
9.1. &namlbc . . . . .	122
9.2. &namppp . . . . .	125
9.3. &nambdy . . . . .	128
9.4. &nambdy_dta . . . . .	129
9.5. &nambdy & nambdy_index . . . . .	129
9.6. &nambdy_tide . . . . .	134

10.1. &namtra_eiv	142
10.2. &namldf_eke	143
10.3. &namtra_mle	143
11.1. &namzdf	146
11.2. &namzdf_ric	147
11.3. &namzdf_tke	148
11.4. &namzdf_gls	152
11.5. &namzdf_osm	154
11.6. &namdrg	162
11.7. &namdrg_top	162
11.8. &namdrg_bot	162
11.9. &namzdf_iwm	166
12.1. &namnc4	183
12.2. &namtrd	184
12.3. &namtrc_trd	184
12.4. &nam_diadct	185
13.1. &namobs	195
13.2. &namobs_dta	196
13.3. &namsao	209
14.1. &nam_asminc	218
15.1. &namsto	222
16.1. &namclo	225
16.2. &namctl	227
17.1. &namcfg	231
17.2. &namusr_def	235
F.1. &namdom_domcfg	290
F.2. &namzgr_sco_domcfg	295
F.3. &namzgr_isf	297
F.4. &namclo	298

## List of Tables

2.1. CFL stability criteria $\alpha^{\max}$ for LF (with Asselin parameter $\gamma$ ) and RK3 time discretizations combined with the most commonly used linear advection schemes. Specific acronyms : C2 = second-order centered, UP3 = third-order upwind, C4 = fourth-order centered, UP5 = fifth-order upwind, Co4 = fourth-order compact. Table extracted from Madec et al. (2024). . . . .	18
2.2. Advective CFL criteria for the leapfrog with Robert Asselin filter time-stepping . . . . .	24
3.1. Location of grid-points . . . . .	29
6.1. Standard value of S-EOS coefficients . . . . .	75
7.1. Ocean variables provided to the surface module) . . . . .	80
7.2. Naming nomenclature for climatological or interannual input file . . . . .	81
8.1. Description of the parameters hard coded into the ISF module . . . . .	111
10.1. Description of expected input files if mixing coefficients are read from NetCDF files . . . . .	140
11.1. Set of predefined GLS parameters or equivalently predefined turbulence models available . . . . .	153
12.1. "xios" context variables in <i>iodef.xml</i> . . . . .	171
12.2. Hierarchy of tag scopes used by XIOS . . . . .	174
12.3. XIOS contexts used by <i>NEMO</i> . . . . .	174
12.4. XIOS tags used by the xios context . . . . .	174
12.5. XIOS tags used by the nemo contexts . . . . .	175
12.6. File name placeholder strings and their substitutions . . . . .	178
12.7. XML attributes set automatically by <i>NEMO</i> . . . . .	179
12.8. Filesize comparison between NetCDF3 and NetCDF4 with chunking and compression . . . . .	184
12.9. Transport section slope coefficients . . . . .	187
17.1. Domain size of ORCA family configurations . . . . .	234
F.1. Default vertical mesh in $z$ -coordinate for 30 layers ORCA2 configuration . . . . .	293

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*<sup>\*</sup> for the sea-ice dynamics and thermodynamics, *TOP*<sup>†</sup> for the biogeochemistry (both transport and sources minus sinks (*PISCES*<sup>‡</sup>)). The ocean component has been developed from the legacy of the *OPA*<sup>§</sup> 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 model design. Advice for users of the modelling framework is provided in the [NEMO User Guide](#).

## 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 height).

**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 and split-explicit free surface formulations are implemented. A number of numerical schemes are available for momentum advection (according to "flux" or "vector" formulations), for the computation of the pressure

---

\*Sea-Ice modelling Integrated Initiative

†Tracer in the Ocean Paradigm

‡Pelagic Interactions Scheme for Carbon and Ecosystem Studies

§Océan PArallélisé (French)

gradients, as well as for the advection of tracers (second or higher order advection schemes, including positive ones).

**Surface Boundary Condition (SBC, SAS, TDE)** 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 (*SI<sup>3</sup>*) 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).

**Land Ice Ocean interactions (ISF and ICB)** describes the **Land Ice Ocean** interactions available in *NEMO*. Land ice / ocean interactions mostly take place in Greenland and Antarctica. These include ice-shelves melting, glacier termini, icebergs melting, and surface and sub-glacial runoff from the ice sheet. This chapter describes how each of these processes can be explicitly represented, parametrised or specified in *NEMO* and how *NEMO* can be coupled to an ice sheet model.

**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 conditions. 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 (**L**ateral **D**i**F**fusion and vertical **Z** **D**i**F**fusion) 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)** describes model **In-Outputs Management** and specific online **DIA**gnostics. The diagnostics includes the output of all the tendencies of the momentum and tracers equations, the output of tracers **TRenDs** 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 for data assimilation, it is a fantastic tool for model and data comparison.

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

**Stochastic Parametrization of EOS (STO)**

**Miscellaneous Topics** (including solvers)

**Configurations** provides a list of the reference configurations, text cases, known well documented regional configurations (and links to further details) along with a summary of the *ORCA* (global) family 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**



## Table of contents

1.1. Primitive equations . . . . .	4
1.1.1. Vector invariant formulation . . . . .	4
1.1.2. Boundary conditions . . . . .	5
1.2. Horizontal pressure gradient . . . . .	6
1.2.1. Pressure formulation . . . . .	6
1.2.2. Free surface formulation . . . . .	6
1.3. Curvilinear $z$ -coordinate system . . . . .	7
1.3.1. Tensorial formalism . . . . .	7
1.3.2. Continuous model equations . . . . .	8
1.4. Curvilinear generalised vertical coordinate system . . . . .	9
1.4.1. $S$ -coordinate formulation . . . . .	10
1.4.2. Curvilinear $z^*$ -coordinate system . . . . .	11
1.4.3. Curvilinear terrain-following $s$ -coordinate . . . . .	12
1.4.4. Curvilinear $\tilde{z}$ -coordinate . . . . .	13
1.5. Subgrid scale physics . . . . .	13
1.5.1. Vertical subgrid scale physics . . . . .	14
1.5.2. Formulation of the lateral diffusive and viscous operators . . . . .	14

## Changes record

Release	Author(s)	Modifications
5.0	<i>Sébastien Masson, Katherine Hutchinson and Gurvan Madec</i>	<i>Full review completed</i>
4.0	<i>Mike Bell</i>	<i>Review</i>
3.6	<i>Tim Graham and Gurvan Madec</i>	<i>Updates</i>
$\leq 3.4$	<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  $U$  is assumed to be zero.

$$\nabla \cdot U = 0 \quad (1.3)$$

*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:  $U$  the vector velocity,  $U = U_h + w k$  (the subscript  $h$  denotes the local horizontal vector, *i.e.* over the  $(i, j)$  plane),  $T$  the potential temperature,  $S$  the salinity,  $\rho$  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

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

– the heat and salt conservation equations

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

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

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

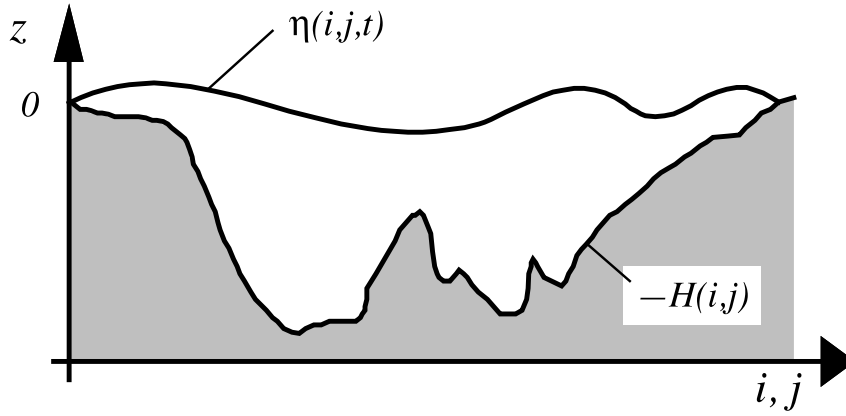


Figure 1.1.: The ocean is bounded by two surfaces,  $z = -H(i, j)$  and  $z = \eta(i, j, t)$ , where  $H$  is the depth of the sea floor and  $\eta$  the height of the sea surface. Both  $H$  and  $\eta$  are referenced to  $z = 0$ .

### 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  $\eta$  is the height of the sea surface (discretisation can introduce additional artificial “side-wall” boundaries). Both  $H$  and  $\eta$  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 6.4.3).

**Land ice - ocean** The ocean and land ice (ice shelves and drifting icebergs) exchange heat and fresh water. These interactions represent a significant source of fresh water for the polar ocean regions. Ice shelves melt in the 200m to 1500m depth range and the buoyant plume generated by the melt triggers specific oceanic circulation and water masses alterations on the Antarctic continental shelf or within Greenlandic Fjords. Icebergs, on the other hand, are created by the breaking of an ice shelf at the calving front. Their drift is subsequently driven by winds and ocean currents, redistributing the ice calved from the ice shelf or glacier termini along the Antarctic or Greenland coastline, and toward lower latitudes. Explicit representation and parametrisation of ice shelves and icebergs is described in ?? and ?? respectively.

## 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:

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

The surface pressure term is introduced via a new variable  $\eta$ , the free surface elevation, for which a prognostic equation can be established and solved. 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.

### 1.2.2. Free surface formulation

In the free surface formulation, a variable  $\eta$ , 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):

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

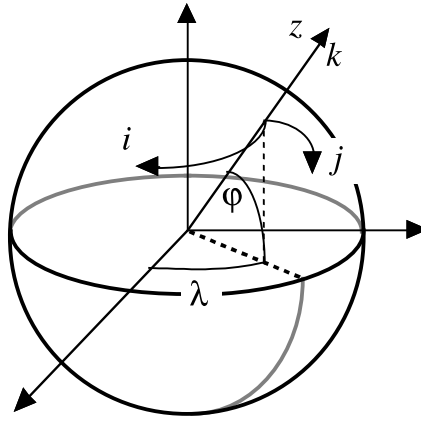
and using (equation 1.2) the surface pressure is given by:  $p_s = \rho g \eta$ .

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 4.1.1 or subsection 4.1.2).
- 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 (Roullet 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 (Roullet and Madec, 2000). With the present release, NEMO offers the choice between an explicit free surface (see subsection 4.1.1) or a split-explicit scheme strongly inspired the one proposed by Shchepetkin and McWilliams (2005) (see subsection 4.1.2).


 Figure 1.2.: the geographical coordinate system  $(\lambda, \varphi, z)$  and the curvilinear coordinate system  $(i, j, k)$ .

## 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 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  $(\lambda, \varphi, z)$  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:

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

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

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

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

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

where  $q$  is a scalar quantity and  $A = (a_1, a_2, a_3)$  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  $U = (u, v, w) = U_h + w k$ , the velocity in the  $(i, j, k)$  coordinates system, and define the relative vorticity  $\zeta$  and the divergence of the horizontal velocity field  $\chi$ , 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:

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

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:

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

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:

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

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

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

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

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

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

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

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

$$p_s = \rho g \eta$$

and  $\eta$  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,  $\chi$  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 7](#).

## 1.4. Curvilinear generalised vertical coordinate system

The ocean domain presents a huge diversity of situations 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  $\sigma$ -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 mix these coordinate systems, as in HYCOM (mixture of  $z$ -coordinate at the surface, isopycnic coordinate in the ocean interior and  $\sigma$  at the ocean



bottom) (Chassignet et al., 2003) or OPA (mixture of  $z$ -coordinate in vicinity the surface and steep topography areas and  $\sigma$ -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 **A**rbitrary **L**agrangian–**E**ulerian (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. The vertical direction is a quasi-vertical direction, which is oriented normal to a constant level surface in the vertical coordinate system.

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  $\sigma$ -coordinates as special cases ( $s = z$ ,  $s = z^*$ , and  $s = \sigma = z/H$  or  $= z/(H + \eta)$ , resp.). A formal derivation of the transformed equations is given in appendix A. Let us define the vertical scale factor by  $e_3 = \partial_s z$  ( $e_3$  is now a function of  $(i, j, k, t)$ ), and the slopes in the  $(i, j)$  directions between  $s$ - and  $z$ -surfaces by:

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

We also introduce  $\omega$ , 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

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

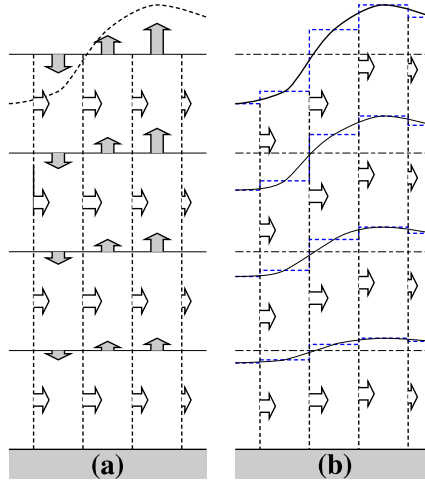


Figure 1.3.: Sea level high, position of the vertical levels and vertical velocities in a barotropic flow. (a)  $z$ -coordinate in linear free-surface case: fixed position levels and non-null vertical velocities; (b) re-scaled height coordinate (become popular as the  $z^*$ -coordinate (Adcroft and Campin, 2004)): time-varying position levels and null vertical velocities (if pure barotropic flow).

### Flux form of the momentum equation

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

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

where the relative vorticity,  $\zeta$ , 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.  $\omega$  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

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

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

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 ( $\delta z^*$ ) 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  $\eta$  coordinate used in many atmospheric models (see [Black \(1994\)](#) for a review of  $\eta$  coordinate atmospheric models). It was originally used in ocean models by [Stacey et al. \(1995\)](#) for studies of tides next to shelves, and it has been recently promoted by [Adcroft and Campin \(2004\)](#) for global climate modelling.

The surfaces of constant  $z^*$  are quasi-horizontal. Indeed, the  $z^*$  coordinate reduces to  $z$  when  $\eta$  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 10.2](#) 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  $\beta$ -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  $\sigma$ -coordinate (i.e. a depth-normalised coordinate system  $\sigma = z/H$ ), [Haney \(1991\)](#) and [Beckmann and Haidvogel \(1993\)](#) have given estimates of the magnitude of this truncation error. It depends on topographic slope, stratification, horizontal and vertical resolution, 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 3.2.2](#)).

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 10](#)).

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 \(2011\)](#). Available in *NEMO* versions 3.4 to 4.2, it was not robust enough to be used in all possible configurations and its use was not recommended. The  $\tilde{z}$ -coordinate was temporarily removed from *NEMO* version 5.0 to be better reintroduced in a future version.

### 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 7](#) and [chapter 11](#) and [section 6.5](#)). 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 11](#)).

### 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 \(1990\)](#) and [Fox-Kemper et al. \(2008\)](#) parameterisations, and various slightly diffusive advection schemes. For momentum, the main ones are: Laplacian and bilaplacian operators acting along geopotential surfaces or iso-neutral surfaces for the Laplacian,



and the 2nd order centered or the 3rd order upstream-biased 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](#)):

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

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.  $\mathfrak{R}$  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  $\sigma_1$  and  $\sigma_2$ , 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:

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

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 10](#)).

### Lateral bilaplacian tracer diffusive operator

The lateral bilaplacian tracer diffusive operator is defined by:

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

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. A rotated version of the bilaplacian tracer diffusive operator is available, following the work of [Lemarié et al. \(2012\)](#).

### 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](#)):

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

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:

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

where  $\mathfrak{R}$  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  $\beta$ -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 schemes: MLF and RK3	18
2.2.	MLF scheme	18
2.2.1.	Non-diffusive part	18
2.2.2.	Diffusive part — Forward or backward scheme	19
2.2.3.	Modified LeapFrog – Robert Asselin filter scheme (LF-RA)	19
2.3.	RK3 scheme	20
2.3.1.	Non-diffusive part	20
2.3.2.	Diffusive part — Forward or backward scheme	21
2.4.	Surface pressure gradient	21
2.5.	Start/Restart strategy	21
2.6.	Initial state ( <i>istate.F90</i> and <i>dtatsd.F90</i> )	23
2.7.	Adaptive-implicit vertical advection ( <i>ln_zad_Aimp</i> )	24
2.7.1.	Adaptive-implicit vertical advection in the OVERFLOW test-case	25

## Changes record

Release	Author(s)	Modifications
5.0	Jérôme Chanut Sibylle Téchéné	Update Review
4.0	Jérôme Chanut Tim Graham	Review Update
3.6	Christian Éthé	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 schemes: MLF and RK3

One can choose between two time stepping schemes for non-diffusive processes in *NEMO*. It can either be a three-stages, two-time level Runge Kutta scheme (RK3, [Wicker and Skamarock \(2002\)](#)) or a three-time level Modified LeapFrog scheme (MLF, [Mesinger and Arakawa \(1976\)](#); [Leclair and Madec \(2009\)](#)). MLF being the "historical" time stepping scheme, it is still the default. RK3 scheme can be activated by adding the cpp key `key_RK3` at compilation time.

The RK3 scheme, beside its higher formal accuracy than MLF (3<sup>rd</sup> order vs less than 2<sup>nd</sup> order for linear terms), has the advantage of requiring less memory storage and being more computationally efficient ([table 2.1](#)). As such, it will soon become the only time stepping option in *NEMO*. The reader is referred to ([Ducouso et al. \(2024\)](#) and [Madec et al. \(2024\)](#)) for further specific details about the implementation of the RK3 scheme in *NEMO*.

In the following, one describes the time evolution of a variable  $x$ , that stands for  $u$ ,  $v$ ,  $T$  or  $S$ . RHS is the **Right-Hand-Side** of the corresponding time evolution equation while  $\Delta t$  is the time step. The superscripts indicate the time at which a quantity is evaluated. Each term of the RHS is evaluated at a specific time step depending on the physics with which it is associated. In both schemes, the final value of  $x$ , *i.e.* at time  $t + \Delta t$ , is obtained where implicit vertical diffusion is computed, *i.e.* in the `trazdf.F90` and `dynzdf.F90` modules.

	$\alpha_{C2}^{\max}$	$\alpha_{UP3}^{\max}$	$\alpha_{C4}^{\max}$	$\alpha_{UP5}^{\max}$	$\alpha_{Co4}^{\max}$
LFRA ( $\gamma = 0.1$ )	$\sqrt{\frac{1-\gamma}{1+\gamma}} = 0.904534$	0.47074	0.659175	Unst.	$\sqrt{\frac{3}{11}} = 0.522233$
RK3	$\sqrt{3} = 1.73205$	1.62589	1.26222	1.43498	1

Table 2.1.: CFL stability criteria  $\alpha^{\max}$  for LF (with Asselin parameter  $\gamma$ ) and RK3 time discretizations combined with the most commonly used linear advection schemes. Specific acronyms : C2 = second-order centered, UP3 = third-order upwind, C4 = fourth-order centered, UP5 = fifth-order upwind, Co4 = fourth-order compact. Table extracted from [Madec et al. \(2024\)](#).

## 2.2. MLF scheme

### 2.2.1. Non-diffusive part

The LeapFrog (LF) time stepping is a three level scheme that can be represented as follows for non-diffusive processes:

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

This 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.

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  $\gamma$  is the Asselin coefficient.  $\gamma$  is initialized as `rn_atfp` (namelist parameter). Its default value is `rn_atfp=10.e-3` (see [subsection 2.2.3](#)), 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  $\gamma$ , 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.

### 2.2.2. 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 6.6](#)), 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 \(1967\)](#)).

### 2.2.3. 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

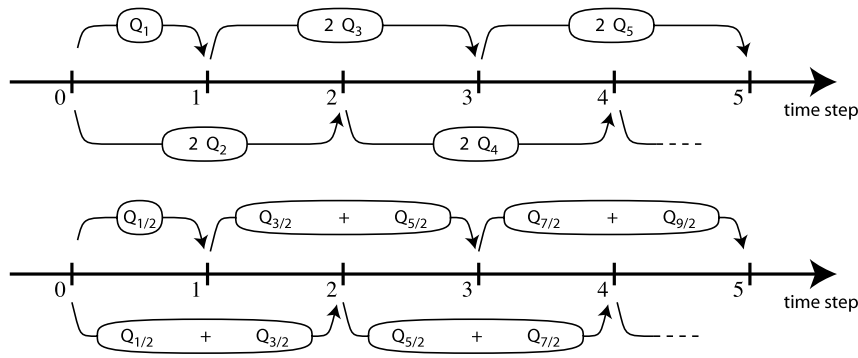


Figure 2.1.: Illustration of forcing integration methods. (top) "Traditional" formulation: the forcing is defined at the same time as the variable to which it is applied (integer value of the time step index) and it is applied over a  $2\Delta t$  period. (bottom) modified formulation: the forcing is 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.

$Q$  is redistributed over several time step. In the modified LF-RA environment, these two formulations have been replaced by:

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

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

The change in the forcing formulation given by [equation 2.6](#) (see [figure 2.1](#)) 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 11.2.2](#)) and of the TKE physics ([subsection 11.1.7](#)) (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  $\Delta t$  since the time filter is no longer applied to the forcing term.

## 2.3. RK3 scheme

### 2.3.1. Non-diffusive part

The RK3 time stepping is a three-stages, two-time level scheme that can be represented as follows (for non-diffusive processes):

$$\begin{aligned} x^{t+\Delta t/3} &= x^t + \frac{\Delta t}{3} \text{RHS}_x^t \\ x^{t+\Delta t/2} &= x^t + \frac{\Delta t}{2} \text{RHS}_x^{t+\Delta t/3} \\ x^{t+\Delta t} &= x^t + \Delta t \text{RHS}_x^{t+\Delta t/2} \end{aligned} \quad (2.8)$$

RHS applies here to the Coriolis force, momentum/tracer advection and hydrostatic pressure terms. From [equation 2.8](#), it appears that in order to advance from time  $t$  to time  $t + \Delta t$ , the RK3 scheme requires 3 evaluations of the right-hand-side whereas only one evaluation is necessary for the LF scheme. However, it must be clear that it does not necessarily mean that the time-to-solution will be longer with RK3. Indeed, larger time-steps are theoretically allowed with RK3 ( $\Delta t$  can be doubled actually, see [table 2.1](#)) and several costly terms (vertical mixing parameterization, rotated diffusion, viscous/diffusive operators) are computed only once per time-steps. Overall the transition from LF to RK3 is expected to lead to a more efficient code on top of the increased accuracy in time.

```

-----
&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)
ln_ist_euler = .false. ! =T force a start with forward time step (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
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_xios 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
ln_top      = .true.  ! Consider (T) or bypass (F) the TOP routines when the key_top is activated
/

```

namelist 2.1.: &namrun

### 2.3.2. Diffusive part — Forward or backward scheme

Similarly to the LF scheme, tendencies due to lateral diffusion or restoring (section 6.6) are applied thanks to a forward in time differencing:

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

These are computed once per time step and added to non-diffusive tendencies at the third stage of equation 2.8. Vertical diffusion processes follow, here again, an unconditionally stable, backward in time differencing, performed at stage 3:

$$x^{t+\Delta t} = x^t + \Delta t \text{RHS}_x^{t+\Delta t} \quad (2.9)$$

## 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. The same applies to RK3, the surface pressure gradient being considered in the 3 stages described in subsection 2.3.1. This refers to as the explicit free surface case in the code ( `ln_dynspg_exp=.true.` , which is not available for RK3 at the time of writing). 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 uses 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 2.2 for MLF scheme) and the mode splitting stability (Ducousso et al. (2024)).

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. Start/Restart strategy

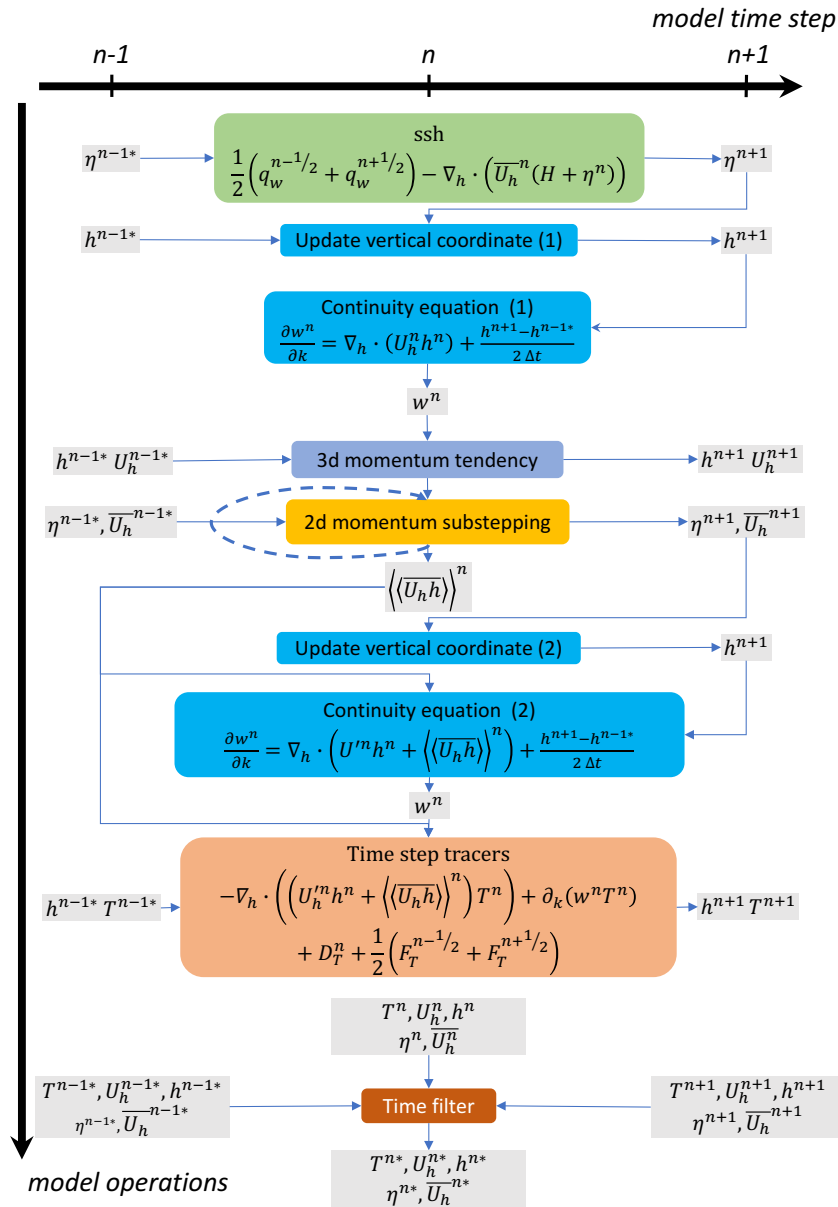


Figure 2.2.: Sketch of the leapfrog time stepping sequence in *NEMO* with split-explicit free surface. The latter combined with non-linear free surface requires the dynamical tendency being updated prior tracers tendency to ensure conservation. Note the use of time integrated fluxes issued from the barotropic loop in subsequent calculations of tracer advection and in the continuity equation. Details about the time-splitting scheme can be found in ??.

```

!-----
&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_1m_potential_temperature_nomask', -1. , 'votemper', .true. , .true. , 'yearly' , '' ,
↪ ,
sn_sal = 'data_1m_salinity_nomask' , -1. , 'vosaline', .true. , .true. , 'yearly' , '' ,
↪ ,
/

```

namelist 2.2.: &namtsd

When starting from initial conditions, and specific to the Leapfrog scheme (*i.e.* no specific procedure is needed with RK3), the first step 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 and setting `ln_restart=.true.`. 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 the necessary time levels (2 in case of LF and 1 with RK3) and many auxiliary data in the restart files in machine precision.

Note that the time step  $\Delta t$ , 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 with Leapfrog. 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.

The consistency between the provided input restart file(s) (`cn_ocerst_in`) and the namelist settings is handled with the parameter `nn_rstctl1`, as detailed in `&namrun` ([namelist 2.1](#)).

Restart files are created through the legacy interface, which allows to specify the root name of the output restart file (`cn_ocerst_out`) and the timestep frequency at which restart file is created (`nn_stock`). Similarly the parameter `nn_write` control the creation of output files. It is also possible to save restart files at specific timesteps during the model execution, by setting `ln_rst_list=.true.` and filling up the restart dump times in `nn_stocklist` (always requires 10 values).

Since version 4.2 it is possible to use the XIOS interface to directly write (`nn_wxios`) and read (`ln_xios_read`) the restart files, as detailed in ??.

## 2.6. Initial state ( *istate.F90* and *dtatsd.F90* )

Basic initial state options are defined in `&namtsd` ([namelist 2.2](#)). 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.`

Initial temperature and salinity input fields can be provided 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 7.2.2](#)). The information relating to the input files is specified in the `sn_tem` and `sn_sal` structures. The computation is done in the `dtatsd.F90` module.

`ln_tsd_init=.false.`

Initial temperature and salinity fields are set as part of a user-defined configuration (subroutine `usr_def_istate` in module `userdef_istate.F90` - see the [user guide](#) for more information). The default configuration (GYRE, see [section 17.5](#)) sets horizontally uniform temperature and salinity fields.



## 2.7. 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 2.2.

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, enabled by setting `ln_zad_Aimp=.true.` in `&namzdf` (namelist 11.1), 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 away 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, this option 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 <sup>nd</sup> order centered	3 <sup>rd</sup> order upwind	4 <sup>th</sup> order compact
Advective CFL criterion	0.904	0.472	0.522

Table 2.2.: 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:

$$\begin{aligned}
 Cu = \frac{2\Delta t}{e_{3t_{ijk}}^n} & \left( [\max(w_{ijk}^n, 0.0) - \min(w_{ijk+1}^n, 0.0)] \right. \\
 & + [\max(e_{2_u ij} e_{3_u ijk}^n u_{ijk}^n, 0.0) - \min(e_{2_u i-1j} e_{3_u i-1jk}^n u_{i-1jk}^n, 0.0)] / e_{1_t ij} e_{2_t ij} \\
 & \left. + [\max(e_{1_v ij} e_{3_v ijk}^n v_{ijk}^n, 0.0) - \min(e_{1_v ij-1} e_{3_v ij-1k}^n v_{ij-1k}^n, 0.0)] / e_{1_t ij} e_{2_t ij} \right) \quad (2.10)
 \end{aligned}$$

and the tapering algorithm follows Shchepetkin (2015) as:

$$\begin{aligned}
 Cu_{min} &= 0.15 \\
 Cu_{max} &= 0.3 \\
 Cu_{cut} &= 2Cu_{max} - Cu_{min} \\
 Fcu &= 4Cu_{max} * (Cu_{max} - Cu_{min}) \\
 Cf &= \begin{cases} 0.0 & \text{if } Cu \leq Cu_{min} \\ (Cu - Cu_{min})^2 / (Fcu + (Cu - Cu_{min})^2) & \text{else if } Cu < Cu_{cut} \\ (Cu - Cu_{max}) / Cu & \text{else} \end{cases} \quad (2.11)
 \end{aligned}$$

The partitioning coefficient ( $Cf$ ) 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:

$$\begin{aligned}
 w_{i_{ijk}} &= Cf_{ijk} w_{n_{ijk}} \\
 w_{n_{ijk}} &= (1 - Cf_{ijk}) w_{n_{ijk}} \quad (2.12)
 \end{aligned}$$

Note that the coefficient is such that the treatment is never fully implicit; the three cases from equation 2.11 can be considered as: fully-explicit; mixed explicit/implicit and mostly-implicit, with  $Cf$  varying as shown in figure 2.3. 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 criterion given in table 2.2 for a 3rd order scheme.

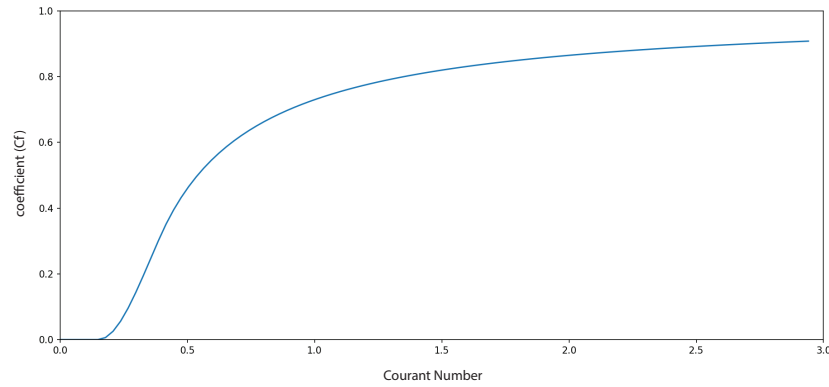


Figure 2.3.: The value of the partitioning coefficient ( $C_f$ ) used to partition vertical velocities into parts to be treated implicitly and explicitly for a range of typical Courant numbers (`ln_zad_Aimp=.true.`).

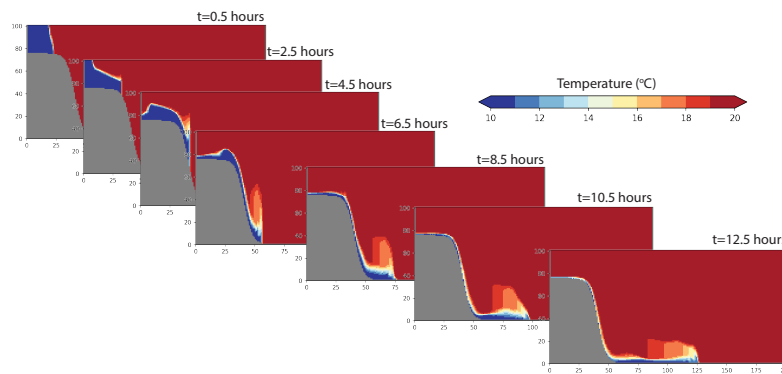


Figure 2.4.: A time-series of temperature vertical cross-sections for the OVERFLOW test case. These results are for the default settings with `rn_Dt=10.0` and without adaptive implicit vertical advection (`ln_zad_Aimp=.false.`).

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 Shepetchkin (2015). This is sufficient for the flux-limited advection scheme (`ln_traadv_mus=.true.`) but further intervention is required when using the flux-corrected advection scheme (`ln_traadv_fct=.true.`). For this scheme, 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:

### 2.7.1. 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
```

which were chosen to provide a slightly more stable and less noisy solution. The result when using the default value of `rn_Dt=10.` without adaptive-implicit vertical velocity is illustrated in figure 2.4. 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 `rn_Dt=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.

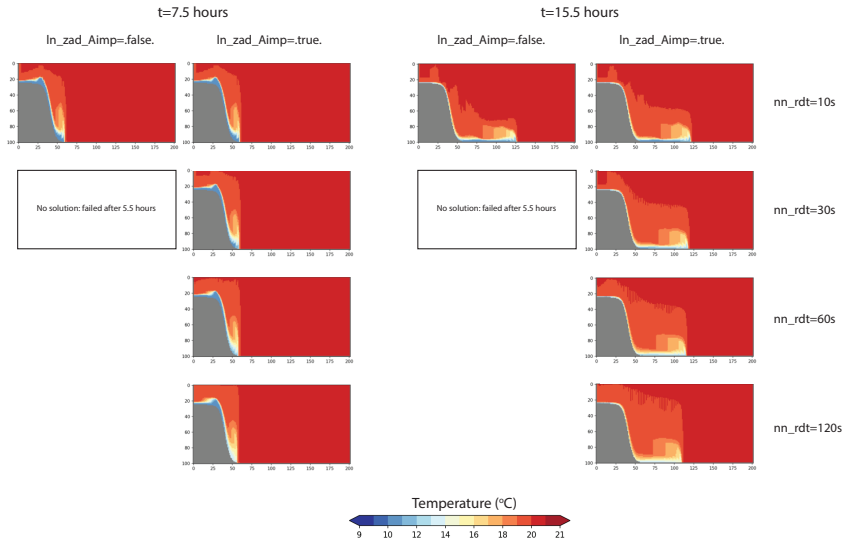


Figure 2.5.: Sample temperature vertical cross-sections from mid- and end-run using different values for `rn_Dt` and with or without adaptive implicit vertical advection. Without the adaptive implicit vertical advection, only the run with the shortest timestep is able to run to completion. Note also that the colour-scale has been chosen to confirm that temperatures remain within the original range of  $10^{\circ}$  to  $20^{\circ}$ .

The results with `ln_zad_Aimp=.true.` and a variety of model timesteps are shown in figure 2.5 (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 `rn_Dt=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 (when using XIOS, `key_xios`) via the `wimp` and `wexp` diagnostics respectively. Likewise, the partitioning coefficient ( $Cf$ ) is also available as `wi_cff`. For a quick oversight of the scheme's 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 16.4 for activation details).

figure 2.6 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 `rn_Dt=10.`, 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 locations 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 2.7 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 `rn_Dt=5400.` is already pushing at stability limits, especially in the initial start-up phase. The time-series does not, however, exhibit any of the 'cuspliness' found with the overflow tests.

A short test with an eORCA1 configuration is more promising, since using a time-step of `rn_Dt=3600.` remains stable with `ln_zad_Aimp=.true.` whereas the time-step is limited to `rn_Dt=2700.` without.

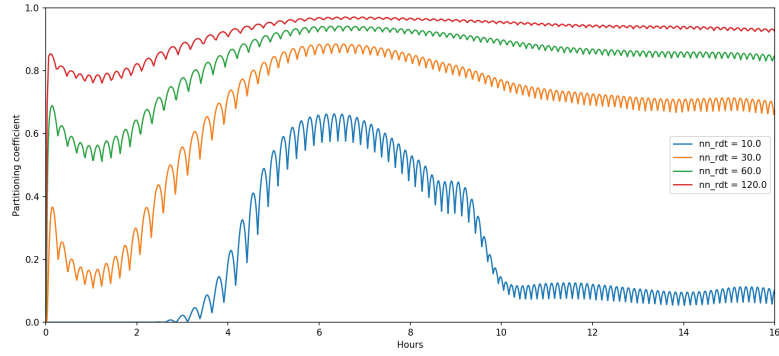


Figure 2.6.: The maximum partitioning coefficient during a series of test runs with increasing model timestep length. At the larger timesteps, the vertical velocity is treated mostly implicitly at some locations throughout the run.

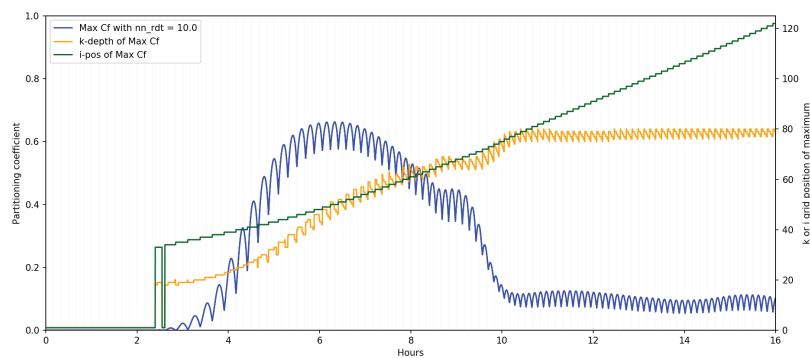


Figure 2.7.: The maximum partitioning coefficient for the  $rn\_Dt=10.0$  case overlaid with information on the gridcell  $i$ - and  $k$ -locations of the maximum value.

## Table of contents

3.1. Fundamentals of the discretisation . . . . .	29
3.1.1. Arrangement of variables . . . . .	29
3.1.2. Discrete operators . . . . .	30
3.1.3. Numerical indexing . . . . .	31
3.2. Spatial domain configuration . . . . .	32
3.2.1. Horizontal grid mesh ( <i>domhgr.F90</i> ) . . . . .	33
3.2.2. Vertical grid ( <i>domzgr.F90</i> ) . . . . .	33
3.2.3. Closed seas . . . . .	36
3.2.4. Output grid files . . . . .	37

## Changes record

Release	Author(s)	Modifications
5.0	<i>Sibylle Techene, Daley Calvert, and Simon Müller</i>	<i>Review, revision, and removal of a summary of model variables required to define the domain configuration</i>
4.0	<i>Simon Müller and Andrew Coward</i>	<i>Compatibility changes: many options moved to external domain configuration tools (see <a href="#">appendix F</a>)</i> <i>Updates</i>
3.6	<i>Simona Flavoni and Tim Graham</i> <i>Rachid Benshila, Christian Éthé, Pierre Mathiot and Gervan Madec</i>	<i>Updates</i>
$\leq 3.4$	<i>Gervan Madec and Sébastien Masson</i>	<i>First version</i>

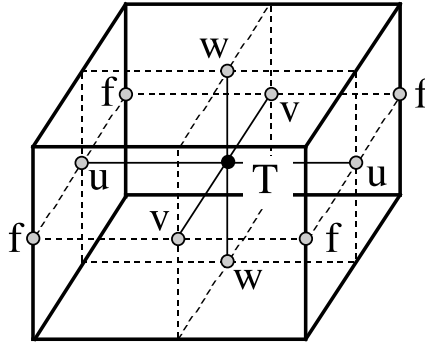


Figure 3.1.: Arrangement of variables in the unit cell of space domain.  $T$  indicates scalar points where temperature, salinity, density, pressure and horizontal divergence are defined.  $(u, v, w)$  indicates vector points, and  $f$  indicates vorticity points where both relative and planetary vorticities 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 values of the column, row, or level. This indexing is only used for the writing of the semi-discrete equations. In the code, the indexing uses integer values only (the relative grid-cell location has to be inferred from the context) 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. Note that *istate.F90* and *dtatsd.F90* are located in the `./src/OCE/DOM` directory. However, they are described alongside the model’s time domain chapter, together with the restart strategy.

## 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 ( $T$ , e.g. temperature  $T$ , salinity  $S$ , pressure  $p$ , density  $\rho$ ) with vector points  $(u, v, w)$  (velocity), whose components are defined in the centre of each cell face (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,  $\zeta$  and  $f$ , are defined at the centre of each vertical edge and the barotropic stream function  $\psi$  is defined at horizontal points overlying the  $\zeta$  and  $f$ -points.

The ocean mesh (i.e. the position of all the scalar and vector points) is defined by the transformation that gives  $(\lambda, \varphi, z)$  as a function of  $(i, j, k)$ . The grid-points are located at integer or integer and a half values of  $(i, j, k)$  as indicated on table 3.1. In all the following, subscripts  $T, u, v, w, f, uw, vw$  or  $fw$  indicate the position of the grid-point where the scale factors  $e_k$  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).

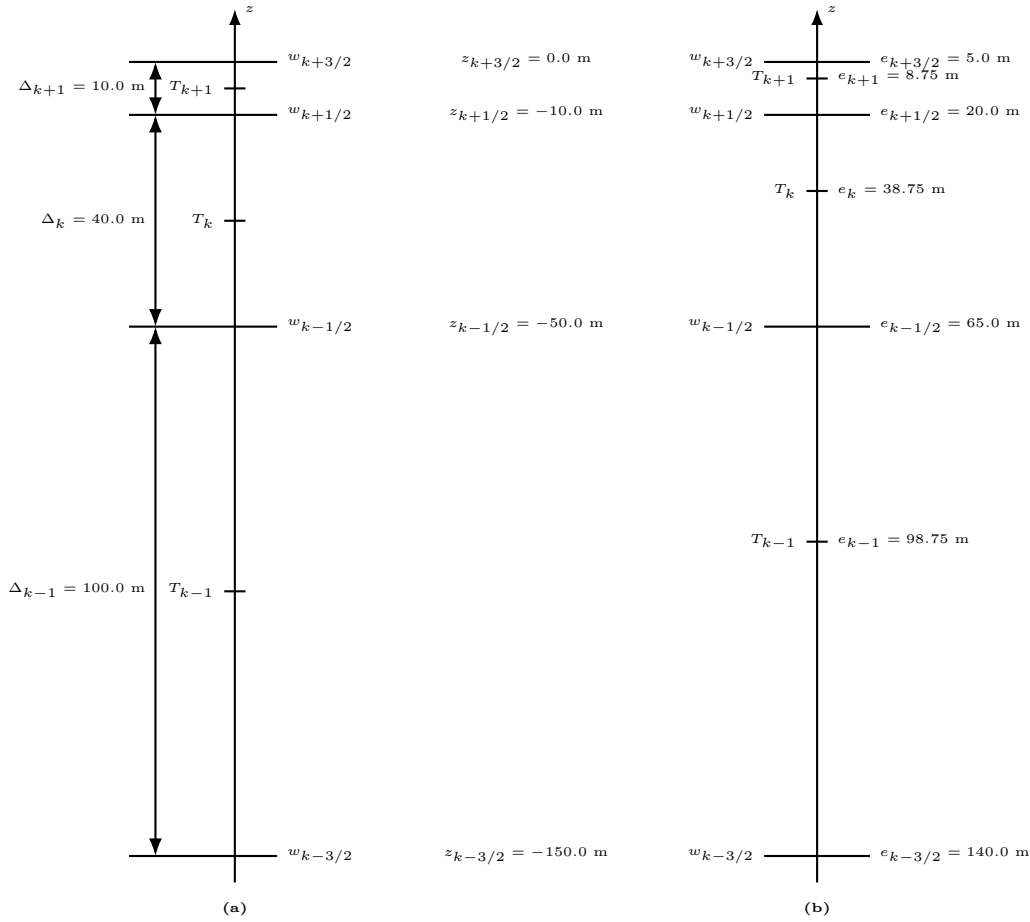


Figure 3.2.: Comparison of (a) traditional definitions of grid-point position and grid-size in the vertical, and (b) analytically derived grid-point position and scale factors. For both grids here, the same  $w$ -point depth has been chosen but in (a) the  $T$ -points are set halfway between  $w$ -points 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  $\Delta_k$  and those of the scale factor  $e_k$ .

Note that the definition of the scale factors (*i.e.* as the analytical first derivative of the transformation that results in  $(\lambda, \varphi, z)$  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: 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  $A = (a_1, a_2, a_3)$  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:

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

The vertical average over the whole water column is denoted by an overbar and is for a land-masked field  $q$  (*i.e.* a quantity that is equal to zero at land points):

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

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 ( $\delta_i$ ,  $\delta_j$  and  $\delta_k$ ) are skew-symmetric linear operators, and further that the averaging operators ( $\overline{\cdot\cdot\cdot}^i$ ,  $\overline{\cdot\cdot\cdot}^j$  and  $\overline{\cdot\cdot\cdot}^k$ ) are symmetric linear operators, *i.e.*

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

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

In other words, the adjoint of the differencing and averaging operators are  $\delta_i^* = -\delta_{i+1/2}$  and  $(\overline{\cdot\cdot\cdot}^i)^* = \overline{\cdot\cdot\cdot}^{i+1/2}$ , 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 such that the  $i$  and  $j$  indices increase towards the east and north of the domain.  $u$ -,  $v$ - and  $f$ -points are distributed such that a  $T$ -point has the same  $i$  index ( $j$  index) as its nearest eastward  $u$ -point (northward  $v$ -point) and the same  $i$ - and  $j$ -indices as its nearest north-east  $f$ -point (see the shaded area in [figure 3.3](#)).

#### 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 given in [subsection 3.1.1](#).

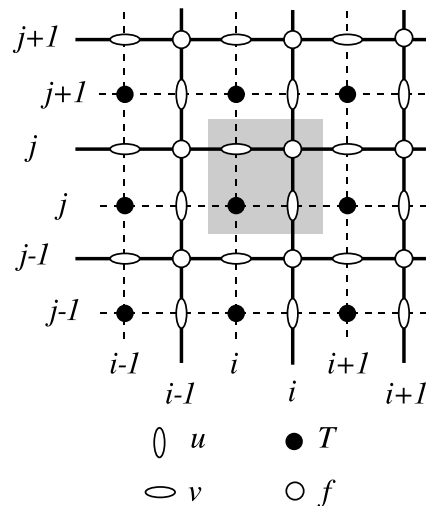


Figure 3.3.: Horizontal integer indexing used in the FORTRAN code. The shaded area indicates the cell in which variables contained in arrays have the same  $i$ - and  $j$ -indices

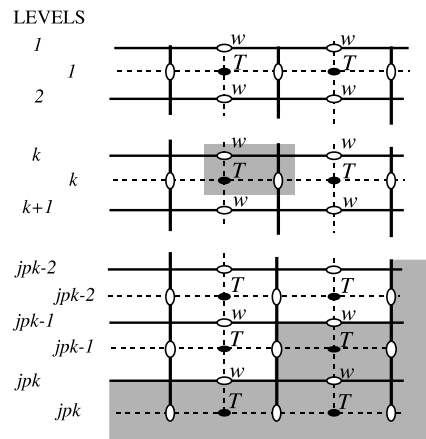


Figure 3.4.: Vertical integer indexing used in the FORTRAN code. Note that the  $k$ -axis is oriented downward. The topmost shaded area indicates the cell in which variables contained in arrays have a common  $k$ -index.

The  $w$ -level at  $k = 1$  corresponds to the sea surface, while the  $w$ -level at  $k = jpk$  either corresponds to or is below the ocean floor.  $T$ -levels are distributed such that the  $t$ -level at index  $k$  is between the  $w$ -levels at indices  $k$  and  $k + 1$ . This is in contrast to the indexing on the horizontal plane, where for example the  $T$ -point at index  $i$  is between the  $u$ -points at indices  $i$  and  $i - 1$  (compare the shaded areas 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 the implementation and documentation.

## 3.2. Spatial domain configuration

Two methods are available to specify the spatial domain configuration and are selected using the namelist parameter `ln_read_cfg` in namelist `&namcfg` ([namelist 17.1](#)) :

`ln_read_cfg=.true.`

The domain-specific parameters and fields are read from a NetCDF input file, whose name can be specified via the `cn_domcfg` parameter in namelist `&namcfg` ([namelist 17.1](#))

`ln_read_cfg=.false.`

The domain-specific parameters and fields are set as part of a user-defined configuration (modules `usrdef_nam.F90`, `usrdef_hgr.F90` and `usrdef_zgr.F90` - see the [user guide](#)).

From 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

```

!-----
&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 => fill namclo)
!                      ! (=F) no control of net precip/evap over closed sea
!
ln_write_cfg = .false.   ! (=T) create the domain configuration file
cn_domcfg_out = "domain_cfg_out" ! newly created domain configuration filename
/

```

namelist 3.1.: &namcfg

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 therefore been incorporated into an external tool (`./tools/DOMAINcfg`) which is briefly described in [appendix F](#).

The following subsections cover several topics relating to the spatial domain configuration.

### 3.2.1. Horizontal grid mesh ( *domhgr.F90* )

The values of the geographic longitude and latitude arrays at indices  $i, j$  correspond to the analytical expressions of the longitude  $\lambda$  and latitude  $\varphi$  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  $\lambda$  and  $\varphi$  with respect to  $i$  and  $j$ , evaluated for the same arguments as  $\lambda$  and  $\varphi$ . Longitudes, latitudes, and horizontal scale factors at  $w$ -points are exactly equal to those at  $T$ -points, thus no specific arrays are defined at  $w$ -points.

*NEMO* can support the local reduction of key strait widths by altering the values of horizontal scale factors at the appropriate locations. This is particularly useful for locations such as Gibraltar or Indonesian Throughflow pinch-points (see [section 16.1](#) for illustrated examples). The key is to reduce the surface area of  $T$ -cells without changing their volume, by altering the values of the horizontal scale factors at  $u$ - and  $v$ -points. Doing otherwise can lead to numerical stability issues.

Normally, the surface areas are computed from the product of the horizontal scale factors ( $e_{1u} * e_{2u}$  and  $e_{1v} * e_{2v}$ ). In cases where these need to be reduced, the modified surface areas (variables `e1e2u` and `e1e2v` at  $u$ - and  $v$ -points respectively) must either be read from the domain configuration file (see [section 3.2](#)) or calculated as part of a user-defined configuration (module `usrdef_hgr.F90` - see the [user guide](#) for more information).

Similar logic applies to the Coriolis parameter. This is normally calculated as  $2 * \Omega * \sin(\varphi)$ , but when the horizontal grid mesh is not on a sphere it must instead be specified (through variables `ff_f` and `ff_t` at  $f$ - and  $T$ -points respectively) using one of the above methods.

### 3.2.2. Vertical grid ( *domzgr.F90* )

The *NEMO* vertical mesh is defined using vertical scale factors, depths, and water heights, with each of these variables structured differently depending on the chosen coordinate system. It is important to distinguish between the temporal structure (how the coordinate system changes over time) and the spatial structure (how it varies across different locations) of this system.

A code substitution mechanism defined in `domzgr_substitute.h90` is used to replace proxy arrays describing the grid point depths (`gdept`, `gdepw`), water column heights (`ht`, `hu`, `hv`, `hf`) and vertical scale factors (`e3t`, `e3u`, `e3v`, `e3f`, `e3w`, `e3uw`, `e3vw`) with appropriate expressions. These expressions always include a reference array that defines the spatial structure of the vertical grid. In case of a non-linear free-surface, *NEMO* uses a quasi-eulerian coordinate and these proxy arrays take into account a dependency with the sea surface height; `key_qco` must be specified. In case of a linear free-surface approximation, vertical proxy arrays do not include any sea surface height dependency; `key_linssh` must be specified.

This formulation for the vertical coordinate arrays uses less memory than in previous versions of *NEMO*. It takes advantage of the use of specific keys (described in the following section) that selectively enable certain processes and data structures, allowing for more efficient memory use without compromising model accuracy where it is needed.

### Quasi-eulerian COordinate (QCO)

The quasi-eulerian coordinate (specified with `key_qco`) is the new name for the star framework ( $z^*$  or  $s^*$ ). The quasi-eulerian coordinate absorbs the divergence of horizontal barotropic velocities. Consequently, the vertical velocity resulting from the barotropic mode is converted into a variation in thickness. The vertical coordinate adapts to the time-varying free surface, making the transformation time-dependent, represented as  $z(i, j, k, t)$  (e.g. figure 3.5f).

The vertical mesh variables (grid point depths, water heights and vertical scale factors) are substituted by an expression (see figure 3.6), which in the case of the vertical scale factor at  $T$ -points (`e3t`) is:

$$e3t(i, j, k, t) \leftarrow e3t\_0(i, j, k) (1 + r3t(i, j, t)tmask(i, j, k))$$

Where  $r3t(i, j, t) = \frac{\eta(i, j, t)}{ht\_0(i, j)}$ , the ratio of sea surface height to reference water column height, is updated at every time step by `domqco_r3c` and where `e3t_0`, `ht_0` and `tmask` are the  $T$ -point variants of the reference vertical scale factor, the reference water column height, and the land-sea mask, respectively. Similar expressions are applied to the scale factors at  $u/v/f$ -points using appropriate interpolation of  $\eta$ . In the expressions for the scale factors at  $w$ -levels, grid point depths and water heights, the ratio is instead unmasked.

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 terrain-following coordinates. As such, the reference values of the grid point depths, water heights and vertical scale factors can be considered as those arising from a flat sea surface with zero elevation.

### Linear free surface

In the case of the linear free-surface approximation (`key_linssh` is specified), the free-surface variation is neglected compared to ocean depths. Then, vertical coordinates are independent from the sea surface height and they remain fixed over time. This setup does not treat the ocean surface as a rigid lid as it enables vertical seawater movement across the top boundary. In this case, the vertical mesh variables are simply substituted for their time-invariant reference counterparts, e.g.

$$e3t(i, j, k, t) \leftarrow e3t\_0(i, j, k)$$

The same substitution is applied to all other scale factors, grid point depths and water heights.

### Vertical coordinate system

The model mesh is initially determined by four elements when setting up the configuration:

1. the bathymetry specified in meters
2. the number of levels of the model (`jpK`)
3. the analytical transformation  $z(i, j, k)$  and the reference vertical scale factors  $e_{3x}^0(i, j, k)$  (derivatives of the transformation)
4. the masking system, *i.e.* the specification of the wet model levels at each  $(i, j)$  location of the horizontal grid

The definition of the reference vertical scale factors ( $e_{3x}^0$ ) is determined by the choice of vertical coordinate system. The vertical scale factors ( $e_{3x}^0$ ) form the basis for deriving all vertical variables. This choice is fixed for the duration of an experiment and cannot be modified midway. As with other components of the spatial domain configuration (see section 3.2), it must be specified either in the domain configuration file or as part of a user-defined configuration (module `usrdef_zgr.F90`). For further details, refer to the [user guide](#).

In addition a set of optimization keys (`key_vco *`) determine whether 1-dimensional or 3-dimensional definitions (as illustrated in figure 3.6) substitute for reference vertical variables, the intention being to minimise memory use. In the case of `key_vco_1d` reference vertical variables are 1-dimensional vertical arrays, it can only be used with a uniform grid. In the case of `key_vco_1d3d` reference vertical variables are 1-dimensional vertical arrays, except for reference scale factors at  $T$ -levels, it cannot be used with non uniform grid. In the case of `key_vco_3d` reference vertical variables are 3-dimensional arrays, it can be used sub-optimally in all cases.

Three main choices are offered (figure 3.5a-c):

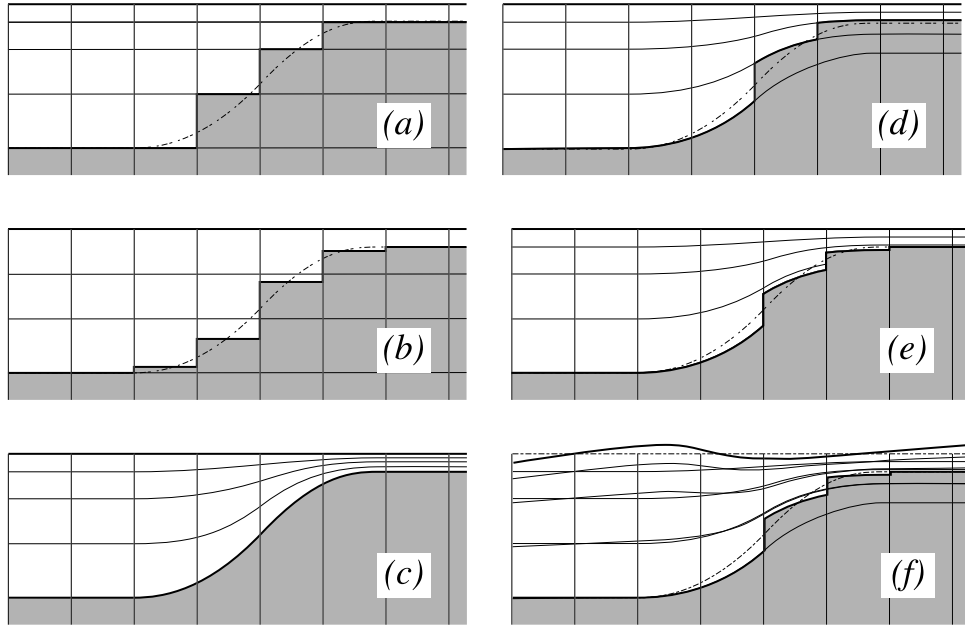


Figure 3.5.: The ocean bottom as seen by the model: (a)  $z$ -coordinate with full step, (b)  $z$ -coordinate with partial step, (c)  $s$ -coordinate: terrain following representation, (d) hybrid  $s - z$  coordinate, (e) hybrid  $s - z$  coordinate with partial step, and (f) same as (e) but in the non-linear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

	spacial contribution			ssh dependency contribution	
	key_vco_1d	key_vco_1d3d	key_vco_3d	key_qco	key_linssh
e3t	e3t_1d(k)	e3t_3d(i,j,k)	e3t_3d(i,j,k)	( 1 + r3t(i,j,t)*tmask(i,j,k) )	
e3u	e3t_1d(k)	e3u_3d(i,j,k)	e3u_3d(i,j,k)	( 1 + r3u(i,j,t)*umask(i,j,k) )	
e3v	e3t_1d(k)	e3v_3d(i,j,k)	e3v_3d(i,j,k)	( 1 + r3v(i,j,t)*vmask(i,j,k) )	
e3f	e3t_1d(k)	e3f_3d(i,j,k)	e3f_3d(i,j,k)	(1 + r3f(i,j,t)*fe3mask(i,j,k))	
e3w	e3w_1d(k)	e3w_1d(k)	e3w_3d(i,j,k)	( 1 + r3t(i,j,t) )	
e3uw	e3w_1d(k)	e3w_1d(k)	e3uw_3d(i,j,k)	( 1 + r3u(i,j,t) )	
e3vw	e3w_1d(k)	e3w_1d(k)	e3vw_3d(i,j,k)	( 1 + r3v(i,j,t) )	
gdept	gdept_1d(k)	gdept_1d(k)	gdept_3d(i,j,k)	( 1 + r3t(i,j,t) )	
gdepw	gdepw_1d(k)	gdepw_1d(k)	gdepw_3d(i,j,k)	( 1 + r3t(i,j,t) )	
ht	SUM_k e3t_1d	SUM_k e3t_3d	SUM_k e3t_3d	( 1 + r3t(i,j,t) )	
hu	SUM_k e3t_1d	SUM_k e3u_3d	SUM_k e3u_3d	( 1 + r3u(i,j,t) )	
hv	SUM_k e3t_1d	SUM_k e3v_3d	SUM_k e3v_3d	( 1 + r3v(i,j,t) )	

Figure 3.6.: Overview of the vertical arrays structure based on vertical space ( `key_vco_1d` , `key_vco_1d3d` or `key_vco_3d` ) and time keys ( `key_qco` and `key_linssh` )

`l_zco = .true. recommended with key_vco_1d`

$z$ -coordinate with full step bathymetry-  $e_{3x}^0$  will be uniform across each horizontal level

`l_zps = .true. recommended with key_vco_1d3d`

$z$ -coordinate with partial step bathymetry ( $zps$ -coordinate)-  $e_{3x}^0$  at  $T$ -levels at the bottom wet level (and, possibly, the top wet level if ice cavities are present) may vary from its horizontal neighbours

`l_sco = .true. recommended with key_vco_3d`

Generalized  $s$ -coordinate- variations in  $e_{3x}^0$  can occur throughout the water column

Hybrid combinations of the three main coordinates are also available such as  $s - z$  or  $s - zps$  coordinates (figure 3.5d and figure 3.5e).

**Partial cells description `l_zps = .true.`**

In  $zps$ -coordinates reference levels are based on the same spatially uniform levels as in  $z$ -coordinates. At the bottom (and at the top) a partial cell volume varies in order to take into account solid boundaries *i.e.* the bathymetry (and the ice-shelf cavities) more accurately.

In *NEMO* v4.2 and previous versions, partial cells were vertically shrunk, causing the mass center and the  $T$ -point location to shift, as illustrated in figure 3.7. All scale factors associated with these cells had to be adjusted accordingly.

In *NEMO* v5.0, partial cells are modeled as porous, consisting of both solid and liquid fractions that are distributed homogeneously within the cell. Cell properties now represent the average of the liquid fraction. Unlike in the previous approach, the height of the  $T$ -points remains unchanged. This representation is described in (Kevlahan et al., 2015), it is based on Brinkman penalization where a control parameter *i.e.* the porosity

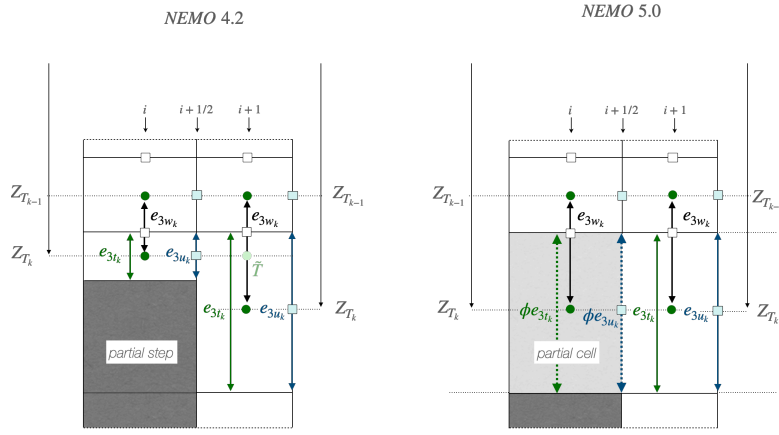


Figure 3.7.: Discretisation of the horizontal difference and average of tracers in the  $zps$ -coordinate. In NEMO v4.2 a linear interpolation is used to estimate  $\tilde{T}$ , the tracer value at the depth of the shallower tracer point of the two adjacent bottom  $T$ -points. While in NEMO v5.0  $zps$ -coordinate takes advantage of partial cells, which are modeled as porous.

modifies fluxes though penalized lateral surfaces. This parameter is encapsulated within vertical scale factors ( $e_{3t^0}$ ,  $e_{3u^0}$ ,  $e_{3v^0}$ ) as illustrated in figure 3.7. In case of ocean cavities, partial cells are also applied at the top interface using the same method as for the bottom interface but upside-down.

### Level bathymetry and mask

The `bottom_level` and `top_level` variables define the bottom and top wet levels in each grid column. The values of `top_level` depend on whether ice shelf cavities are used (subsection 8.1.6): without ice cavities, `top_level` is essentially a land mask (0 on land; 1 everywhere else); with ice cavities, in locations below an overlying ice shelf `top_level` determines the topmost wet point instead.

Based on variables `top_level` and `bottom_level`, the mask variables are defined as follows:

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

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

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

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

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

$$\text{with } wmask(i, j, 1) = tmask(i, j, 1)$$

$$wumask(i, j, k) = umask(i, j, k) * umask(i, j, k - 1)$$

$$\text{with } wumask(i, j, 1) = umask(i, j, 1)$$

$$wvmask(i, j, k) = vmask(i, j, k) * vmask(i, j, k - 1)$$

$$\text{with } wvmask(i, j, 1) = vmask(i, j, 1)$$

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

### 3.2.3. 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 available options to handle closed seas are explained in section 16.2, but it should be noted here that their use requires the appropriate mask fields to be present in the domain configuration file (see section 3.2).

Note that, the user has the option to set the bathymetry in closed seas to zero (see section 16.2) and to optionally decide on the fate of any freshwater imbalance over the area.

```

!-----
&namdom      !  time and space domain
!-----
rn_Dt       = 5400.    !  time step for the dynamics and tracer
rn_atfp     =  0.1    !  asselin time filter parameter
!
ln_cid      = .false.  !  Single column domain (1x1pt)           (T => fill namcid)
!
ln_meshmask = .true.  !  =T create a mesh file
!
ln_shuman   = .false. !  =T shuman averaging active (RK3 only)
/

```

namelist 3.2.: &namdom

### 3.2.4. Output grid files

The model variables describing the spatial domain configuration (latitude/longitude, scale factors, etc) can be written to a NetCDF file by using one of the following namelist parameters:

**ln\_write\_cfg=.true.** - namelist **&namcfg** ([namelist 17.1](#))

Produces a file whose name is set by the **cn\_domcfg\_out** namelist parameter

**ln\_meshmask=.true.** - namelist **&namdom** ([namelist 3.2](#))

Produces a file named **mesh\_mask**

Similar files are produced by both methods, but the **mesh\_mask** file will contain additional variables describing the land-sea mask and depth coordinate that may be useful for post-processing applications. Both files also contain a number of the same variables found in files generated by the **DOMAINcfg** tool (see [appendix F](#)).



## Sea surface height and 2D external mode (D2D)

### Table of contents

4.1. Sea surface height ( <i>sshwzv.F90/stp2d.F90</i> ) . . . . .	39
4.1.1. Explicit free surface ( <i>ln_dynspg_exp, sshwzv.F90</i> ) . . . . .	39
4.1.2. Split-explicit free surface ( <i>ln_dynspg_ts</i> ) . . . . .	39
4.1.3. Split-explicit time-stepping ( <i>dynspg_ts.F90</i> ) . . . . .	40
4.1.4. External forcings . . . . .	41

### Changes record

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

```

-----
&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.
nn_btflt = 1 ! Add dissipation with either boxcar averaging or dissipative Forward-Backward
! ! ! = 0 None
! ! ! = 1 Boxcar over nn_e sub-steps
! ! ! = 2 Boxcar over 2*nn_e " "
! ! ! = 3 Temporal dissipation (Demange 2019)
rn_bt_alpha = 0. ! (if nn_btflt=3) ==> Temporal diffusion parameter (recommended values = 0.07-0.09)
!
ln_bt_auto = .true. ! Number of sub-step defined from:
rn_bt_cmax = 0.8 ! =T : the Maximum Courant Number allowed
nn_e = 30 ! =F : the number of sub-step in rn_Dt seconds
/
    
```

namelist 4.1.: &namdyn\_spg

In the present chapter we describe the equations used to compute the the sea surface height. Recall from [subsection 1.2.2](#) that allowing a free surface permits the existence of External Gravity Waves (EGWs). Resolving these fast moving waves explicitly imposes a severe limit on the model timestep to avoid breaching the CFL condition. Alternatively, EGWs can be filtered by discretisation of the temporal derivatives using a split-explicit scheme. This chapter describes these options controlled via the `&namdyn_spg` ([namelist 4.1](#)) namelist.

## 4.1. Sea surface height ( *sshwzv.F90* / *stp2d.F90* )

The sea surface height is given by the vertical average of the kinematic surface condition ([subsection 1.2.2](#)):

$$\begin{aligned}
 \frac{\partial \eta}{\partial t} &\equiv \frac{1}{e_{1t}e_{2t}} \sum_k \{ \delta_i [e_{2u} e_{3u} u] + \delta_j [e_{1v} e_{3v} v] \} - \frac{emp}{\rho_w} \\
 &\equiv \sum_k \chi e_{3t} - \frac{emp}{\rho_w}
 \end{aligned}
 \tag{4.1}$$

where  $emp$  is the surface freshwater budget (i.e. evaporation minus precipitation + optional contributions (see [subsection 4.1.4](#))), expressed in  $\text{Kg}/\text{m}^2/\text{s}$  (which is equal to  $\text{mm}/\text{s}$ ), and  $\rho_w=1,026 \text{ Kg}/\text{m}^3$  is the reference density of sea water (Boussinesq approximation). The sea-surface height is evaluated using exactly the same time stepping scheme as the tracer [equation 6.21](#). E.g., in the case of MLF this is a leapfrog scheme in combination with an Asselin time filter, *i.e.* the velocity appearing in [equation 4.1](#) 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](#)).

### 4.1.1. Explicit free surface ( `ln_dynspg_exp` )

With MLF time-stepping there is an explicit free surface formulation ( `ln_dynspg_exp=.true.` ), for which the model time step must be chosen to be small enough to resolve the external gravity waves (typically a few tens of seconds). The sea surface height is evaluated from [equation 4.1](#) using a leap-frog scheme (*i.e.* centered in time) with  $ssh$  at the *now* timestep coming from the previous time step computation.

Note that this option is not yet available with RK3 time-stepping but, if implemented,  $ssh$  would be computed at each stage from [equation 4.1](#) using values at  $n$ ,  $n + \frac{1}{3}$  and  $n + \frac{1}{2}$  respectively.

### 4.1.2. Split-explicit free surface ( `ln_dynspg_ts` )

The split-explicit free surface formulation used in *NEMO* ( `ln_dynspg_ts=.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  $\Delta t$ , the time step used for the three dimensional prognostic variables ([figure 4.1](#)). The size of the small time step,  $\Delta t_e$  (the external mode or barotropic time step) is provided through the `nn_e` namelist parameter as:  $\Delta t_e = \Delta t/nn_e$ . This parameter can be optionally defined automatically ( `ln_bt_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,  $\Delta t_e$  is adjusted so that the Maximum allowed Courant number is smaller than `rn_bt_cmax`.

The barotropic mode solves the following equations:

$$\begin{aligned} \frac{\partial \mathbf{U}_h}{\partial t} &= -f \mathbf{k} \times \mathbf{U}_h - g \nabla_h \eta - \frac{c_b^U}{H + \eta} \bar{\mathbf{U}}_h + \bar{\mathbf{G}} \\ \frac{\partial \eta}{\partial t} &= -\nabla \cdot [(H + \eta) \mathbf{U}_h] + P - E \end{aligned} \quad (4.2)$$

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 4.2](#) represents the bottom stress (see [section 11.4](#)), explicitly accounted for at each barotropic iteration.

For 'single 1<sup>st</sup>', RK3 time-stepping, barotropic velocities and sea surface height are calculated for the *after* timestep from the *before* values prior to the first stage of the main RK3 timestepping. This occurs in the new module `stp2d.F90` which computes the RHS forcing terms from *before* fields and then calls the `dyn_spg_ts` routine to perform the time-stepping as detailed below. In the MLF time-stepping case, RHS forcing terms are computed inside `dyn_spg_ts` itself.

For clarity, in RK3 the RHS forcing terms on momentum are vertically averaged 3D trends of:

$$\text{HPG} + \text{LDF} + (\text{COR} + \text{RVO}) + \text{KEG} + \text{ZAD}$$

for the Vector invariant form (for which the 3D fields are also the RHS terms for the 1<sup>st</sup> stage RK3 time stepping) and:

$$\text{HPG} + \text{LDF} + (\text{COR} + \text{MET}) + \text{ADV}$$

for the flux form. In the latter case ADV must be recomputed at the 1<sup>st</sup> stage. The new terms in these summaries are: RVO - Relative Vorticity and MET - Metric term. External forcing is also applied in the form of baroclinic bottom drag and surface winds. Other external forcings may be optionally applied as detailed in [subsection 4.1.4](#).

For ssh, the external forcing is the net column-average freshwater flux. This is evaporation minus precipitation plus optional contributions from other sources as listed in [subsection 4.1.4](#).

### 4.1.3. Split-explicit time-stepping ( *dynspg\_ts.F90* )

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 ( `nn_bt_filt=1` ), the external mode is integrated between *now* and *after* baroclinic time-steps ([figure 4.1a](#)). To avoid aliasing of fast barotropic motions into three dimensional equations, time filtering is eventually applied on barotropic quantities. 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 ( `nn_bt_filt=2` ). Although more computationally expensive ( `nn_e` 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. Patrick Marsaleix' work here. Also work done by SHOM group.

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. 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 ( `nn_bt_filt=3` ). 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 in the POM model for example, 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 would be inevitably unstable. One can however add some dissipation, but in the time domain, by slightly modifying the barotropic time stepping coefficients ([Demange et al. \(2019\)](#)). This is implemented here through an additional parameter ( `rn_bt_alpha` ), which controls the amount of temporal diffusion.

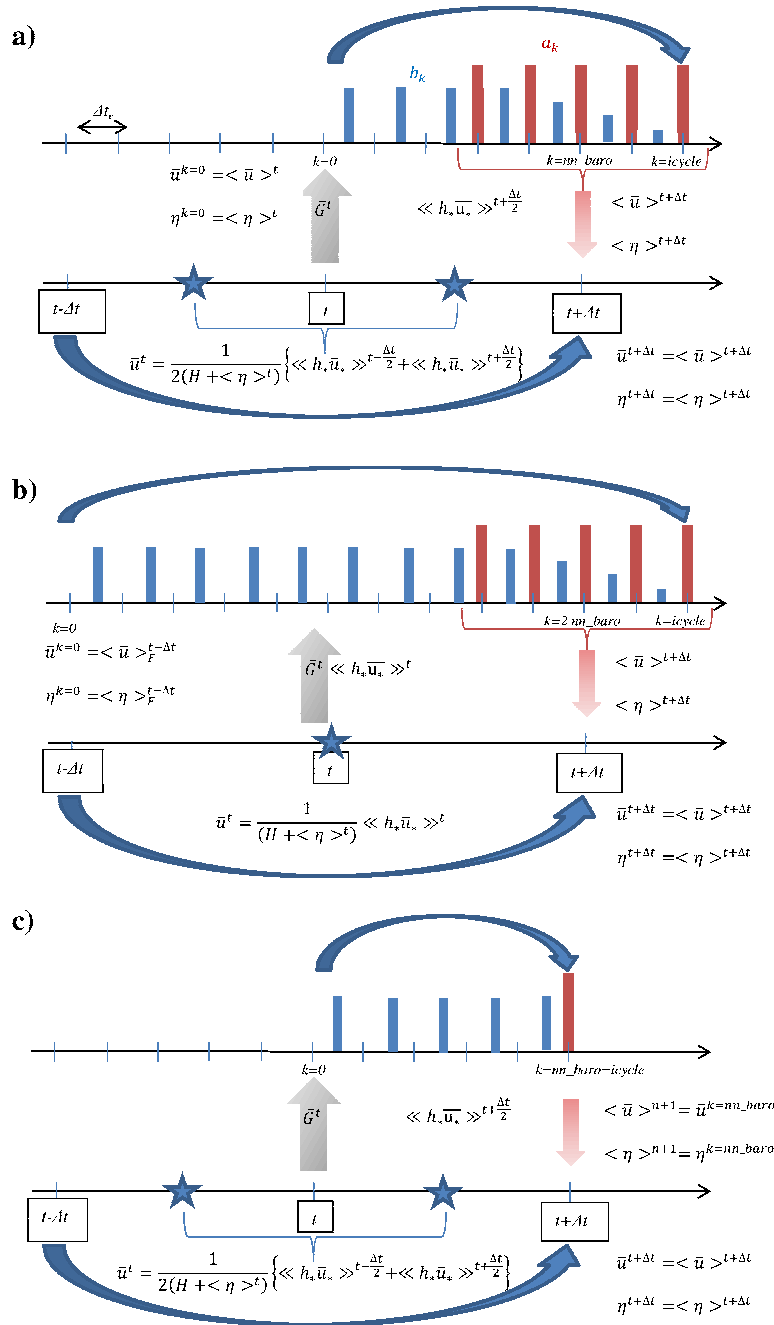


Figure 4.1.: Schematic of the split-explicit time stepping scheme for the external and internal modes with MLF. Time increases to the right. In this particular example, a boxcar averaging window over `nn_e` barotropic time steps is used (`nn_bt_flt=1`) and `nn_e=5`. Internal mode time steps (which are 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: `nn_bt_flt=1`. b) Centred time integration: `nn_bt_flt=2`. c) Forward time integration with no time filtering (POM-like scheme): `nn_bt_flt=3`.

#### 4.1.4. External forcings

The net column-average freshwater flux applied to the ssh equation can also be modified by the following options:

(1) When `ln_rnf=true`. (see [section 7.9](#)), river runoff is taken into account when computing the net freshwater flux.

(2) When `ln_isf=true`. (see [section 8.1](#)), explicit or parameterised contributions from ice-shelf cavities are taken into account when computing the net freshwater flux. Furthermore, if `ln_isfcpl_cons=true`, the corrective increment flux applied to ensure the mass conservation when NEMO is coupled to an ice sheet model is also taken into account (see [subsection 8.1.4](#) for details).

(3) When `ln_sdw=true`. (see [subsection 7.10.3](#)), the contribution from divergence due to Stoke's drift is

taken into account when computing the net freshwater flux.

(4) When `lk_asminc .AND. ln_sshinc .AND. ln_asmiau=.true.` (see [section 14.2](#)), the contribution from IAU weighted ssh increments is taken into account when computing the net freshwater flux.

Besides the surface and bottom baroclinic stresses four other forcings may enter the barotropic momentum equations.

(1) When `ln_apr_dyn=.true.` (see [section 7.7](#)), 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 7.8](#)), the tidal potential is taken into account when computing the surface pressure gradient.

(3) When `nn_ice_embd=2` and SI3 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.

(4) When `ln_wave .AND. ln_bern_srfc=.true.` (see [subsection 7.10.6](#)), a depth-uniform wave-induced kinematic pressure term (the Bernoulli head) is added to the mean pressure.

## Table of contents

5.1.	Continuity equation ( <i>sshwzv.F90, w</i> ) . . . . .	44
5.1.1.	Horizontal divergence ( <i>divhor.F90, <math>\chi</math></i> ) . . . . .	44
5.1.2.	Vertical velocity ( <i>sshwzv.F90</i> ) . . . . .	45
5.2.	Coriolis and advection: vector invariant form . . . . .	45
5.2.1.	Vorticity term ( <i>dynvor.F90</i> ) . . . . .	45
5.2.2.	Kinetic energy gradient term ( <i>dynkeg.F90</i> ) . . . . .	48
5.2.3.	Vertical advection term ( <i>dynzad.F90</i> ) . . . . .	48
5.3.	Coriolis and advection: flux form . . . . .	48
5.3.1.	Coriolis plus curvature metric terms ( <i>dynvor.F90</i> ) . . . . .	48
5.3.2.	Flux form advection term ( <i>dynadv.F90</i> ) . . . . .	49
5.4.	Hydrostatic pressure gradient ( <i>dynhpg.F90</i> ) . . . . .	50
5.4.1.	Full step <i>Z</i> -coordinate ( <i>ln_dynhpg_zco</i> ) . . . . .	50
5.4.2.	Generalised <i>S</i> -coordinates . . . . .	51
5.5.	Surface pressure gradient ( <i>dynspg.F90/stp2d.F90</i> ) . . . . .	52
5.6.	Lateral diffusion term and operators ( <i>dynldf.F90</i> ) . . . . .	52
5.6.1.	Iso-level laplacian ( <i>ln_dynldf_lap</i> ) . . . . .	53
5.6.2.	Rotated laplacian ( <i>ln_dynldf_iso</i> ) . . . . .	53
5.6.3.	Iso-level bilaplacian ( <i>ln_dynldf_bilap</i> ) . . . . .	54
5.7.	Vertical diffusion term ( <i>dynzdf.F90</i> ) . . . . .	54
5.8.	Wetting and drying . . . . .	55
5.8.1.	Directional limiter ( <i>wet_dry.F90</i> ) . . . . .	56
5.8.2.	The WAD test cases ( <i>usrdef_zgr.F90</i> ) . . . . .	57
5.9.	Time evolution term - leapfrog ( <i>dynatf.F90</i> ) . . . . .	57

## 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 7](#), [chapter 10](#) and [chapter 11](#), respectively.

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

The different options available to the user are managed by namelist variables. For term *tnt* in the momentum equations, the logical namelist variables are *ln\_dynntt\_XXX*, where *XXX* is a 3 or 4 letter acronym corresponding to each optional scheme. The corresponding code can be found in the *dynntt\_XXX* module in the DYN directory, and it is usually computed in the *dyn\_tnt\_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 16](#). Furthermore, the tendency terms associated with the 2D barotropic vorticity balance (when *trdvor?* is defined) can be derived from the 3D terms.

## 5.1. Continuity equation ( *sshwzv.F90* )

The evolution of sea surface height ( $\eta$ ) and vertical velocity ( $w$ ) in ocean modeling is fundamentally governed by the continuity equation, which ensures the conservation of volume. Through this equation, we derive expressions for  $\eta$  and  $w$  that are essential for maintaining the balance between horizontal and vertical transport in the ocean's volume-conserving framework. Both are deduced from the horizontal divergence  $\chi$  and require the horizontal divergence calculation. Because  $\eta$  evolution is related to the external mode it is described in [chapter 4](#) while  $w$  evolution is described here after.

### 5.1.1. Horizontal divergence ( *divhor.F90* )

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

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

Besides the velocity three other sources may be added to the horizontal divergence :

- (1) When `ln_rnf=.true.` (see [section 7.9](#)), the divergence caused by river runoff is included.
- (2) When `ln_isf=.true.` (see ??), explicit or parameterised contributions from ice-shelf cavities are taken into account.
- (3) When `lk_asminc .AND. ln_sshinc .AND. ln_asmiau=.true.` (see [section 14.2](#)), the contribution from IAU weighted ssh increments is included.

In a leapfrog time-stepping scheme, the divergence at *now* time step is used to calculate both nonlinear advection and vertical velocity. In an RK3 time-stepping scheme, the divergence at *before* time step is applied during the first stage, while the *now* time step divergence is used for calculating nonlinear advection and vertical velocity in the following stages.



```

!-----
&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_up3 = .false. ! flux form - 3rd order UBS scheme
/

```

namelist 5.1.: &namdyn\_adv

### 5.1.2. Vertical velocity ( *sshwzv.F90* )

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:

$$\begin{cases} w|_{k_b-1/2} = 0 & \text{where } k_b \text{ is the level just above the sea floor} \\ w|_{k+1/2} = w|_{k-1/2} + e_{3t}|_k \chi|_k - \frac{1}{2\Delta t} (e_{3t}^{t+1}|_k - e_{3t}^{t-1}|_k) \end{cases} \quad (5.1)$$

In the case of a linear free surface ( **key\_linssh** ), the time derivative in [equation 5.1](#) 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 ??).

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 5.1](#) (see [subsubsection 3.1.3](#)).

When `ln_zad_Aimp=.true.`, a proportion of the vertical advection can be treated implicitly (see [section 5.7](#)) depending on the Courant number. This option can be useful when the value of the timestep is limited by vertical advection ([Lemarié et al., 2015](#)).

## 5.2. Coriolis and advection: vector invariant form

The vector invariant form of the momentum equation is most commonly used in coarse-resolution ( $1^\circ$ ) applications of the *NEMO* ocean model. For higher resolutions it requires the activation of Hollingsworth correction ( `nn_dynkeg=1` ) following Arakawa (2001) The flux form option (see next section) has been present since version 2. By structuring the equations in vector invariant form, the dynamics are expressed in terms of intrinsic geometric properties like gradients, curls, and divergences. This ensures that the physics remain consistent and interpretable regardless of the underlying curvilinear grid or coordinate system. It highlights key physical terms like the kinetic energy advection and the relative vorticity.

Options are defined through the `&namdyn_adv` ([namelist 5.1](#)) namelist variables Coriolis and momentum advection terms are evaluated either using a leapfrog scheme or a RK3 scheme. In the leapfrog case the velocity appearing in these expressions is centred in time (*now* velocity). In the RK3 case the velocity appearing in these expressions is forward in time (*before* velocity) at stage 1, it is centred in time (*now* velocity) at stage 2 and 3. At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied following [chapter 9](#).

### 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 [subsubsection 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_msk=.true.` ). The vorticity terms are all computed in dedicated routines that can be found in the *dynvor.F90* module.

```

!-----
&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_een = .false. ! energy & enstrophy scheme
!
ln_dynvor_msk = .false. ! vorticity multiplied by fmask (=T)           ==>>> PLEASE DO NOT ACTIVATE
!                               ! (f-point vorticity schemes only)
!
nn_e3f_typ = 0           ! type of e3f (EEN, ENE, ENS, MIX only) =0 e3f = mi(mj(e3t))/4
!                               !                               =1 e3f = mi(mj(e3t))/mi(mj( tmask))
/

```

namelist 5.2.: &namdyn\_vor

### 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:

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

### 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:

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

### 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 5.2) for the relative vorticity term, and of the ENE scheme (equation 5.3) applied to the planetary vorticity term.

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

### 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 (??), the Coriolis parameter is given by  $f = 2\Omega \sin \varphi_f$  and the layer thickness at  $f$ -points is:

$$e_{3f} = \overline{\overline{e_{3t}}}^{i+1/2, j+1/2} \tag{5.4}$$

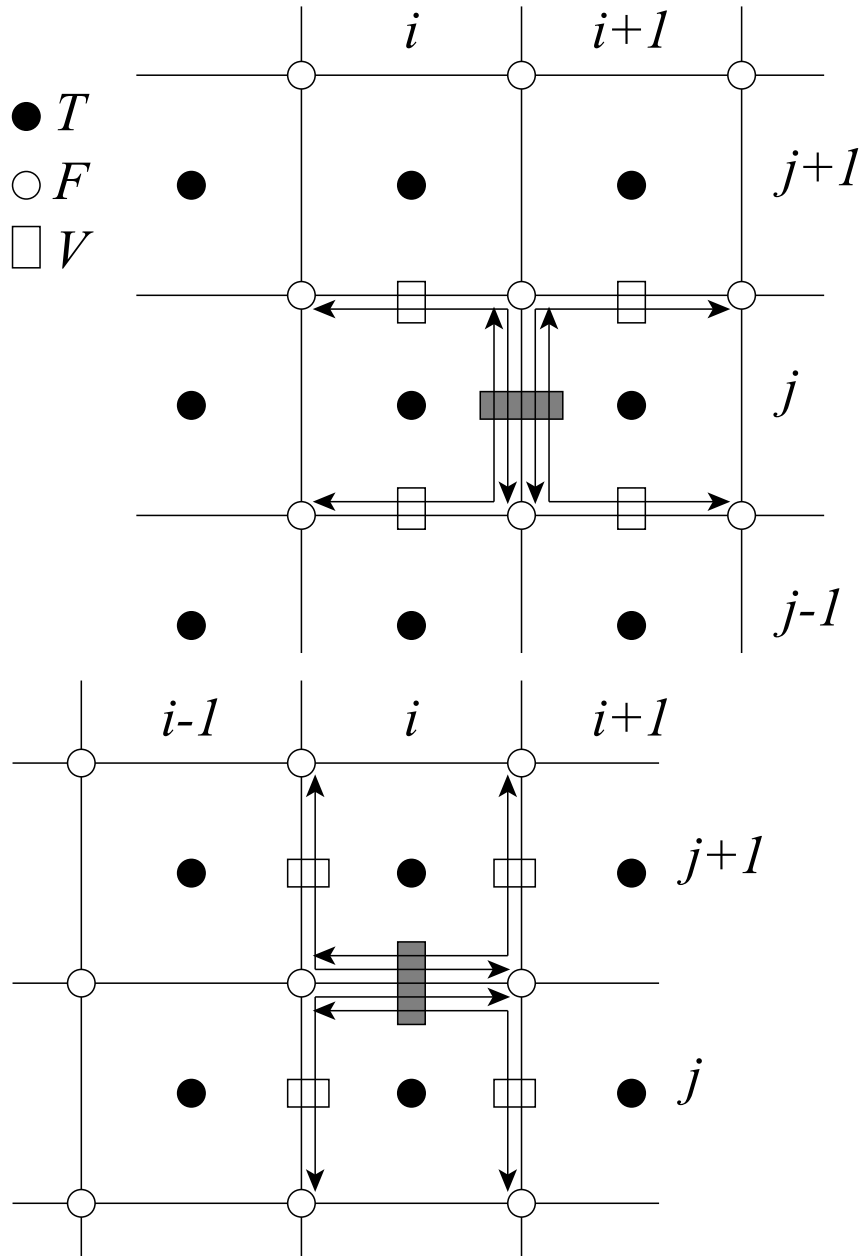


Figure 5.1.: Triads used in the energy and enstrophy conserving scheme (EEN) for  $u$ -component (upper panel) and  $v$ -component (lower panel).

A key point in equation 5.4 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=0`). 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,  ${}^i\mathbb{Q}_{j_p}^{i_p}$  can be defined at a  $T$ -point as the following triad combinations of the neighbouring potential vorticities defined at  $f$ -points (figure 5.1):

$${}^j\mathbb{Q}_{j_p}^{i_p} = \frac{1}{12} \left( q_{j+j_p}^{i-i_p} + q_{j+i_p}^{i+j_p} + q_{j-j_p}^{i+i_p} \right) \quad (5.5)$$

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:

$$\begin{cases} +q e_3 v \equiv +\frac{1}{e_{1u}} \sum_{i_p, k_p} {}^{i+1/2-i_p}\mathbb{Q}_{j_p}^{i_p} (e_{1v} e_{3v} v)_{j+j_p}^{i+1/2-i_p} \\ -q e_3 u \equiv -\frac{1}{e_{2v}} \sum_{i_p, k_p} {}^{i+1/2-j_p}\mathbb{Q}_{j_p}^{i_p} (e_{2u} e_{3u} u)_{j+1/2-j_p}^{i+1/2-j_p} \end{cases} \quad (5.6)$$

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 [subsubsection 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:

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

### 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](#)).

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

## 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 either a leapfrog scheme or a RK3 scheme. In the leapfrog case the velocity appearing in these expressions is centred in time (*now* velocity). In the RK3 case the velocity appearing in these expressions is forward in time (*before* velocity) at stage 1, it is centred in time (*now* velocity) at stage 2 and 3. At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied following [chapter 9](#).

### 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:

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

### Energy conserving scheme ( `ln_dynvor_ent` )

The kinetic energy conserving scheme at T-point (ENT scheme) conserves the global kinetic energy but not the global enstrophy. In this case the altered Coriolis parameter is discretised at  $t$ -points. ENT scheme is given by:

$$\left\{ \begin{array}{l} + \frac{1}{e_{1u} e_{2u} e_{3u}} \overline{(f^T) e_{1t} e_{2t} e_{3t} \bar{v}^j}^{i+1/2} \\ - \frac{1}{e_{1v} e_{2v} e_{3v}} \overline{(f^T) e_{1t} e_{2t} e_{3t} \bar{u}^i}^{j+1/2} \end{array} \right. \quad (5.7)$$

Any of the (equation 5.2), (equation 5.3), (equation 5.7) and (equation 5.6) schemes can be used to compute the product of the Coriolis parameter and the vorticity. However, the energy-conserving schemes (equation 5.6 and equation 5.7) have exclusively been used to date.

This term is evaluated using either a leapfrog scheme or a RK3 scheme. In the leapfrog case it is centred in time (*now* velocity). In the RK3 case it is forward in time (*before* velocity) at stage 1, it is centred in time (*now* velocity) at stage 2 and 3.

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

The discrete expression of the advection term is given by:

$$\left\{ \begin{array}{l} \frac{1}{e_{1u} e_{2u} e_{3u}} \left( \delta_{i+1/2} \left[ \overline{e_{2u} e_{3u} \bar{u}^i} u_t \right] + \delta_j \left[ \overline{e_{1u} e_{3u} \bar{v}^{i+1/2}} u_f \right] \right. \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \left. + \delta_k \left[ \overline{e_{1w} e_{2w} \bar{w}^{i+1/2}} u_{uw} \right] \right) \\ \frac{1}{e_{1v} e_{2v} e_{3v}} \left( \delta_i \left[ \overline{e_{2u} e_{3u} \bar{u}^{j+1/2}} v_f \right] + \delta_{j+1/2} \left[ \overline{e_{1u} e_{3u} \bar{v}^i} v_t \right] \right. \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \left. + \delta_k \left[ \overline{e_{1w} e_{2w} \bar{w}^{j+1/2}} v_{vw} \right] \right) \end{array} \right. \quad (5.8)$$

Two advection schemes are available: a  $2^{nd}$  order centered finite difference scheme, CEN2, or a  $3^{rd}$  order upstream biased scheme, UP3. The latter is described in Shchepetkin and McWilliams (2005). The schemes are selected using the namelist logicals `ln_dynadv_cen2` and `ln_dynadv_up3`. 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:

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

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.

#### UP3: Upstream Biased Scheme ( `ln_dynadv_up3` )

The UP3 advection scheme is an upstream biased third order scheme based on an upstream-biased parabolic interpolation. For example, the evaluation of  $u_T^{up3}$  is done as follows:

$$u_T^{up3} = \bar{u}^i - \frac{1}{6} \begin{cases} u''_{i-1/2} & \text{if } \overline{e_{2u} e_{3u} \bar{u}^i} \geq 0 \\ u''_{i+1/2} & \text{if } \overline{e_{2u} e_{3u} \bar{u}^i} < 0 \end{cases} \quad (5.10)$$

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

```

!-----
&namdyn_hpg      ! Hydrostatic pressure gradient option          (default: NO selection)
!-----
ln_hpg_zco = .false. ! z-coordinate - full steps
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_djc_vnh = .true. ! hor. bc type for djc scheme (T=von Neumann, F=linear extrapolation)
  ln_hpg_djc_vnv = .true. ! vert. bc type for djc scheme (T=von Neumann, F=linear extrapolation)
ln_hpg_prj = .false. ! s-coordinate (Pressure Jacobian scheme)
/

```

namelist 5.3.: &namdyn\_hpg

component, it can be used without explicit lateral diffusion on momentum (*i.e.* `ln_dynldf_OFF=.true.`), and it is recommended to do so.

The UP3 scheme is used in all directions. UP3 is diffusive and is associated with vertical mixing of momentum.

In a leapfrog environment, 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). In an RK3 environment, the first term in (equation 5.10), which corresponds to a second order centred scheme, is evaluated using the *before* velocity at stage 1 and using the *before* velocity (centred in time) at stage 2 and 3, 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 UP3 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_up3.F90` module and obtain a QUICK scheme.

## 5.4. Hydrostatic pressure gradient ( *dynhpg.F90* )

NEMO offers a selection of different algorithms to compute the hydrostatic pressure gradient (HPG) term in the momentum equation. Options are defined through the `&namdyn_hpg` (namelist 5.3) namelist variables.

Since HPGs are computed along geopotential surfaces, a key distinction between the various algorithms is the type of vertical coordinate they target. In particular, NEMO offers a number of options to compute HPGs with generalised *s*-coordinates that may be not aligned with geopotentials.

The hydrostatic pressure gradient term is evaluated either using a leapfrog scheme, *i.e.* the density appearing in its expression is centred in time (*now*  $\rho$ ), or a RK3 scheme *i.e.* the density appearing in its expression is forward in time (*before*  $\rho$ ), it is centred in time (*now*  $\rho$ ) at stage 2 and 3. 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` )

When using standard geopotential coordinates (`ln_zco=.true.`), the hydrostatic pressure can be directly obtained by vertically integrating the hydrostatic equation from the surface to the bottom. However, pressure is large at great depths 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)

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

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 ( $z = 0$ ).

### 5.4.2. Generalised $S$ -coordinates

Pressure gradient formulations with a generalised  $s(x, y, z, t)$  coordinate have been the subject of a vast number of papers (e.g., Song (1998); Shchepetkin and McWilliams (2003)). A number of different pressure gradient options are available in NEMO:

- **Traditional coding** ( `ln_hpg_sco=.true.` , e.g. Madec et al. (1996)):

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

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 ( $e_{3w}$ ). Note that this scheme is not recommended when using steeply inclined computational levels (e.g., terrain-following or hybrid generalised vertical coordinates, i.e., `ln_sco=.true.` ) - see e.g. Shchepetkin and McWilliams (2003). However, it should be the standard choice when using  $z$ -coordinates ( `ln_zco=.true.` or `ln_zps=.true.` ) with the non-linear free surface ( `ln_linssh=.false.` and `key_qco` ), since in this case model levels will follow the barotropic motion of the ocean (Levier et al., 2007).

- **Traditional coding with adaptation for ice shelf cavities** ( `ln_hpg_isf=.true.` ):

In the presence of ice shelves, the traditional coding has been adapted to accommodate the load provided by the ice shelves. This scheme must be used when ice shelf cavities are activated ( `ln_isfcav=.true.` and the inclusion of `key_isf` . All the details on the modification are provided in subsection 8.1.5.

- **Pressure Jacobian scheme** ( `ln_hpg_prj=.true.` ):

this scheme uses a constrained cubic spline to reconstruct the vertical density profile within a water column. This method maintains the monotonicity between the density nodes. The pressure is calculated by analytical integration of the density profile. For the force in the  $i$ -direction, it calculates the difference of the pressures on the  $i + \frac{1}{2}$  and  $i - \frac{1}{2}$  faces of the cell using pressures calculated at the same height. In grid cells just above the bathymetry, this height is higher than the cells' centre. This scheme works well for moderately steep computational levels but produces large velocities in the SEAMOUNT test case when model levels are steeply inclined.

- **Density Jacobian with cubic polynomial scheme** ( `ln_hpg_djc=.true.` , Shchepetkin and McWilliams (2003)):

the ROMS-like, density Jacobian with cubic polynomial method has been debugged and from vn4.2 is available as an option. This scheme is based on section 5 of Shchepetkin and McWilliams (2003) For the force in the  $i$ -direction, it uses constrained cubic splines to re-construct the density along lines of constant  $s$  and constant  $i$  in the  $(i, s)$  plane. It calculates a line integral of  $\rho$  and then integrates vertically to obtain the horizontal pressure gradient. The constrained cubic splines require boundary conditions to be specified at the upper and lower boundaries and at points where model levels encrop the model bathymetry (i.e., with geopotential or hybrid vertical coordinates). The user can choose between von Neumann and linear extrapolation boundary conditions via the `ln_hpg_djc_vnh` and `ln_hpg_djc_vnv` namelist switches, respectively. This scheme can be used with any type of generalised  $s$ -coordinates - i.e.,  $z$  or  $z^*$ , terrain-following or hybrids of these two (e.g., via the vanishing quasi-sigma or multi-envelope methods, see e.g. Shapiro et al. (2013); Bruciaferri et al. (2018); Wise et al. (2021)) - but at the moment can not be used with ice shelf cavities.

Starting from version 4.2, the density field used by `dyn_hpg` is the density anomaly field `rhd` rather than  $1 + rhd$ . The calculation of the source term for the free surface has been adjusted to take this into account. The true in situ density  $\rho = \rho_0(1 + r_0(z) + rhd)$  where  $r_0(z)$  accounts for the variation of density with depth for water with a potential temperature of  $4^\circ\text{C}$  and salinity of  $35.16504\text{g/kg}$  (see (13) and (14) of Roquet et al. (2015b)).



```

!-----
&namdyn_ldf ! lateral diffusion on momentum (default: NO selection)
!-----
!
! Type of the operator :
ln_dynldf_OFF = .false. ! No operator (i.e. no explicit diffusion)
nn_dynldf_typ = 0 ! =0 div-rot (default) ; =1 symmetric
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 5.4.: &namdyn\_ldf

## 5.5. Surface pressure gradient ( *dynspg.F90 / stp2d.F90* )

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, *i.e.* with the inclusion of **key\_linssh**) and the variable volume case (nonlinear free surface, **key\_qco**). 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 (**ln\_dynspg\_exp=.true.** (MLF time-stepping only)). With explicit time-stepping, the surface pressure gradient is evaluated using the leap-frog scheme (*i.e.* centred in time) and is thus simply given by:

$$\begin{cases} -\frac{1}{e_{1u} \rho_o} \delta_{i+1/2} [\rho \eta] \\ -\frac{1}{e_{2v} \rho_o} \delta_{j+1/2} [\rho \eta] \end{cases} \quad (5.14)$$

where values are evaluated at the now timestep (*dynspg\_exp.F90*).

To allow a longer time step for the three-dimensional equations, one can use a split-explicit free surface (**ln\_dynspg\_ts=.true.**). In that case, a quasi-linear form of 2d barotropic equations is substepped with a small time increment. Details of this were provided in chapter 4. Options are defined through the **&namdyn\_spg** (namelist 4.1) namelist variables.

## 5.6. Lateral diffusion term and operators ( *dynldf.F90* )

Options are defined through the **&namdyn\_ldf** (namelist 5.4) 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 10). 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).

At the lateral boundaries either free slip, no slip or partial slip boundary conditions are applied according to the user's choice (see chapter 9).

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

For lateral iso-level diffusion ( `nn_dynldf_typ=0` ), the discrete operator is:

$$\begin{cases} D_u^{lU} = \frac{1}{e_{1u}} \delta_{i+1/2} [A_T^{lm} \chi] - \frac{1}{e_{2u} e_{3u}} \delta_j [A_f^{lm} e_{3f} \zeta] \\ D_v^{lU} = \frac{1}{e_{2v}} \delta_{j+1/2} [A_T^{lm} \chi] + \frac{1}{e_{1v} e_{3v}} \delta_i [A_f^{lm} e_{3f} \zeta] \end{cases} \quad (5.15)$$

As explained in [section B.3](#), 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.

In v5.0 a symmetrical lateral iso-level operator ( `nn_dynldf_typ=1` ) has been introduced :

$$\begin{cases} D_u^{lU} = \frac{1}{e_{1u} e_{2u} e_{3u}} \left( \frac{1}{e_{2u}} \delta_{i+1/2} [e_{2t} e_{2t} e_{3t} A_T^{lm} \epsilon_T] - \frac{1}{e_{1u}} \delta_{j+1/2} [e_{1f} e_{1f} e_{3f} A_F^{lm} \epsilon_F] \right) \\ D_v^{lU} = \frac{1}{e_{1v} e_{2v} e_{3v}} \left( \frac{1}{e_{2v}} \delta_{j+1/2} [e_{2f} e_{2f} e_{3f} A_F^{lm} \epsilon_F] - \frac{1}{e_{1v}} \delta_{i+1/2} [e_{1t} e_{1t} e_{3t} A_T^{lm} \epsilon_T] \right) \end{cases} \quad (5.16)$$

Where  $\epsilon_F$  and  $\epsilon_T$  are respectively the shearing stress component (F-point) and the tension stress component (T-point) defined as :

$$\begin{cases} \epsilon_F = \frac{e_{1f}}{e_{2f}} \delta_{j+1/2} \left[ \frac{u}{e_{1u}} \right] + \frac{e_{2f}}{e_{1f}} \delta_{i+1/2} \left[ \frac{v}{e_{2v}} \right] \\ \epsilon_T = \frac{e_{2t}}{e_{1t}} \delta_i \left[ \frac{u}{e_{2u}} \right] - \frac{e_{1t}}{e_{2t}} \delta_j \left[ \frac{v}{e_{1v}} \right] \end{cases} \quad (5.17)$$

### 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:

$$\begin{aligned}
D_u^{IU} &= \frac{1}{e_{1u} e_{2u} e_{3u}} \\
&\left\{ \delta_{i+1/2} \left[ A_T^{lm} \left( \frac{e_{2t} e_{3t}}{e_{1t}} \delta_i[u] - e_{2t} r_{1t} \overline{\overline{\delta_{k+1/2}[u]}}^{i,k} \right) \right] \right. \\
&\quad + \delta_j \left[ A_f^{lm} \left( \frac{e_{1f} e_{3f}}{e_{2f}} \delta_{j+1/2}[u] - e_{1f} r_{2f} \overline{\overline{\delta_{k+1/2}[u]}}^{j+1/2,k} \right) \right] \\
&\quad + \delta_k \left[ A_{uw}^{lm} \left( -e_{2u} r_{1uw} \overline{\overline{\delta_{i+1/2}[u]}}^{i+1/2,k+1/2} \right. \right. \\
&\quad \quad \left. \left. - e_{1u} r_{2uw} \overline{\overline{\delta_{j+1/2}[u]}}^{j,k+1/2} \right. \right. \\
&\quad \quad \left. \left. + \frac{e_{1u} e_{2u}}{e_{3uw}} (r_{1uw}^2 + r_{2uw}^2) \delta_{k+1/2}[u] \right) \right] \left. \right\}
\end{aligned} \tag{5.18}$$

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

where  $r_1$  and  $r_2$  are the slopes between the surface along which the diffusion operator acts and the surface of computation ( $z$ - or  $s$ -surfaces). The way these slopes are evaluated is given in the lateral physics chapter (chapter 10).

### 5.6.3. Iso-level bilaplacian operator ( `ln_dynldf_bilap` )

The lateral fourth order operator formulation on momentum is obtained by applying equation 5.15 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 9).

## 5.7. Vertical diffusion term ( *dynzdf.F90* )

Options are defined through the `&namzdf` (namelist 11.1) 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. In v5.0 only a backward (or implicit) time differencing scheme can be used for the vertical diffusion term. (see chapter 2).

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:

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

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 11).

```

!-----
&namwad      !   Wetting and Drying (WaD)                               (default: OFF)
!-----
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
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 5.5.: &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,

$$\left( \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_{z=1} = \frac{1}{\rho_o} \begin{pmatrix} \tau_u \\ \tau_v \end{pmatrix} \quad (5.19)$$

where  $(\tau_u, \tau_v)$  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 7](#)).

The turbulent flux of momentum at the bottom of the ocean is specified through a bottom friction parameterisation (see [section 11.4](#))

When activated (`ln_zad_Aimp=.true.`) vertical advection of momentum is done partly implicitly in areas where there is potential to breach the vertical CFL condition (see ??). The following expressions are included in the implicit solvers in *dynzdf.F90* and *trazdf.F90*:

In vector form case :

$$\begin{cases} w_i \delta_k [u] = \frac{1}{e_{1u} e_{2u}} \overline{\overline{e_{1t} e_{2t} w_i}}^{i,k+1/2} \delta_k \left[ \frac{u}{e_{3uw}} \right]^{up1}, \\ w_i \delta_k [v] = \frac{1}{e_{1v} e_{2v}} \overline{\overline{e_{1t} e_{2t} w_i}}^{j,k+1/2} \delta_k \left[ \frac{v}{e_{3vw}} \right]^{up1}. \end{cases} \quad (5.20)$$

In flux form case :

$$\begin{cases} \delta_k [w_i u] = \frac{1}{e_{1u} e_{2u} e_{3u}} \left( \overline{\overline{e_{1t} e_{2t} w_i}}^{i,up1} - \overline{\overline{e_{1t} e_{2t} w_i}}^{i,k+1,up1} \right), \\ \delta_k [w_i v] = \frac{1}{e_{1v} e_{2v} e_{3v}} \left( \overline{\overline{e_{1t} e_{2t} w_i}}^{j,up1} - \overline{\overline{e_{1t} e_{2t} w_i}}^{j,k+1,up1} \right). \end{cases} \quad (5.21)$$

where  $w_i$  is the part of the vertical velocity to be treated implicitly and all other variables are implicit in time (*after*). Note vertical derivatives are done with a 1<sup>st</sup> order upstream scheme which provides stability but is diffusive. It is therefore important, when using this option, to confirm that the active partitioning of the vertical velocity is not frequently required in areas critical for the study being undertaken.

## 5.8. Wetting and drying

There is currently only one choice of limiter for the wetting and drying code (wd): a directional limiter (dl). Previous versions also provided an iterative limiter (il) but this has been removed due to performance and robustness issues. The framework for providing alternatives has been retained in case of future interest so the directional limiter has to be explicitly selected despite being the only choice.

The directional limiter is based on the scheme developed by [Warner et al. \(2013\)](#) for ROMS which was in turn based on ideas developed for POM by [Oey \(2006\)](#). The directional limiter is activated by setting `ln_wd_dl=.true.`

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`).

Wetting and Drying schemes take all points in the domain below a land elevation of `rn_wdld` to be covered by water. The topography specified with a model configuration is required 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 "`ssh_ref`" value may be supplied as the `rn_wd_ref_depth` variable in the domain configuration file. Otherwise it is assumed to be zero. 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 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 and may need their surface fluxes reduced. The depth at which to start tapering is controlled by the user by setting `rn_wd_sbcdep`. The fraction ( $< 1$ ) of surface fluxes to use at this depth is set by `rn_wd_sbcfra`.

The code has been tested in seven 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. The final sub-section covers some additional considerations that are relevant to all possible limiting schemes.

### 5.8.1. 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`.

### 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 using the `rn_wd_sbcdep` and `rn_wd_sbcfra` options discussed above.

### 5.8.2. The WAD test cases ( *usrdef\_zgr.F90* )

See the WAD tests MY\_DOC documentation for details of the WAD test cases.

## 5.9. Time evolution term - leapfrog ( *dynatf.F90* )

Options are defined through the `&namdom` (namelist 3.2) 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 ( `key_qco` ), where it has to be applied to the thickness weighted velocity (see section A.3) The time integration is done within *dynzdf.F90*

- vector invariant form or linear free surface ( `ln_dynadv_vec=.true.` or `key_linssh` ):

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

- flux form and nonlinear free surface ( `ln_dynadv_vec=.false.` and `key_qco` ):

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

where RHS is the right hand side of the momentum equation, the subscript *f* denotes filtered values and  $\gamma$  is the Asselin coefficient.  $\gamma$  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.

## Table of contents

6.1. Tracer advection ( <i>traadv.F90</i> ) . . . . .	59
6.1.1. CEN: Centred scheme ( <i>ln_traadv_cen</i> ) . . . . .	61
6.1.2. FCT: Flux Corrected Transport scheme ( <i>ln_traadv_fct</i> ) . . . . .	61
6.1.3. MUSCL: Monotone Upstream Scheme for Conservative Laws ( <i>ln_traadv_mus</i> ) . . . . .	62
6.1.4. UBS a.k.a. UP3: Upstream-Biased Scheme ( <i>ln_traadv_ubs</i> ) . . . . .	62
6.1.5. QCK: QuiCKest scheme ( <i>ln_traadv_qck</i> ) . . . . .	63
6.2. Tracer lateral diffusion ( <i>traldf.F90</i> ) . . . . .	63
6.2.1. Type of operator ( <i>ln_traldf_{OFF,lap,blp}</i> ) . . . . .	64
6.2.2. Action direction ( <i>ln_traldf_{lev,hor,iso,triad}</i> ) . . . . .	64
6.2.3. Iso-level (bi-)laplacian operator ( <i>ln_traldf_iso</i> ) . . . . .	65
6.2.4. Standard and triad (bi-)laplacian operator . . . . .	65
6.3. Tracer vertical diffusion ( <i>trazdf.F90</i> ) . . . . .	66
6.4. External forcing . . . . .	66
6.4.1. Surface boundary condition ( <i>trasbc.F90</i> ) . . . . .	67
6.4.2. Solar radiation penetration ( <i>traqsr.F90</i> ) . . . . .	68
6.4.3. Bottom boundary condition ( <i>trabbc.F90</i> ) - <i>ln_trabbc</i> ) . . . . .	69
6.5. Bottom boundary layer ( <i>trabbl.F90</i> - <i>ln_trabbl</i> ) . . . . .	71
6.5.1. Diffusive bottom boundary layer ( <i>nn_bbl_ldf=1</i> ) . . . . .	71
6.5.2. Advective bottom boundary layer ( <i>nn_bbl_adv=1,2</i> ) . . . . .	72
6.6. Tracer damping ( <i>tradmp.F90</i> ) . . . . .	73
6.7. Tracer time evolution ( <i>traatf.F90</i> ) . . . . .	73
6.8. Equation of state ( <i>eosbn2.F90</i> ) . . . . .	74
6.8.1. Equation of seawater ( <i>ln_{teos10,eos80,seos}</i> ) . . . . .	74
6.8.2. Brunt-Väisälä frequency . . . . .	75
6.8.3. Freezing point of seawater . . . . .	76
6.9. Wave induced transport . . . . .	76
6.10. Internal wave filtering . . . . .	76

## 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
  nn_fct_imp = 1 ! =1/2, optimized / accurate treatment of implicit part
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 6.1.: &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:

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

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, an internal damping (DMP) and the contribution from the floating ice shelves (ISF) term. The terms QSR, BBC, BBL, DMP and ISF 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, ZDF and ISF modules and described in [chapter 7](#), [chapter 10](#), [chapter 11](#) and [chapter 8](#), respectively. Note that *tranpc.F90*, the non-penetrative convection module, is located in the `./src/OCE/TRA` directory because it directly modifies the tracer fields. However, it is described alongside the model's vertical physics (ZDF), together with other available parameterizations of convection. Similarly, *trample.F90*, which implements the mixed-layer eddy parameterization, and *traisf.F90*, which handles ice-shelf fluxes, are also included in the `./src/OCE/TRA` directory but are addressed in their respective contexts respectively (LDF) and (LIO).

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_mx1=.true.`), as described in [chapter 12](#).

## 6.1. 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:

$$\text{ADV}_\tau = -\frac{1}{b_t} \left( \delta_i [e_{2u} e_{3u} u \tau_u] + \delta_j [e_{1v} e_{3v} v \tau_v] \right) - \frac{1}{e_{3t}} \delta_k [w \tau_w] \quad (6.1)$$

where  $\tau$  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 6.1](#) 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.* `key_linssh` specified instead of `key_qco`). 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  $\tau = 1$

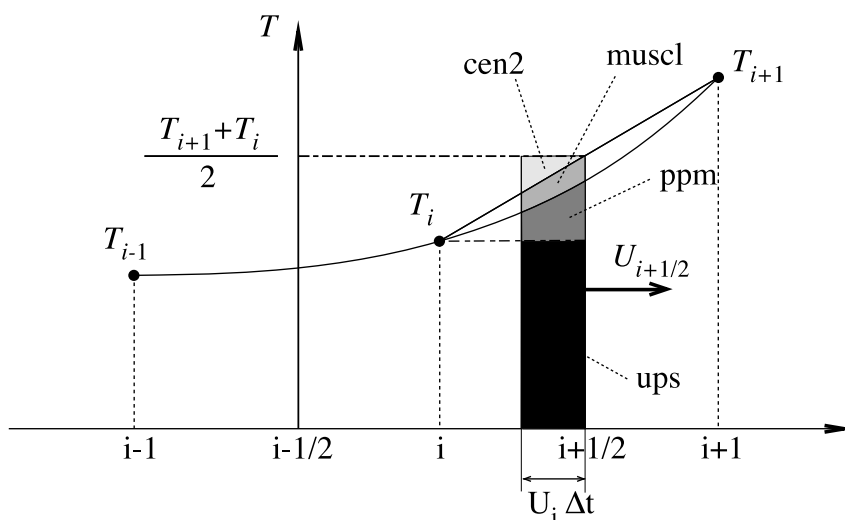


Figure 6.1.: Schematic representation of some ways used to evaluate the tracer value at  $u$ -point and the amount of tracer exchanged between two neighbouring grid points. Upstream biased scheme (ups): the upstream value is used and the black area is exchanged. Piecewise parabolic method (ppm): a parabolic 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.

in (equation 6.1) 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 6.1).

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

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

**non-linear free surface** (`key_qco` specified) 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 (Roullet and Madec, 2000; Griffies et al., 2001; Campin et al., 2004).

The velocity field that appears in (equation 6.1) 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 10).

Several tracer advection scheme are proposed, namely a 2<sup>nd</sup> or 4<sup>th</sup> order **CEN**tréd schemes (CEN), a 2<sup>nd</sup> or 4<sup>th</sup> order **Flux Corrected Transport** scheme (FCT), a **Monotone Upstream Scheme for Conservative Laws** scheme (MUSCL), a 3<sup>rd</sup> **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 6.1) 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.

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.

### 6.1.1. CEN: Centred scheme ( `ln_traadv_cen` )

The **CEN**trred 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 :

$$\tau_u^{cen2} = \overline{T}^{i+1/2} \quad (6.2)$$

CEN2 is non diffusive (*i.e.* it conserves the tracer variance,  $\tau^2$ ) 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. When the model is time-stepped using a leapfrog scheme in conjunction with an Asselin time-filter,  $T$  in (equation 6.2) is the *now* tracer value. In contrast, when the model employs an RK3 scheme,  $T$  in (equation 6.2) refers to the *before* tracer value during the first stage, and the *now* tracer value during the second and third stages.

Note that using the CEN2, the overall tracer advection is of second order accuracy since both (equation 6.1) and (equation 6.2) 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:

$$\tau_u^{cen4} = T - \frac{1}{6} \delta_i \left[ \delta_{i+1/2}[T] \right]^{i+1/2} \quad (6.3)$$

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 6.1 is kept at  $2^{nd}$  order. The expression *4<sup>th</sup> 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, when the model is time-stepped using a leapfrog scheme in conjunction with an Asselin time-filter,  $T$  in (equation 6.3) is the *now* tracer value. In contrast, when the model employs an RK3 scheme,  $T$  in (equation 6.3) refers to the *before* tracer value during the first stage, and the *now* tracer value during the second and third stages.

At a  $T$ -grid cell adjacent to a boundary (coastline, bottom and surface), an additional hypothesis must be made to evaluate  $\tau_u^{cen4}$ . 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.

### 6.1.2. 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.

---

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

In FCT formulation, the tracer at velocity points is evaluated using a combination of an upstream (UP1) and a centred scheme. For example, in the  $i$ -direction :

$$\begin{aligned}\tau_u^{up1} &= \begin{cases} T_{i+1} & \text{if } u_{i+1/2} < 0 \\ T_i & \text{if } u_{i+1/2} \geq 0 \end{cases} \\ \tau_u^{fct} &= \tau_u^{up1} + c_u (\tau_u^{cen} - \tau_u^{up1})\end{aligned}\quad (6.4)$$

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),  $\tau_u^{cen}$  is evaluated in (equation 6.4) using the *now* tracer while  $\tau_u^{up1}$  is evaluated using the *before* tracer.

The FCT procedure is among the most computationally intensive components of the code. Under RK3 time-stepping, it typically runs three times per cycle. A practical optimization is to use a CEN algorithm to estimate tracer advective terms during stages 1 and 2, reserving the full FCT procedure for stage 3.

The FCT involves integrating low-order advection and fluxes, which are then used to compute corrective fluxes. These are obtained by subtracting low-order fluxes from high-order fluxes and applying the nonoscillatory limiter routine. The resulting corrective flux divergence is added to the right-hand side. To mitigate instabilities caused by low-order integration, sub-stepping is applied.

Local implicit vertical advection (`ln_zad_Aimp=.true.`) is required within the FCT scheme due to the low-order integration. The process is optimized when `nn_fct_imp=1` and less efficient when `nn_fct_imp=2`.

### 6.1.3. 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 6.1). 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  $\tau_u^{mus}$ .

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.`).

### 6.1.4. 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\_ubs.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:

$$\tau_u^{ubs} = \overline{T}^{i+1/2} - \frac{1}{6} \begin{cases} \tau''_i & \text{if } u_{i+1/2} \geq 0 \\ \tau''_{i+1} & \text{if } u_{i+1/2} < 0 \end{cases} \quad \text{where } \tau''_i = \delta_i [\delta_{i+1/2}[\tau]] \quad (6.5)$$

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 6.5 (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 6.5 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 6.5 as follows:

$$\tau_u^{ubs} = \tau_u^{cen4} + \frac{1}{12} \begin{cases} +\tau''_i & \text{if } u_{i+1/2} \geq 0 \\ -\tau''_{i+1} & \text{if } u_{i+1/2} < 0 \end{cases} \quad (6.6)$$

or equivalently

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

equation 6.6 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 6.5. 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 6.5.

### 6.1.5. QCK: QuiCKest scheme ( `ln_traadv_qck` )

The Quadratic Upstream Interpolation for Convective Kinematics with Estimated Streaming Terms (QUICK-EST) 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.

## 6.2. Tracer lateral diffusion ( `traldf.F90` )

The lateral diffusion operator for tracers models the physical process of horizontal mixing. *NEMO* provides multiple options that can be combined to construct a lateral diffusion operator suited to the advection scheme, resolution, and coordinate system of the experiment. These options are defined in the `&namtra_ldf` (namelist 6.2) namelist variables. Choices for specifying the lateral diffusion operator includes : (i) the operator type and scale which specifies the form of the operator and thus the spatial scale at which it acts, (ii) the direction of action which defines the orientation of the operator. For rotated operators, additional options must be specified, slope is computed in the `ldfslp.F90` module (see chapter 10) (iii) the eddy diffusivity structure, which control the temporal and spatial distribution of eddy diffusivity (see chapter 10). These combinations allow users to construct a diffusion operator that aligns with the specific requirements of their experiment.

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. It is not true in case of a rotated operator. In this case a vertical component that appears, it 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).



```

!-----
&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) =Ireguier 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 6.2.: &namtra\_ldf

### 6.2.1. 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 6.2.2), and  $A_{ht}$  is the eddy diffusivity coefficient expressed in  $m^2/s$  (see chapter 10).

**`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 10). 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  $\lambda^{-4}$  for disturbances of wavelength  $\lambda$  (so that short waves damped more rapidly than long ones), whereas the laplacian damping time scales only like  $\lambda^{-2}$ .

### 6.2.2. Direction of action ( `ln_traldf_lev` , `ln_traldf_hor` , `ln_traldf_iso` , or `ln_traldf_triad` )

*NEMO* provides three primary approaches for simulating diffusion, each influencing whether a rotation needs to be applied to the diffusion operator. (see figure 6.2)

**`ln_traldf_lev=.true.`** : the direction of action aligns with the vertical coordinate level and there is no need for any rotation of the operator (this is not recommended in *s*-coordinate).

**`ln_traldf_hor=.true.`** : horizontal levels approximates geopotential levels. They do not necessarily align along the vertical coordinate levels. In *z*-coordinate ( `ln_zco=.true.` ) horizontal levels align with *z*-levels. So no rotation is needed. In *s*-coordinate ( `ln_sco=.true.` ) horizontal levels do not align with *s*-levels. Therefore a rotation of the diffusion operator is required.

operator direction	coordinate		
	l_zco	l_zps	l_sco
ln_traldf_lev	traldf_lev.F90 (No rotation)	traldf_lev.F90 (No rotation)	traldf_lev.F90 (No rotation)
ln_traldf_hor	traldf_lev.F90 (No rotation)	traldf_lev.F90 (No rotation)	traldf_iso.F90 (Rotation)
ln_traldf_iso	traldf_iso.F90 (Rotation)	traldf_iso.F90 (Rotation)	traldf_iso.F90 (Rotation)
ln_traldf_triad	traldf_triad.F90	traldf_triad.F90	traldf_triad.F90

Figure 6.2.: Overview of the tracer diffusive operator selected (rotated (iso, triad) or non-rotated (lev)) based on the vertical coordinate and the direction of action specified

**ln\_traldf\_iso=.true. or ln\_traldf\_triad=.true. :** diffusion follows isoneutral layers, which are often sloped relative to both horizontal and models levels. In this case the diffusion operator is always rotated.

The discrete forms of these operators are presented in the following two subsections. They include one iso-level operator ( *traldf\_lev.F90* ), which requires no additional rotation beyond the natural one associated with the model levels, and two rotated operators ( *traldf\_iso.F90* and *traldf\_triad.F90* ).

### 6.2.3. Iso-level (bi-)laplacian operator ( **ln\_traldf\_iso** )

In z-coordinate with or without partial steps ( **ln\_zco=.true.** or **ln\_zps=.true.** ), the iso-level diffusion operator is also a horizontal operator (i.e.acting along geopotential surfaces). Choices **ln\_traldf\_lev=.true.** and **ln\_traldf\_hor=.true.** are thus equivalent. While in s-coordinate ( **ln\_sco=.true.** ), it is simply an iso-level diffusion operator ( **ln\_traldf\_lev=.true.** ). In both cases, it significantly contributes to diapycnal mixing. It is therefore never recommended, even when using it in the bilaplacian case.

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

$$D_t^{lT} = \frac{1}{b_t} \left( \delta_i \left[ A_u^{lT} \frac{e_{2u} e_{3u}}{e_{1u}} \delta_{i+1/2}[T] \right] + \delta_j \left[ A_v^{lT} \frac{e_{1v} e_{3v}}{e_{2v}} \delta_{j+1/2}[T] \right] \right) \quad (6.7)$$

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. The iso-level laplacian operator ( **ln\_traldf\_lap=.true.** ) is implemented in the `tra_ldf_lap` subroutine found in the *traldf\_lev.F90* module. The module also contains `tra_ldf_blp`, to compute the iso-level bilaplacian operator ( **ln\_traldf\_blp=.true.** ).

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.

### 6.2.4. Standard and triad (bi-)laplacian operator

#### Standard rotated (bi-)laplacian operator ( *traldf\_iso.F90* )

This operator is used when **ln\_traldf\_iso=.true.**, regardless of the chosen vertical coordinate, to ensure that diffusion is oriented along isoneutral surfaces. In s-coordinates ( **ln\_sco=.true.** ), it is also applied when a horizontal diffusion direction is specified ( **ln\_traldf\_hor=.true.** ).

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:

$$D_T^{lT} = \frac{1}{b_t} \left[ \begin{aligned} & \delta_i A_u^{lT} \left( \frac{e_{2u} e_{3u}}{e_{1u}} \delta_{i+1/2}[T] - e_{2u} r_{1u} \overline{\overline{\delta_{k+1/2}[T]}}^{i+1/2,k} \right) \\ & + \delta_j A_v^{lT} \left( \frac{e_{1v} e_{3v}}{e_{2v}} \delta_{j+1/2}[T] - e_{1v} r_{2v} \overline{\overline{\delta_{k+1/2}[T]}}^{j+1/2,k} \right) \\ & + \delta_k A_w^{lT} \left( \frac{e_{1w} e_{2w}}{e_{3w}} (r_{1w}^2 + r_{2w}^2) \delta_{k+1/2}[T] \right. \\ & \quad \left. - e_{2w} r_{1w} \overline{\overline{\delta_{i+1/2}[T]}}^{i,k+1/2} - e_{1w} r_{2w} \overline{\overline{\delta_{j+1/2}[T]}}^{j,k+1/2} \right) \end{aligned} \right] \quad (6.8)$$

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). The way these slopes are evaluated is given in section 10.2. At the surface, bottom and lateral boundaries, the turbulent fluxes of heat and salt are set to zero using the mask technique (see section 9.1).

The operator in equation 6.8 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 6.3). 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 10) allows the model to run safely without any additional background horizontal diffusion (Guilyardi et al., 2001).

### 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 6.7) 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 6.8) 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)

## 6.3. Tracer vertical diffusion ( *trazdf.F90* )

Options are defined through the `&namzdf` (namelist 11.1) 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.`) or when differential mixing is activated in the parameterization of internal wave-driven mixing (*i.e.* `ln_tsdiff=.true.`). The way these coefficients are evaluated is given in chapter 11 (ZDF). Furthermore, when iso-neutral mixing is used, both mixing coefficients are increased by  $\frac{e_{1w}e_{2w}}{e_{3w}}(r_{1w}^2 + r_{2w}^2)$  to account for the vertical second derivative of equation 6.8.

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 6.4.1), 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 6.4.3).

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.

Because the vertical mixing is always solved implicitly, the update of the tracer fields described in equation 6.21 is done in *trazdf.F90* module.

## 6.4. External forcing

Changes in the heat and salt content of the ocean's surface layer result from water mass exchanges between the ocean and the atmosphere, land surfaces, icebergs or sea ice (*trasbc.F90*). Runoff related fluxes are distributed vertically. The assimilation module integrates the ocean model with observational data. When activated, it is necessary to correct from the concentration and dilution effects associated with variations in sea surface height. Changes due to water mass exchanges between the ocean and ice-shelves are managed in *traisf.F90*



. Additionally, there is the solar flux, which requires special treatment as it penetrates deeper into the ocean, necessitating the vertical distribution of the associated heat content ( *traqsr.F90* ). *NEMO* can also account for heating caused by geothermal flux ( *trabbl.F90* ) across the seafloor.

### 6.4.1. 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.

Changes in heat content and salt content associated with surface mass fluxes are linked to exchanges with the atmosphere (*emp*) and sea ice (freezing, melting, ridging, etc.). These are exclusively accounted for in the non-solar surface flux ( $Q_{ns}$ ) for heat content and in the surface salt flux (*sfx*) for salt content (see [chapter 7](#) for further details). This help since the forcing formulation is the same for any tracer (including temperature and salinity). When ( `ln_traqsr=.false.` ), the solar flux is not distributed over the surface and is simply added to the non-solar flux.

Changes in heat content and salt content associated with runoff (*rnf*) when `ln_rnf=.true.` are distributed vertically, requiring specific treatment, and are therefore handled separately.

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

$Q_{ns}$  The non-solar part of the net surface heat flux that crosses the sea surface (*i.e.* the difference between the total surface heat flux and the fraction of the short wave flux that penetrates into the water column, see [subsection 6.4.2](#)) 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 icebergs.

*rnf* The mass flux associated with runoff (see [section 7.9](#) for further detail of how it acts on temperature and salinity tendencies)

In an **RK3 environment**, the first two stages focus solely on estimating advection, Coriolis, and pressure gradient terms. The absence of vertical diffusion terms during these initial integrations justifies the exclusion of salt and heat forcing. Indeed, it saves two calls of `traqsr` which is computationally expensive. However, for compatibility with the continuity equation, it is necessary to account for mass flux forcing (*emp*) at the surface. This change of mass at the first level should not impact the salt and temperature, so the first level must be corrected from concentration and dilution effects.

$$F_{kstg=1,2}^T = -\frac{emp}{\rho_o e_{3t}|_{k=1}} T|_{k=1} \quad F_{kstg=1,2}^S = -\frac{emp}{\rho_o e_{3t}|_{k=1}} S|_{k=1} \quad (6.9)$$

At the last stage, the surface boundary condition on temperature and salinity is applied as follows:

$$F_{kstg=3}^T = \frac{1}{C_p} \frac{1}{\rho_o e_{3t}|_{k=1}} Q_{ns} \quad F_{kstg=3}^S = \frac{1}{\rho_o e_{3t}|_{k=1}} sfx \quad (6.10)$$

In the linear free surface case ( `key_linssh` ), an additional terms correct both temperature and salinity at the third stage only. 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. As *emp* includes all the mass fluxes, resulting surface boundary condition is applied as follows:

$$F_{kstg=3}^T = \frac{1}{C_p} \frac{1}{\rho_o e_{3t}|_{k=1}} (Q_{ns} + C_p emp T|_{k=1}) \quad F_{kstg=3}^S = \frac{1}{\rho_o e_{3t}|_{k=1}} (sfx + emp S|_{k=1}) \quad (6.11)$$

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.

In a **leapfrog environment**, the surface boundary condition on temperature and salinity is applied as follows:

$$F^T = \frac{1}{C_p} \frac{1}{\rho_o e_{3t}|_{k=1}} \overline{Q_{ns}}^t \quad F^S = \frac{1}{\rho_o e_{3t}|_{k=1}} \overline{sfx}^t \quad (6.12)$$

```

!-----
&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_5bd = .false. ! 5BD light penetration (IR-Red-Green-Blue-UV)
ln_qsr_bio = .false. ! bio-model light penetration
!
! RGB, 2BD & 5BD 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 : 3D Chl data (=2), Surface Chl data (=1) or Cst value (=0)
rn_si1 = 23.0 ! 2BD : longest depth of extinction
rn_par = 0.47 ! 5BD : fraction of photosynthetically active radiation

cn_dir = './' ! root directory for the chlorophyll 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_chl = 'chlorophyll' , -1. , 'CHLA' , .true. , .true. , 'yearly' , '' ,
!-----
/

```

namelist 6.3.: &amp;namtra\_qsr

where  $\bar{x}^t$  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 ( **key\_linssh** ), 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:

$$F^T = \frac{1}{C_p \rho_o e_{3t}|_{k=1}} \overline{(Q_{ns} - C_p emp T|_{k=1})^t} \quad F^S = \frac{1}{\rho_o e_{3t}|_{k=1}} \overline{(sfx - emp S|_{k=1})^t} \quad (6.13)$$

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](#)).

### 6.4.2. Solar radiation penetration ( *traqsr.F90* )

Options are defined through the **&namtra\_qsr** ([namelist 6.3](#)) 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:

$$\frac{\partial T}{\partial t} = \dots + \frac{1}{\rho_o C_p e_3} \frac{\partial I}{\partial k} \quad (6.14)$$

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

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 6.14](#) is discretized as follows:

$$\frac{1}{\rho_o C_p e_3} \frac{\partial I}{\partial k} \equiv \frac{1}{\rho_o C_p e_{3t}} \delta_k [I_w] \quad (6.15)$$

The shortwave radiation,  $Q_{sr}$ , consists of energy distributed across a wide spectral range. The ocean is strongly absorbing for wavelengths longer than 700 *nm* and these wavelengths contribute to 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,  $\xi_0$ , of a few tens of centimetres (typically  $\xi_0 = 0.35$  *m* set as **rn\_si0** in the **&namtra\_qsr** ([namelist 6.3](#)) 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  $\xi_1$  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 ( $\xi_0, \xi_1, R$ ) 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 6.3). 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 Maritorena (2001)), assuming the same power-law relationship. As shown in figure 6.3, 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.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.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.

A new 5-bands scheme is also available ( `ln_qsr_5bd=.true.` ). It is an extension of the RGB scheme by adding a fifth ultra-violet component (UV) to the 4 other wavebands. As for the RGB scheme, the UV waveband follows the same power-law as described in Morel (1988) and Morel and Maritorena (2001). The chlorophyll-dependent attenuation coefficients are also derived from these articles. The proportion of each band is set up in the namelist by using `rn_abs` to define the near-infrared proportion and `rn_par` to define the visible proportion. The UV proportion is then simply determined by  $1 - rn\_abs - rn\_par$ .

The trend in equation 6.15 associated with the penetration of the solar radiation is added to the temperature trend, and the surface heat flux is modified in routine `traqsr.F90`.

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

### 6.4.3. Bottom boundary condition ( `trabbc.F90` - `ln_trabbc` )

Usually, it is assumed that there is no heat or salt exchange across the ocean floor, *i.e.* a no-flux boundary condition is applied to active tracers on the ocean floor. This is the default option in *NEMO*, which is implemented using the masking technique. Nevertheless, there is a non-zero heat flux from the seabed associated with solid earth cooling. This flux is small compared with surface fluxes (the global mean value is thought to be close to 0.1 W m<sup>-2</sup> (Stein and Stein, 1992)), but it continuously warms the ocean and acts on the densest water masses. Including this flux in a global ocean model can increase the deepest overturning cell (*i.e.* the one associated with the Antarctic Bottom Water) by a few Sverdrups (Emile-Geay and Madec, 2009). It should be recalled that including geothermal heating also implies a persistent heat flux out of the surface ocean (and ultimately through the top-of-atmosphere), even at steady state. The net heat flux out of the ocean induced by geothermal forcing is focused in the Southern Ocean, where Antarctic Bottom Water is ultimately upwelled.

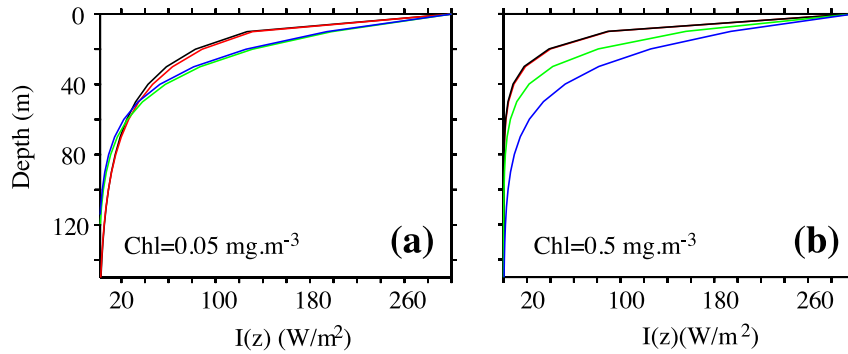


Figure 6.3.: Penetration profile of the downward solar irradiance calculated by four models. Two waveband chlorophyll-independent formulation (blue), a chlorophyll-dependent monochromatic formulation (green), 4 waveband RGB formulation (red), 61 waveband Morel (1988) formulation (black) for a chlorophyll 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).

```

!-----
&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 6.4.: &nambbc

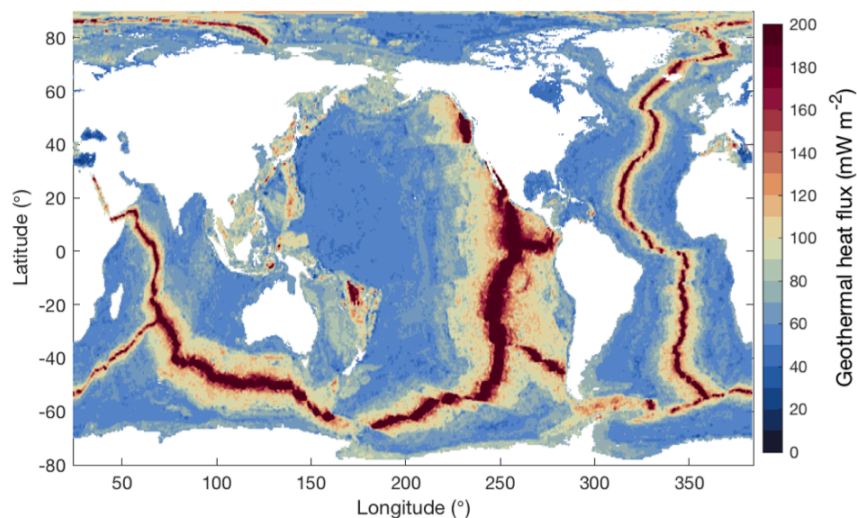


Figure 6.4.: Geothermal heat flux (in  $\text{mW m}^{-2}$ ) estimated by Lucazeau (2019).

```

!-----
&namtbl      !   bottom boundary layer scheme                (default: OFF)
!-----
ln_trabtbl   = .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_gambtbl   = 10.      ! advective bbl coefficient          [s]
/

```

namelist 6.5.: &namtbl

The options for this heat exchange at the bottom of the ocean are defined in section `&namtbl` (namelist 6.4). The presence of geothermal heating is controlled by setting the namelist parameter `ln_trabtbl` to true. When `nn_geoflx` is set to 1, a constant geothermal heating is applied, the value of which is given by the namelist parameter `rn_geoflx_cst`. When `nn_geoflx` is set to 2, a spatially varying geothermal heat flux is applied, provided by a NetCDF file whose name is defined in the namelist section `&namtbl` (namelist 6.4). It is recommended to use the state-of-the-art estimate of geothermal heat fluxes by [Lucazeau \(2019\)](https://doi.org/10.17882/103233), available on NEMO global grids at <https://doi.org/10.17882/103233>. This estimate is illustrated in [figure 6.4](#); the global mean ocean value is  $85 \text{ mW m}^{-2}$  and the globally integrated flux is 31 TW.

## 6.5. Bottom boundary layer ( *trabtbl.F90* - `ln_trabtbl` )

Options are defined through the `&namtbl` (namelist 6.5) 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. 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 \(1999\)](#).

### 6.5.1. Diffusive bottom boundary layer ( `nn_bbl_ldf=1` )

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

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

with  $\nabla_{\sigma}$  the lateral gradient operator taken between bottom cells, and  $A_i^{\sigma}$  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

$$A_i^{\sigma}(i, j, t) = \begin{cases} A_{bbl} & \text{if } \nabla_{\sigma} \rho \cdot \nabla H < 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.16)$$

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 6.16](#) 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 6.5](#)). In practice, this constraint is applied separately in the two horizontal directions, and the density gradient in [equation 6.16](#) is evaluated with the log gradient formulation:

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

where  $\rho$ ,  $\alpha$  and  $\beta$  are functions of  $\bar{T}^{\sigma}$ ,  $\bar{S}^{\sigma}$  and  $\bar{H}^{\sigma}$ , the along bottom mean temperature, salinity and depth, respectively.



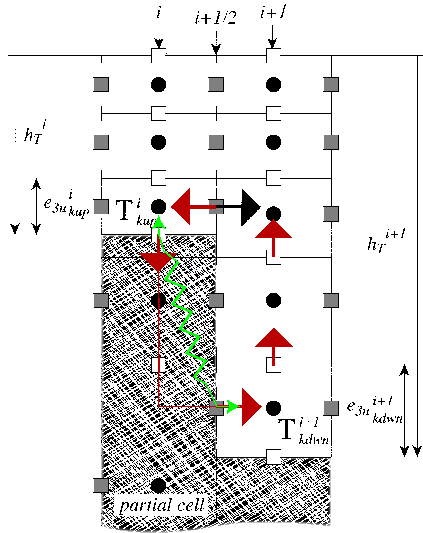


Figure 6.5.: Advective/diffusive Bottom Boundary Layer. The BBL parameterisation is activated when  $\rho_{kup}^i$  is larger than  $\rho_{kdown}^{i+1}$ . Red arrows indicate the additional overturning circulation due to the advective BBL. The transport of the downslope flow is defined either as the 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.

### 6.5.2. 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 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 6.5) (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 Goosse, 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 6.5), is simply given by the following expression:

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

where  $\gamma$ , expressed in seconds, is the coefficient of proportionality provided as `rn_gambbl`, a namelist parameter, and *kup* and *kdown* are the vertical index of the higher and lower cells, respectively. The parameter  $\gamma$  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  $\gamma$  range between 1 and 10 s (Campin and Goosse, 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 6.5 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 6.17, the horizontal equation 6.18 and the upward equation 6.19 return flows as follows:

$$\partial_t T_{kdown}^{do} \equiv \partial_t T_{kdown}^{do} + \frac{u_{bbl}^{tr}}{b_{tkdown}^{do}} (T_{kup}^{sh} - T_{kdown}^{do}) \quad (6.17)$$

$$\partial_t T_{kup}^{sh} \equiv \partial_t T_{kup}^{sh} + \frac{u_{bbl}^{tr}}{b_{tkup}^{sh}} (T_{kup}^{do} - T_{kup}^{sh}) \quad (6.18)$$

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

$$\partial_t T_k^{do} \equiv \partial_t T_k^{do} + \frac{u_{bbl}^{tr}}{b_{tk}^{do}} (T_{k+1}^{do} - T_k^{sh}) \quad (6.19)$$

```

!-----
&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 6.6.: &namtra\_dmp

where  $b_t$  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.

## 6.6. Tracer damping ( *tradmp.F90* )

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

$$\frac{\partial T}{\partial t} = \dots - \gamma(T - T_o) \quad \frac{\partial S}{\partial t} = \dots - \gamma(S - S_o) \quad (6.20)$$

where  $\gamma$  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 in the namelist section `&namtra_dmp` (namelist 6.6) . 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 the namelist section `&namtsd` (namelist 2.2) and that `sn_tem` and `sn_sal` structures are correctly defined (*i.e.* that  $T_o$  and  $S_o$  are provided in input files and read using `fldread.F90`, see subsection 7.2.1). The restoring coefficient  $\gamma$  is a three-dimensional array read in 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 6.20 is used are (a) the specification of the boundary conditions along the artificial walls of a limited-domain basin, and (b) the computation of the velocity field associated with a given  $T$ - $S$  field (e.g. 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 with artificial walls instead of open boundaries. In the vicinity of these walls,  $\gamma$  takes on large values (equivalent to a time scale of a few days), whereas it is zero inside 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, while keeping 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 in the ocean. It also has undesirable effects on the ocean convection. It tends to prevent deep convection and the subsequent formation of deep-waters by over-stabilising the water column.

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

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

## 6.7. Tracer time evolution ( *traatf.F90* )

Options are defined through the `&namdom` (namelist 3.2) 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. subsection 2.2.3):

$$\begin{aligned} (e_{3t}T)^{t+\Delta t} &= (e_{3t}T)_f^{t-\Delta t} + 2\Delta t e_{3t}^t \text{RHS}^t \\ (e_{3t}T)_f^t &= (e_{3t}T)^t + \gamma \left[ (e_{3t}T)_f^{t-\Delta t} - 2(e_{3t}T)^t + (e_{3t}T)^{t+\Delta t} \right] \\ &\quad - \gamma \Delta t \left[ Q^{t+\Delta t/2} - Q^{t-\Delta t/2} \right] \end{aligned} \quad (6.21)$$

where RHS is the right hand side of the temperature equation, the subscript  $f$  denotes filtered values,  $\gamma$  is the Asselin coefficient, and  $S$  is the total forcing applied on  $T$  (*i.e.* fluxes plus content in mass exchanges).  $\gamma$  is

```

!-----
&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)*rho0 = -a0*(1+.5*lambda*dT+mu*Z+nu*dS)*dT+b0*dS
!           !   dT = T-rn_T0 ; dS = S-rn_S0
rn_T0        = 10.              ! reference temperature
rn_S0        = 35.              ! reference salinity
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 6.7.: `&nameos`

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 (`key_linssh`) (see [subsection 6.4.1](#)). Not also that in constant volume case, the time stepping is performed on  $T$ , not on its content,  $e_{3t}T$ .

Vertical mixing is always solved implicitly, with updates to the *next* tracer fields handled in the *trazdf.F90* module. The array swapping and Asselin filtering are performed in the *traatf.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  $T^t = T_f$ .

## 6.8. Equation of state (*eosbn2.F90*)

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

The **Equation Of Seawater** (EOS) is an empirical nonlinear thermodynamic relationship linking seawater density,  $\rho$ , 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 purposed, a simplified EOS (S-EOS) inspired by [Vallis \(2006\)](#) is also available.

In the computer code, a density anomaly,  $d_a = \rho/\rho_o - 1$ , is computed, with  $\rho_o$  a reference density. Called *rho0* in the code,  $\rho_o$  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 6.7](#)) 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:  $\Theta$ ). The pressure in decibars is approximated by the depth in meters. With TEOS10, the



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.
$\lambda_1$	rn_lambda1	$5.9520 \cdot 10^{-2}$	cabbeling coeff. in $T^2$
$\lambda_2$	rn_lambda2	$5.4914 \cdot 10^{-4}$	cabbeling coeff. in $S^2$
$\nu$	rn_nu	$2.4341 \cdot 10^{-3}$	cabbeling coeff. in $T S$
$\mu_1$	rn_mu1	$1.4970 \cdot 10^{-4}$	thermobaric coeff. in T
$\mu_2$	rn_mu2	$1.1090 \cdot 10^{-5}$	thermobaric coeff. in S

Table 6.1.: Standard value of S-EOS coefficients

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  $\Theta$  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:  $^{\circ}\text{C}$ , notation:  $\theta$ ). 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 6.1](#). 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 ( $\mu_1, \mu_2$ ) 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.

## 6.8.2. 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,  $(\theta, S_p)$  for TEOS-80, or  $(T, S)$  for S-EOS, and,  $\alpha$  and  $\beta$  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`.

### 6.8.3. Freezing point of seawater

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

$$T_f(S, p) = (a + b\sqrt{S} + cS) S + dp \quad (6.22)$$

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

equation 6.22 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 6.22 (last term) have been dropped. The freezing point is computed through `eos_fzp`, a FORTRAN function that can be found in `eosbn2.F90`.

## 6.9. Wave induced transport

The Stokes drift is a wave-driven mechanism that results in the net transport of mass and momentum, defined as the difference between the motion of a fluid parcel (Lagrangian velocity) and the flow observed at a fixed point (Eulerian velocity). Due to the asymmetry in the orbital paths of water particles, a net forward motion occurs. Further details are available in subsection 7.10.3.

Incorporating Stokes drift is critical for improving ocean circulation models. Notably, the tracer advection equation is modified to allow Eulerian ocean models to account for unresolved wave effects.

When simulating waves `ln_wave` and activating Stokes drift effect `ln_sdw` three-dimensional Stokes velocity is merely added to the tracers advective transports by `tra_adv_trp`, a FORTRAN function that can be found in `traadv.F90`. Since horizontal velocities are modified, the vertical velocity requires to be recomputed.

The divergence of the wave tracer flux equals the mean tracer advection induced by the three-dimensional Stokes velocity, ensuring the consistence between the continuity equation and tracers evolution equations. Thus the barotropic divergence requires to take the stoke drift divergence into account.

Note that Stoke drift velocity also contributes to the shear computation, the Coriolis term (see subsection 7.10.4). It is also used to compute the vortex force added to the relative vorticity term in the Vector Invariant Formulation of the momentum equations (see subsection 7.10.5).

## 6.10. Internal wave filtering

Internal gravity waves can sometimes disrupt an experiment by making the model unstable, restricting the usable time step, or enhancing undesirable vertical mixing. They can also introduce complex frequencies that complicate result analysis.

The studies by (Brown and Campana, 1978) and Shuman (1971) demonstrate that applying a time-averaging technique to the pressure term can mitigate the impact of these waves, thereby allowing the maximum time step to be reached without causing instabilities linked to internal gravity waves. It is equivalent to time-average the pressure term in the momentum right hand side equations and the time integrated transports. In *NEMO*, for sake of simplicity, we retain the second option. When `ln_shuman=.true.` transports are time-averaged in `tra_adv_trp`, a FORTRAN function that can be found in `traadv.F90`. Since horizontal velocities are modified, the vertical velocity requires to be recomputed.

## Surface Boundary Condition (SBC, SAS, TDE)

### Table of contents

7.1. Surface boundary condition for the ocean . . . . .	80
7.2. Input data generic interface . . . . .	81
7.2.1. Input data specification ( <i>fldread.F90</i> ) . . . . .	81
7.2.2. Interpolation on-the-fly . . . . .	82
7.2.3. Standalone surface boundary condition scheme (SAS) . . . . .	84
7.3. Flux formulation ( <i>sbcflx.F90</i> ) . . . . .	85
7.4. Bulk formulation ( <i>sbcblk.F90</i> ) . . . . .	86
7.4.1. Bulk formulae . . . . .	86
7.4.2. Bulk parametrizations . . . . .	88
7.4.3. Cool-skin and warm-layer parameterizations ( <i>ln_skin_cs</i> & <i>ln_skin_wl</i> ) . . . . .	88
7.4.4. Appropriate use of each bulk parametrization . . . . .	88
7.4.5. Ice-Atmosphere Bulk formulae . . . . .	90
7.4.6. Prescribed near-surface atmospheric state . . . . .	91
7.5. Atmospheric Boundary Layer (ABL) model ( <i>sbcabl.F90</i> ) . . . . .	91
7.5.1. ABL1D pre-processing . . . . .	91
7.5.2. ABL1D namelist . . . . .	92
7.6. Coupled formulation ( <i>sbcctl.F90</i> ) . . . . .	94
7.7. Atmospheric pressure ( <i>sbcapr.F90</i> ) . . . . .	94
7.8. Surface tides (TDE) . . . . .	96
7.8.1. Tidal constituents . . . . .	96
7.8.2. Surface tidal forcing . . . . .	96
7.9. River runoffs ( <i>sbcrrf.F90</i> ) . . . . .	97
7.10. Interactions with waves ( <i>sbcwave.F90</i> ) . . . . .	99
7.10.1. Neutral drag coefficient from wave model ( <i>ln_cdgw</i> ) . . . . .	100
7.10.2. Charnok coefficient from wave model ( <i>ln_charn</i> ) . . . . .	100
7.10.3. 3D Stokes Drift ( <i>ln_sdw</i> ) . . . . .	100
7.10.4. Stokes-Coriolis term ( <i>ln_stcor</i> ) . . . . .	101
7.10.5. Vortex-force term ( <i>ln_vortex_force</i> ) . . . . .	101
7.10.6. Wave-induced pressure term ( <i>ln_bern_srfc</i> ) . . . . .	101
7.10.7. Wave modified stress ( <i>ln_tauoc</i> & <i>ln_taw</i> ) . . . . .	102
7.10.8. Waves impact vertical mixing ( <i>ln_phioc</i> & <i>ln_stshear</i> ) . . . . .	102
7.11. Miscellaneous options . . . . .	102
7.11.1. Diurnal cycle ( <i>sbcscy.F90</i> ) . . . . .	102
7.11.2. Rotation of vector pairs onto the model grid directions . . . . .	103
7.11.3. Surface restoring to observed SST and/or SSS ( <i>sbcssr.F90</i> ) . . . . .	103
7.11.4. Handling of ice-covered area ( <i>sbcice_...</i> ) . . . . .	105
7.11.5. Freshwater budget control ( <i>sbcfwb.F90</i> ) . . . . .	105

## Changes record

Release	Author(s)	Modifications
5.0	<i>A. Moulin, G. Samson and P. Mathiot</i>	<i>Update of the SBC chapter and moving ICB and ISF in there own chapter</i>
4.2	<i>A. Moulin, E. Clementi</i>	<i>Update of section 7.10</i>
4.2	<i>Simon Müller</i>	<i>Update of section 7.8; revision of subsection 7.11.5</i>
4.2	<i>Pierre Mathiot</i>	<i>update of the ice shelf section (2019 developments)</i>
4.0	...	...
3.6	...	...
3.4	...	...
<=3.4	...	...

```

-----
&namcbc      !   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 namcbc_flux )
ln_blk       = .false.   !   Bulk formulation                (T => fill namcbc_blk )
ln_abl       = .false.   !   ABL formulation                (T => fill namcbc_abl )
!             !   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., OCE 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 namcbc_iif )
!             !   =2 or 3 for SI3 and CICE, respectively
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 namcbc_ssr)
nn_fwb       = 0         !   FreshWater Budget: =0 unchecked
!             !   =1 volume set to zero at each time step
!             !   =2 volume adjusted from previous year budget (uniform correction to emp)
!             !   =3 volume adjusted from previous year budget (non-uniform correction - proportional to erp)
!             !   =4 special treatment for ISOMIP+ test case
ln_rnf       = .false.   !   runoffs                            (T => fill namcbc_rnf)
ln_apr_dyn   = .false.   !   Patm gradient added in ocean & ice Eqs. (T => fill namcbc_apr )
ln_wave      = .false.   !   Activate coupling with wave         (T => fill namcbc_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 7.1.: &namcbc

The ocean needs seven fields as surface boundary condition:

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

Five different ways are available to provide these fields to the ocean. They are controlled by namelist `&namcbc` (namelist 7.1) variables:

- a bulk formulation (section 7.4), featuring a selection of six bulk parameterization algorithms,
- an atmospheric boundary layer model (section 7.5) associated with the bulk formulation,
- a flux formulation (section 7.3),
- a coupled or mixed forced/coupled formulation (exchanges with an atmospheric model via the OASIS coupler)(section 7.6),
- 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, abl, 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 7.2.2). 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:

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 7.1.: 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.

- the rotation of vector components supplied relative to an east-north coordinate system onto the local grid directions in the model (subsection 7.11.2),
- the use of a land/sea mask for input fields (subsection 7.2.2),
- the addition of a surface restoring term to observed SST and/or SSS (subsection 7.11.3),
- the modification of fluxes below ice-covered areas (using climatological ice-cover or a sea-ice model) (subsection 7.11.4),
- the addition of river runoffs as surface freshwater fluxes or lateral inflow (section 7.9),
- the addition of iceberg melting as surface freshwater flux and latent heat flux (section 8.2),
- the addition of a freshwater flux adjustment in order to avoid a mean sea-level drift (subsection 7.11.5),
- the transformation of the solar radiation (if provided as daily mean) into an analytical diurnal cycle (subsection 7.11.1),
- the activation of wave effects from an external wave model (section 7.10),
- the light penetration in the ocean (subsection 6.4.2),
- the atmospheric surface pressure gradient effect on ocean and ice dynamics (section 7.7),
- 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 four 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.

## 7.1. Surface boundary condition for the ocean

The surface ocean stress is the stress exerted by the wind and the sea-ice on the ocean. It is applied in `dynzdf.F90` module as a surface boundary condition of the computation of the momentum vertical mixing trend (see equation 5.19 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 boundary condition trend of the first level temperature time evolution equation (see equation 6.12 and equation 6.13 in subsection 6.4.1). 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 6.4.2).

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 icebergs (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 icebergs.

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 7.1), and these averaged fields are used to compute the surface fluxes at the frequency of `nn_fsbc` time-steps.

## 7.2. Input data generic interface

A generic interface has been introduced to manage the way input data (2D or 3D fields, like surface forcing or ocean T and S) are 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 7.2.1), the period it cover is one year, month, week or day, and, if on-the-fly interpolation is used, a file of weights must be supplied (see subsection 7.2.2).

Note that when an input data is archived on a disc which is accessible directly from the workspace where the code is executed, then the 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='./'`.

### 7.2.1. Input data specification ( *fldread.F90* )

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

```
! file name ! frequency (hours) ! variable ! time interp. ! clim ! 'yearly' / ! weights ! rotation ! 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 7.2 provides the resulting file name in all possible cases according to whether it is a climatological file or not, and to the open/close frequency (see below for definition).

	daily or weekLLL	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 7.2.: Naming nomenclature for climatological or interannual input file, as a function of the open/close frequency. The stem name is assumed to be 'fn'. For weekly files, the 'LLL' corresponds to the first three letters of the first day of the week (i.e. 'sun', 'sat', 'fri', 'thu', 'wed', 'tue', 'mon'). The 'YYYY', 'MM' and 'DD' should be replaced by the actual year/month/day, always coded with 4 or 2 digits. Note that (1) in mpp, if the file is split over each subdomain, the suffix '.nc' is replaced by '\_PPPP.nc', where 'PPPP' is the process number coded with 4 digits; (2) when using AGRIF, the prefix '\_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. If you want to change this behaviour, it is possible to prepend the variable name with a '-' or a '+' sign. In the first case, the records will be dated at the beginning of the forcing period, while in the second case, the records will be dated at the end of the forcing period.

**Climatological forcing** : a logical to specify if an 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 7.2.2](#).

Additional remarks:

(1) The time interpolation is a simple linear interpolation between two consecutive records of the input data. The only tricky point is therefore to specify the date at which we need to do the interpolation and the date of the records read in the input files. Following [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.

### 7.2.2. Interpolation on-the-fly

Interpolation on the Fly allows the user to supply input files required for the surface forcing on grids other than the model grid. To do this, he or she must supply, in addition to the source data file(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 `namslc` 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 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).

```

!-----
&namsrc_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_osp = 'sas_grid_U' , 120. , 'uos' , .true. , .true. , 'yearly' , '' ,
sn_vsp = 'sas_grid_V' , 120. , 'vos' , .true. , .true. , 'yearly' , '' ,
sn_tem = 'sas_grid_T' , 120. , 'sosstsst' , .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' , '' ,
!!
!! Following only needed with STATION_ASF compiled with "sea-ice" support: "key_si3" (ice fraction, ice surface temperature
and sea-ice albedo:
sn_ifr = 'NOT USED' , 1. , 'siconc' , .true. , .false. , 'yearly' , '' ,
sn_tic = 'NOT USED' , 1. , 'istl1' , .true. , .false. , 'yearly' , '' ,
sn_ial = 'NOT USED' , 1. , 'fal' , .true. , .false. , 'yearly' , '' ,
/

```

namelist 7.2.: &namsrc\_sas

## 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 /tools/WEIGHTS).

### 7.2.3. 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 7.2) namelist variables. Here are the available options :

- `l_sasread=.true.` : ocean fields are coming from netcdf files
- `l_sasread=.false.` : ocean fields are coming from OASIS

To use SAS, the model has to be compiled with the `ORCA2_SAS_ICE` configuration. In standalone mode ( `l_sasread=.true.` ), the namelist parameters that are set in `cfgs/ORCA2_SAS_ICE/EXPREF/namelist_cfg` can be used directly. In 'coupled mode' ( `l_sasread=.false.` ) with OASIS, the user need to compile two executable (one for OCE and one for SAS). In the namelist of the first one `nn_components` need to be set to 1 and in the `sas` namelist set to 2. However, the coupling of SAS and NEMO via OASIS does not seem to work properly yet. This is under investigation.

In both cases, a few routines in the standard model are overwritten by SAS specific 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, 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.
- `sbcasm.F90` : When `l_sasread=.true.` , this module initialises the input files needed for reading temperature, salinity, velocity arrays at the surface, surface vertical scale factor and fraction of energy absorbed by the first level (optional, `ln_read_frq=.true.` . These parameters need to be supplied in the namelist `namsbc_sas` parameters `sn_ustp` , `sn_vsp` , `sn_tem` , `sn_sal` , `sn_ssh` , `sn_e3t` and `sn_frq` respectively. If For velocities, either 2D velocities ( `ln_3d_uve=.false.` ) or 3D ones ( `ln_3d_uve=.true.` ) can be read from the input files. 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 7.2) 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.

### 7.3. 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 7.3) 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 7.2.1 for a more detailed description of the parameters.

Note that in general, a flux formulation is used in associated with a restoring term to observed SST and/or SSS. See subsection 7.11.3 for its specification.

```

!-----
&namcbc_flg ! surface boundary condition : flux formulation (ln_flg =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 !
! pairing ! filename ! ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
!-----!-----!-----!-----!-----!-----!-----!-----!-----!-----
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 7.3.: &namcbc\_flg

## 7.4. Bulk formulation (*sbcblk.F90*)

If the bulk formulation is selected (`ln_blk=.true.`), the air-sea fluxes associated with surface boundary conditions are estimated by means of the traditional *bulk formulae*. As input, bulk formulae rely on a prescribed near-surface atmosphere state (typically extracted from a weather reanalysis) and the prognostic sea (-ice) surface state averaged over `nn_fsb` time-step(s).

Note: all the NEMO Fortran routines involved in the present section have been initially developed (and are still developed in parallel) in the **AeroBulk** open-source project ([Brodeau et al., 2016](#)).

### 7.4.1. Bulk formulae

In NEMO, the set of equations that relate each component of the surface fluxes to the near-surface atmosphere and sea surface states writes

$$\tau = \rho C_D \mathbf{U}_z U_B \quad (7.1a)$$

$$Q_H = \rho C_H C_P [\theta_z - T_s] U_B \quad (7.1b)$$

$$E = \rho C_E [q_s - q_z] U_B \quad (7.1c)$$

$$Q_L = -L_v E \quad (7.1d)$$

$$Q_{sr} = (1 - a) Q_{sw\downarrow} \quad (7.1e)$$

$$Q_{ir} = \delta(Q_{lw\downarrow} - \sigma T_s^4) \quad (7.1f)$$

with

$$\theta_z \simeq T_z + \gamma z$$

$$q_s \simeq 0.98 q_{sat}(T_s, p_a)$$

from which, the the non-solar heat flux is

$$Q_{ns} = Q_L + Q_H + Q_{ir}$$

where  $\tau$  is the wind stress vector,  $Q_H$  the sensible heat flux,  $E$  the evaporation,  $Q_L$  the latent heat flux, and  $Q_{ir}$  the net longwave flux.  $Q_{sw\downarrow}$  and  $Q_{lw\downarrow}$  are the surface downwelling shortwave and longwave radiative fluxes, respectively. Note: a positive sign for  $\tau$ ,  $Q_H$ ,  $Q_L$ ,  $Q_{sr}$  or  $Q_{ir}$  implies a gain of the relevant quantity for the ocean, while a positive  $E$  implies a freshwater loss for the ocean.  $\rho$  is the density of air.  $C_D$ ,  $C_H$  and  $C_E$  are the bulk transfer coefficients for momentum, sensible heat, and moisture, respectively.  $C_P$  is the heat capacity of moist air, and  $L_v$  is the latent heat of vaporization of water.  $\theta_z$ ,  $T_z$  and  $q_z$  are the potential temperature, absolute temperature, and specific humidity of air at height  $z$  above the sea surface, respectively.  $\gamma z$  is a temperature correction term which accounts for the adiabatic lapse rate and approximates the potential temperature at height  $z$  ([Josey et al., 2013](#)).  $\mathbf{U}_z$  is the wind speed vector at height  $z$  above the sea surface (possibly referenced to the surface current  $\mathbf{u}_0$ ). The bulk scalar wind speed, namely  $U_B$ , is the scalar wind speed,  $|\mathbf{U}_z|$ , with the potential inclusion of a gustiness contribution.  $a$  and  $\delta$  are the albedo and emissivity of the sea surface, respectively.

```

!-----
&namcbc_blk ! namcbc_blk generic Bulk formula (ln_blk =T)
!-----
!
! bulk algorithm :
ln_NCAR = .true. ! "NCAR" algorithm (Large and Yeager 2008)
ln_COARE_3p0 = .false. ! "COARE 3.0" algorithm (Fairall et al. 2003)
ln_COARE_3p6 = .false. ! "COARE 3.6" algorithm (Edson et al. 2013)
ln_ECMWF = .false. ! "ECMWF" algorithm (IFS cycle 45r1)
ln_MFS = .false. ! "MFS" algorithm (MFS/BS Copernicus, Petenuzzo et al 2010)
ln_ANDREAS = .false. ! "ANDREAS" algorithm (Andreas et al. 2015)
  rn_zqt = 10. ! Air temperature & humidity reference height (m)
  rn_zu = 10. ! Wind vector reference height (m)
  nn_iter_algo = 5 ! Number of iterations in bulk param. algo ("stable ABL + weak wind" requires more)
  ln_skin_cs = .false. ! use the cool-skin parameterization => use at least nn_iter_algo > 10
  ln_skin_wl = .false. ! use the warm-layer parameterization => use at least nn_iter_algo > 10
!
rn_pfac = 1. ! multipl. factor for precipitation (total & snow)
rn_efac = 1. ! multipl. factor for evaporation (0. or 1.)
!
ln_crt_fbk = .false. ! Add surface current feedback to the wind stress (Renault et al. 2020, doi: 10.1029/2019MS001715)
  rn_stau_a = -2.9e-3 ! Alpha from eq. 10: Stau = Alpha * Wnd + Beta
  rn_stau_b = 8.0e-3 ! Beta
!
ln_humi_sph = .true. ! humidity "sn_humi" is specific humidity [kg/kg]
ln_humi_dpt = .false. ! humidity "sn_humi" is dew-point temperature [K]
ln_humi_rlh = .false. ! humidity "sn_humi" is relative humidity [%]
ln_tair_pot = .false. ! air temperature read in "sn_tair" is already POTENTIAL TEMPERATURE, NOT ABSOLUTE (ECMWF =>
-> ln_tair_pot=.false.)
ln_prec_met = .false. ! precipitation read in "sn_prec" is in [m]
!!
!! Bulk transfer coefficients over sea-ice: (relevant IF: nn_ice >=1 )
ln_Cx_ice_cst = .true. ! use constant ice-air bulk transfer coefficients (value given below)
  rn_Cd_ia = 1.4e-3 ! sea-ice drag coefficient
  rn_Ce_ia = 1.4e-3 ! " sublimation coefficient
  rn_Ch_ia = 1.4e-3 ! " sensible heat flux coefficient
ln_Cx_ice_frm = .false. ! use form drag param from Tsamadoes et al. 2014
  nn_frm = 2 ! = 1 : affects momentum and heat transfer coefficient (ocean-ice and atmos-ice)
  ! = 2 : affects only momentum transfer coefficient (ocean-ice and atmos-ice) (default)
  ! = 3 : affect momentum and heat transfer coefficient (atmos-ice), and only momentum
-> transfer coefficient (ocean-ice)
  rn-Cs_io = 0.002 ! ice-ocn skin drag [0.0005,0.005]
  rn-Cs_ia = 0.0005 ! ice-air skin drag [0.0001,0.001]
  rn-Cr_ia = 0.2 ! ridge/keel drag coefficient [0,1]
  rn-Cr_io = 0.2 ! ridge/keel drag coefficient [0,1]
  rn-Cf_ia = 0.2 ! floe edge atm [0,1]
  rn-Cf_io = 0.2 ! floe edge ocean [0,1]
ln_Cx_ice_AN05 = .false. ! (Andreas et al. 2005)
ln_Cx_ice_LU12 = .false. ! (Lupkes et al. 2012)
ln_Cx_ice_LG15 = .false. ! (Lupkes & Gryanik 2015)
!
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' , '' ,
sn_wndj = 'v_10.15JUNE2009_fill' , 6. , 'V_10_MOD' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bicubic_noc.nc' , 'Vwnd' , '' ,
sn_qsr = 'ncar_rad.15JUNE2009_fill' , 24. , 'SWDN_MOD' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , '' , '' ,
sn_qlw = 'ncar_rad.15JUNE2009_fill' , 24. , 'LWDN_MOD' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , '' , '' ,
sn_tair = 't_10.15JUNE2009_fill' , 6. , 'T_10_MOD' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , '' , '' ,
sn_humi = 'q_10.15JUNE2009_fill' , 6. , 'Q_10_MOD' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , '' , '' ,
sn_prec = 'ncar_precip.15JUNE2009_fill' , -1. , 'PRC_MOD1' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , '' , '' ,
sn_snow = 'ncar_precip.15JUNE2009_fill' , -1. , 'SNOW' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , '' , '' ,
sn_slp = 'slp.15JUNE2009_fill' , 6. , 'SLP' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , '' , '' ,
sn_uoatm = 'NOT USED' , 6. , 'UOATM' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , 'Uoceatm' , '' ,
sn_voatm = 'NOT USED' , 6. , 'VOATM' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , 'Voceatm' , '' ,
sn_cc = 'NOT USED' , 24. , 'CC' , .false. , .true. , 'yearly' ,
-> 'weights_core_orca2_bilinear_noc.nc' , '' , '' ,
sn_hpgi = 'NOT USED' , 24. , 'uhpg' , .false. , .false. , 'monthly' ,
-> 'weights_ERAI3D_F128_2_ORCA2_bicubic' , 'UG' , '' ,
sn_hpgj = 'NOT USED' , 24. , 'vhpg' , .false. , .false. , 'monthly' ,
-> 'weights_ERAI3D_F128_2_ORCA2_bicubic' , 'VG' , '' ,
/

```

namelist 7.4.: &namcbc\_blk



$T_s$  is the sea surface temperature.  $q_s$  is the saturation specific humidity of air at temperature  $T_s$ ; it includes a 2% reduction to account for the presence of salt in seawater (Sverdrup et al., 1942; Kraus and Businger, 1996). Depending on the bulk parametrization used,  $T_s$  can either be the temperature at the air-sea interface (skin temperature, hereafter SSST) or at typically a few tens of centimeters below the surface (bulk sea surface temperature, hereafter SST). The SSST differs from the SST due to the contributions of two effects of opposite sign, the *cool skin* and *warm layer* (hereafter CS and WL, respectively, see subsection 7.4.3). Technically, when the ECMWF or COARE\* or MFS bulk parametrizations are selected (`ln_ECMWF=.true.` or `ln_COARE*=.true.` or `ln_MFS=.true.`),  $T_s$  is the SSST, as opposed to the NCAR bulk parametrization (`ln_NCAR=.true.`) for which  $T_s$  is the bulk SST (*i.e.* temperature at first T-point level).

For more details on all these aspects the reader is invited to refer to Brodeau et al. (2016).

### 7.4.2. Bulk parametrizations

Accuracy of the estimate of surface turbulent fluxes by means of bulk formulae strongly relies on that of the bulk transfer coefficients:  $C_D$ ,  $C_H$  and  $C_E$ . They are estimated with what we refer to as a *bulk parametrization* algorithm. When relevant, these algorithms also perform the height adjustment of humidity and temperature to the wind reference measurement height (from `rn_zqt` to `rn_zu`).

For the open ocean, six bulk parametrization algorithms are available in NEMO:

- NCAR, formerly known as CORE, (Large and Yeager, 2004, 2009)
- COARE 3.0 (Fairall et al., 2003)
- COARE 3.6 (Edson et al., 2013)
- ECMWF (IFS documentation, cy45)
- ANDREAS (Andreas et al., 2015)
- MFS (Castellari et al., 1998)

With respect to version 3, the principal advances in version 3.6 of the COARE bulk parametrization are built around improvements in the representation of the effects of waves on fluxes (Edson et al., 2013; Brodeau et al., 2016). This includes improved relationships of surface roughness, and whitecap fraction on wave parameters. It is therefore recommended to chose version 3.6 over 3.

### 7.4.3. Cool-skin and warm-layer parameterizations ( `ln_skin_cs` & `ln_skin_wl` )

As opposed to the NCAR bulk parametrization, more advanced bulk parametrizations such as COARE3.x and ECMWF are meant to be used with the skin temperature  $T_s$  rather than the bulk SST (which, in NEMO is the temperature at the first T-point level, see subsection 7.4.1).

As such, the relevant cool-skin and warm-layer parametrization must be activated through `ln_skin_cs=T` and `ln_skin_wl=T` to use COARE3.x or ECMWF in a consistent way.

#LB: ADD BLBLA ABOUT THE TWO CS/WL PARAMETRIZATIONS (ECMWF and COARE) !!!

For the cool-skin scheme parametrization COARE and ECMWF algorithms share the same basis: Fairall et al. (1996). With some minor updates based on Zeng and Beljaars (2005) for ECMWF 3.6.

For the warm-layer scheme, ECMWF is based on Zeng and Beljaars (2005) with a recent update from Takaya et al. (2010) (consideration of the turbulence input from Langmuir circulation).

Importantly, COARE warm-layer scheme includes a prognostic equation for the thickness of the warm-layer, while it is considered as constant in the ECWMF algorithm.

### 7.4.4. Appropriate use of each bulk parametrization

#### NCAR

NCAR bulk parametrizations (formerly known as CORE) is meant to be used with the CORE II atmospheric forcing (Large and Yeager, 2009). The expected sea surface temperature is the bulk SST. Hence the following namelist parameters must be set:

```

...
ln_NCAR   = .true.
...
rn_zqt   = 10.    ! Air temperature & humidity reference height (m)
rn_zu    = 10.    ! Wind vector reference height (m)
...
ln_skin_cs = .false. ! use the cool-skin parameterization

```



```

ln_skin_wl = .false. ! use the warm-layer parameterization
...
ln_humi_sph = .true. ! humidity "sn_humi" is specific humidity [kg/kg]

```

## ECMWF

With an atmospheric forcing based on a reanalysis of the ECMWF, such as the Drakkar Forcing Set (Brodeau et al., 2010), we strongly recommend to use the ECMWF bulk parametrizations with the cool-skin and warm-layer parametrizations activated. In ECMWF reanalyzes, since air temperature and humidity are provided at the 2 m height, and given that the humidity is distributed as the dew-point temperature, the namelist must be tuned as follows:

```

...
ln_ECMWF = .true.
...
rn_zqt = 2. ! Air temperature & humidity reference height (m)
rn_zu = 10. ! Wind vector reference height (m)
...
ln_skin_cs = .true. ! use the cool-skin parameterization
ln_skin_wl = .true. ! use the warm-layer parameterization
...
ln_humi_dpt = .true. ! humidity "sn_humi" is dew-point temperature [K]
...

```

Note: when `ln_ECMWF` is selected, the selection of `ln_skin_cs` and `ln_skin_wl` implicitly triggers the use of the ECMWF cool-skin and warm-layer parametrizations, respectively (found in `sbcblk_skin_ecmwf.F90`).

## COARE 3.x

Since the ECMWF parametrization is largely based on the COARE\* parametrization, the two algorithms are very similar in terms of structure and closure approach. As such, the namelist tuning for COARE 3.x is identical to that of ECMWF:

```

...
ln_COARE_3p6 = .true.
...
ln_skin_cs = .true. ! use the cool-skin parameterization
ln_skin_wl = .true. ! use the warm-layer parameterization
...

```

Note: when `ln_COARE_3p0=T` is selected, the selection of `ln_skin_cs` and `ln_skin_wl` implicitly triggers the use of the COARE cool-skin and warm-layer parametrizations, respectively (found in `sbcblk_skin_coare.F90`).

## MFS

The MFS bulk formulae have been developed by Castellari et al. (1998). They have been designed to handle ECMWF operational data for the Mediterranean (Oddo et al., 2009) and Black Sea (Ciliberti et al., 2022) Monitoring Forecasting Centre. The bulk transfer coefficients  $C_H$  and  $C_E$  are computing using an empiric formula by Kondo (1975) whereas the drag coefficient  $C_D$  is computed according to Hellerman and Rosenstein (1983). In this bulk parametrization the net solar radiation depends on the cloud cover and is computed by means of an astronomical formula (Reed, 1977). Albedo monthly values are from Payne (1972) as means of the values at 40°N and 30°N for the Atlantic Ocean (hence the same latitudinal band of the Mediterranean Sea). The net long-wave radiation flux are computed as a function of 2m air temperature, sea-surface temperature, cloud cover and 2m dew point humidity (Bignami et al., 1995).

This parametrization required as input fields the total cloud cover in %.

The following namelist parameters must be set:

```

...
ln_MFS = .true.
...
rn_zqt = 2. ! Air temperature & humidity reference height (m)
rn_zu = 10. ! Wind vector reference height (m)
...
ln_skin_cs = .false. ! use the cool-skin parameterization
ln_skin_wl = .false. ! use the warm-layer parameterization
...
ln_humi_dpt = .true. ! humidity "sn_humi" is dew point Temperature [kg/kg]

```

### 7.4.5. Ice-Atmosphere Bulk formulae

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

- Constant value (`ln_Cx_ice_cst=.true.`): Constant values are used for momentum (`rn_Cd_ia`), sublimation (`rn_Cd_ia`) and sensible heat (`rn_Ch_ia`) transfer coefficients. Default value for all coefficients is set to  $1.4e-3$ .
- Lüpkes et al. (2012) (`ln_Cx_ice_LU12=.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  $1.5e-3$  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_Cx_ice_LG15=.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.
- Tsamados (2014) (`ln_Cx_ice_frm=.true.`): Sea ice contains pressure ridges as well as floe and melt pond edges that act as discrete obstructions to the flow of air or water past the ice, and are a source of form drag. Here, the neutral drag coefficients are estimated from sea ice properties such as ice concentration, vertical extent and area of the ridges, freeboard and floe draft, and size of floes and melt ponds. The new parameterization allows the drag coefficients to be coupled to the sea ice state and therefore to evolve spatially and temporally. For default settings, only the transfer coefficient for the turbulent momentum fluxes are affected. The namelist allows 3 different options:

`nn_frm=1` : affects momentum and heat transfer coefficient (ocean-ice and atm-ice)

`nn_frm=2` : affects only momentum transfer coefficient (ocean-ice and atm-ice) (default)

`nn_frm=3` : affects momentum and heat transfer coefficient (atm-ice), and only momentum transfer coefficient (ocean-ice)

The total drag coefficients are derived as follows:

$$zdrag_{ia} = zdrag_{ia\_skin} + zdrag_{ia\_floe} + zdrag_{ia\_rdg} + zdrag_{ia\_pond}$$

$$zdrag_{io} = zdrag_{io\_skin} + zdrag_{io\_floe} + zdrag_{io\_keel}$$

`zdrag_ia` : total drag coefficient for momentum exchange between ice and atmosphere

`zdrag_ia_skin` : skin drag coefficient (top of the ice)

`zdrag_ia_floe` : floe edge drag coefficient (top of the ice)

`zdrag_ia_rdg` : sail drag coefficient

`zdrag_ia_pond` : pond edge drag coefficient

`zdrag_io` : total drag for momentum exchange between ice and ocean

`zdrag_io_skin` : skin drag coefficient (under the ice)

`zdrag_io_floe` : floe edge drag coefficient (under the ice)

`zdrag_io_rdg` : keel drag coefficient

The area and volume of ridged ice (required input parameters) are derived from mean ice thickness and concentration based on a polynomial fit (?). In the namelist, the skin drag coefficients and factors can be modified for the strength of each contribution:

`rn-Cs_io=0.002` : ice-ocean skin drag [0.0005,0.005]

`rn-Cs_ia=0.0005` : ice-air skin drag [0.0001,0.001]

`rn-Cr_ia=0.2` : factor for ridge/keel drag coefficient [0,1]

`rn-Cr_io=0.2` : factor for ridge/keel drag coefficient [0,1]

`rn-Cf_ia=0.2` : factor for floe edge atm [0,1]

`rn-Cf_io=0.2` : factor floe edge ocean [0,1]

### 7.4.6. Prescribed near-surface atmospheric state

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.

Common options are defined through the `&namsbc_blk` (namelist 7.4) namelist variables. The required 9 input fields are:

Variable description	Model variable	Units	point
i-component of the 10m air velocity	wndi	$m.s^{-1}$	T
j-component of the 10m air velocity	wndj	$m.s^{-1}$	T
10m absolute air temperature	tair	$K$	T
Potential air temperature		$K$	T
Specific humidity	humi	—	T
Relative humidity		%	T
Dew-point temperature		$K$	T
Downwelling longwave radiation	qlw	$W.m^{-2}$	T
Downwelling shortwave radiation	qsr	$W.m^{-2}$	T
Total precipitation (liquid + solid)	precip	$Kg.m^{-2}.s^{-1}$	T
		$m$	T
Solid precipitation	snow	$Kg.m^{-2}.s^{-1}$	T
Mean sea-level pressure	slp	$Pa$	T

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

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 `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 parametrization, information about the input data required by the model is provided in the `namsbc_blk` namelist (see subsection 7.2.1).

#### Air humidity, temperature, precipitation parameters and units:

Air humidity can be provided as one of three parameters:

specific humidity [ $kg/kg$ ], using `ln_humi_sph=.true.` ; relative humidity [%], using `ln_humi_rlh=.true.` ; or dew-point temperature [ $K$ ], using `ln_humi_dpt=.true.` .

Air temperature can be provided as potential temperature (`ln_tair_pot=.true.` ) or as absolute temperature (`ln_tair_pot=.false.` ).

Total precipitation can be provided in meters (`ln_prec_met=.true.` ) or, by default, in  $kg \cdot m^{-2} \cdot s^{-1}$ .

## 7.5. Atmospheric Boundary Layer (ABL) model ( *sbcabl.F90* )

An atmospheric boundary layer (ABL) model is available as an alternative choice to the prescribed near-surface atmospheric forcings using `ln_abl=.true.` . It computes the wind, air potential temperature and specific humidity evolutions in the lower atmosphere following a single-column approach on the same horizontal grid as the ocean component. It represents the adjustment of the air column between the large-scale atmospheric forcing and the surface boundary conditions over both ocean and sea-ice through vertical turbulent mixing. This 1D implementation of the ABL model (ABL1D) and its validation are described in details in Lemarié et al. (2021).

### 7.5.1. ABL1D pre-processing

To use it, an atmospheric vertical grid and specific atmospheric forcing files must be provided to ABL1D. This is because the model expects atmospheric data on its vertical grid and not only near the surface as usually

```

:
:-----
: Atmospheric Boundary Layer preprocessing tool
:-----
:
&nml_dom
  jpka      = 50,           ! ABL vertical levels number
  hmax      = 2000.,       ! ABL last level altitude
  theta_s   = 2.,         ! vertical grid stretching parameters
  hc        = 100.,       !
  ln_impose_z1 = .true.,   ! force ABL first level altitude
  z1        = 10.,       ! ABL first level imposed altitude [m]
/

&nml_opt
  ptemp_method = 3,      ! potential temperature computation method
  ln_slp_smth  = .true., ! smooth slp and ghv at high latitudes only
  ln_drw_smth  = .false., ! smooth after drowning
  ln_slp_log   = .false., ! read log(slp)
  ln_read_zsurf = .false., ! read surface geopotential
  ln_hpg_frc   = .true., ! compute horizontal pressure gradient
  ln_geo_wnd   = .false., ! compute geostrophic wind
  ln_c1d       = .false., ! 1D case
  ln_read_mask = .true., ! read mask file
  ln_lsm_land  = .false., ! inverse land & sea masks
  ln_perio_latbc = .true., ! periodic lateral boundary conditions
/

&nml_fld
  cn_dir      = '',
  mask_var    = 'LSM',
  file_m      = 'MASK.nc',
  file_u      = 'U3D.nc',
  file_v      = 'V3D.nc',
  file_t      = 'T3D.nc',
  file_q      = 'Q3D.nc',
  file_p      = 'P2D.nc',
  file_z      = 'Z2D.nc',
  file_geos   = 'UVG_OUT.nc',
  file_hpg    = 'HPG_OUT.nc',
/

&nml_out
  grd_file    = 'dom_cfg_abl_L50Z10.nc',
  abl_file    = 'ABL_L50Z10_OUT.nc',
  drwn_file   = 'ABL_DRWN_L50Z10_OUT.nc',
  var_name    = '',
/

&nml_c1d
  iloc = 283,
  jloc = 52,
/

```

done. Another specificity of ABL1D is that it can be dynamically driven by geostrophic wind or horizontal air pressure gradient, instead of being classically relaxed toward the large-scale wind forcing.

To generate the ABL1D vertical grid and atmospheric forcings, specific tools and an associated namelist are provided in the ABL\_TOOLS directory. They have been developed specifically to deal with ECMWF atmospheric products (such as ERA-Interim, ERA5 and IFS) on their native vertical eta-coordinates. The namelist is used to setup the ABL1D vertical grid (`&nml_dom`), atmospheric forcing options (`&nml_opt`), input atmospheric filenames (`&nml_fld`) and outputs filenames (`&nml_out`).

Each of the three steps needed to generate the atmospheric forcings corresponds to a tool:

- `main_uvghpg` (optional):  
geostrophic wind or horizontal pressure gradient computation on ECMWF eta-levels
- `main_vinterp`:  
air potential temperature computation and vertical interpolation from ECMWF vertical eta-levels to ABL z-levels
- `main_hdrown`:  
3D-fields horizontal drowning (extrapolation over land totally inspired from SOSIE by L. Brodeau)

### 7.5.2. ABL1D namelist

ABL1D model is activated by adding ABL sources directory to the sources list file (`_cfgs.txt`) and by setting `ln_abl=.true.` (and `ln_blk=.false.`) in `&namsbc` (namelist 7.1).

```

!-----
&namcbc_abl  !  Atmospheric Boundary Layer formulation      (ln_abl = T)
!-----
cn_dir       = './'      ! root directory for the location of the ABL grid file
cn_dom       = 'dom_cfg_abl'

cn_ablrst_in  = "restart_abl"  ! suffix of abl restart name (input)
cn_ablrst_out = "restart_abl"  ! suffix of abl restart name (output)
cn_ablrst_indir = "."        ! directory to read  input abl restarts
cn_ablrst_outdir = "."        ! directory to write output abl restarts

ln_rstart_abl = .false.
ln_hppls_frc  = .false.
  ln_pga_abl  = .false.  ! ABL pressure gradient anomaly forcing
ln_geos_winds = .false.
ln_smth_pblh  = .false.
nn_dyn_restore = 0      ! restoring option for dynamical ABL variables: = 0 no restoring
                        !                                           = 1 equatorial restoring
                        !                                           = 2 global restoring

rn_ldyn_min   = 0.      ! dynamics nudging magnitude inside the ABL [hour] (~3 rn_Dt)
rn_ldyn_max   = 0.      ! dynamics nudging magnitude above the ABL [hour] (~1 rn_Dt)
rn_ltra_min   = 0.      ! tracers nudging magnitude inside the ABL [hour] (~3 rn_Dt)
rn_ltra_max   = 0.      ! tracers nudging magnitude above the ABL [hour] (~1 rn_Dt)
rn_vfac       = 0.
nn_amxl       = 0      ! mixing length: = 0 Deardorff 80 length-scale
                        !                                           = 1 length-scale based on the distance to the PBL height
                        !                                           = 2 Bougeault & Lacarrere 89 length-scale
                        ! CBROO ! CCH02 ! MesoNH !
rn_Cm         = 0.0667 ! 0.0667 ! 0.1260 ! 0.1260 !
rn_Ct         = 0.1667 ! 0.1667 ! 0.1430 ! 0.1430 !
rn_Ce         = 0.40    ! 0.40    ! 0.34    ! 0.40    !
rn_Ceps       = 0.700  ! 0.700  ! 0.845  ! 0.850  !
rn_Ric        = 0.139  ! 0.139  ! 0.143  ! ?    ! Critical Richardson number (to compute PBL height and diffusivities)
rn_Rod        = 0.15   ! c0 in RMCA17 mixing length formulation (not yet implemented)
/

```

namelist 7.5.: &namcbc\_abl

It is fully compatible with Nemo Standalone Surface module and can be consequently forced by sea surface temperature and currents external data.

The namelist `&namcbc_abl` (namelist 7.5) is used to setup the ABL1D options.

Atmospheric forcing files needed by ABL1D must be specified directly using the `sn_wndi`, `sn_wndj`, `sn_tair` and `sn_humi` parameters from the `&namcbc_blk` (namelist 7.4).

When using geostrophic wind (`ln_geos_winds=.true.`) or horizontal air pressure gradient (`ln_hppls_frc=.true.`) as dynamical guide, additional `sn_hpgi` and `sn_hpgj` parameters must be provided using geostrophic wind/pressure gradient i/j-components files generated during the pre-processing steps.

Note that due to fldread limitations, the interpolation weight filenames must be different between 2D and 3D atmospheric forcings (even if it is the same weight file).

### Tracers and Dynamics relaxation time

ABL1D tracers needs to be relaxed toward atmospheric temperature (`sn_tair`) and humidity (`sn_humi`) forcings to provide a top boundary condition to the model and to avoid the formation of biases due to the lack of representation of some important atmospheric processes such as advection and convection. This relaxation time can be setup independently inside the ABL and above the ABL and it is expressed in hours.

The recommended values for the tracers relaxation time is typically 3 times the ocean model timestep inside the ABL (`rn_ltra_min`) and 1 ocean model timestep above the ABL (`rn_ltra_max`).

The dynamical relaxation time inside (`rn_ldyn_min`) and above (`rn_ldyn_max`) the ABL is only needed in two cases:

- when geostrophic wind / horizontal pressure gradient options are not used.
- when geostrophic wind / horizontal pressure gradient options are used and the geographical domain includes the equatorial band where the geostrophic equilibrium is too weak to constrain efficiently ABL1D dynamics.

The recommended minimum and maximum dynamical relaxation values are identical to the tracers relaxation times.

## Turbulent vertical mixing length and constants

The ABL1D turbulence scheme used to compute eddy diffusivities for momentum and scalars relies on a TKE prognostic equation (following Cuxart et al. (2000)) which depends on mixing length scales and turbulent constants. To address the ABL1D sensitivity to these parameters, various mixing length formulations and turbulent constants sets are provided in namelist:

- Three different mixing length scales can be selected using `mn_amx1` :
  - (0) Deardorff (1980)
  - (1) PBL height distance function (as in Nemo TKE scheme)
  - (2) Bougeault and Lacarrere (1989)
- Three different sets of turbulent constants are proposed: Cuxart et al. (2000), Cheng et al. (2002) and Lac et al. (2018)

	CBR00	CCH02	MNH54
<code>rn_Cm</code>	0.0667	0.1260	0.1260
<code>rn_Ct</code>	0.1667	0.1430	0.1430
<code>rn_Ce</code>	0.40	0.34	0.40
<code>rn_Ceps</code>	0.700	0.845	0.850
<code>rn_Ric</code>	0.139	0.143	?
<code>rn_Rod</code>	0.15	0.15	0.15

More details about the turbulence scheme parameters and their effect on ABL properties can be found in Lemarié et al. (2021).

## 7.6. 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).

When PISCES biogeochemical model ( `key_top` ) is also used in the coupled system, the whole carbon cycle is computed. In this case, CO<sub>2</sub> fluxes will be exchanged between the atmosphere and the ice-ocean system (and need to be activated in `&namsbc_cpl` ( [namelist 7.6](#) ) ).

When an external wave model (see [section 7.10](#)) is used in the coupled system, wave parameters, surface currents and sea surface height can be exchanged between both models (and need to be activated in `&namsbc_cpl` ( [namelist 7.6](#) ) ).

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 SI3). When indicating a multi-category coupling field in `&namsbc_cpl` ( [namelist 7.6](#) ), 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.

## 7.7. Atmospheric pressure ( *sbcapr.F90* )

The optional atmospheric pressure can be used to force ocean and ice dynamics ( `ln_apr_dyn=.true.` , `&namsbc` ( [namelist 7.1](#) ) namelist). The input atmospheric forcing defined via `sn_apr` structure ( `&namsbc_apr` ( [namelist 7.7](#) ) 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,  $\eta_{ib}$ , using:

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

```

!-----
&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)
ln_scale_ice_flux = .false. ! use ice fluxes that are already "ice weighted" ( i.e. multiplied ice concentration)
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' , '' , '' , '' , 'U,V'
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_w10m = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_tauomod = 'coupled' , 'no' , '' , '' , '' , ''
sn_rcv_tau = 'oce only' , 'no' , 'cartesian' , 'eastward-northward' , '' , ''
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_iceflx = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_mslp = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_ts_ice = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_qtrice = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_isf = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_icb = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_hsig = '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_charn = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_taw = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_bhd = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_tusd = 'none' , 'no' , '' , '' , '' , ''
sn_rcv_tvsd = 'none' , 'no' , '' , '' , '' , ''
/

```

namelist 7.6.: &namcbc\_cpl

```

!-----
&namcbc_apr ! Atmospheric pressure used as ocean forcing (ln_apr_dyn =T)
!-----
rn_pref = 101000. ! reference atmospheric pressure [N/m2]/
nn_ref_apr = 0 ! ref. pressure: 0: constant, 1: global mean or 2: read in a file
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 ! ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' ! !
↪ pairing ! filename !
↪ sn_apr = 'patm' , -1. , 'soms1pre' , .true. , .true. , 'yearly' , '' ,
↪ '' , '' , '' , '' , '' , '' , '' ,
↪ sn_apref = 'mean_patm' , -1. , 'meanapr' , .true. , .true. , 'yearly' , '' ,
↪ '' , '' , '' , '' , '' , '' , '' ,
/

```

namelist 7.7.: &namcbc\_apr



```

!-----
&nam_tide      !  tide parameters                                (default: OFF)
!-----
ln_tide       = .false.      ! Activate tides
nn_tide_var   = 1           ! Variant of tidal parameter set and tide-potential computation
!                   ! (1: default; 0: compatibility with previous versions)
ln_tide_pot   = .false.      ! use tidal potential forcing
rn_tide_gamma = 0.7         ! Tidal tilt factor
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
rn_tide_ramp_dt = 0.        ! ramp duration in days
sn_tide_cnames(1) = 'DUMMY' ! name of constituent - all tidal components must be set in namelist_cfg
/

```

namelist 7.8.: &nam\_tide

where  $P_{atm}$  is the atmospheric pressure and  $P_o$  a reference atmospheric pressure. 3 different options are available to define this reference atmospheric pressure using `nn_ref_apr` :

- 0 sets  $P_o$  to a constant value of  $101,000 \text{ N/m}^2$
- 1 sets  $P_o$  to the value of  $P_{atm}$  averaged over the ocean domain (the mean value of  $\eta_{ib}$  is kept to zero at all time steps)
- 2 reads a time-dependant  $P_o$  value from a 1D file defined via `sn_apref` structure (usefull for regional configurations)

The gradient of  $\eta_{ib}$  is added to the RHS of the ocean momentum equation (see *dynspg.F90* for the ocean). For sea-ice, the sea surface height,  $\eta_m$ , which is provided to the sea ice model is set to  $\eta - \eta_{ib}$  (see *sbcssr.F90* module).  $\eta_{ib}$  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  $\eta_{ib}$  can be added to BDY ssh data: `ln_apr_obc` might be set to true.

## 7.8. Surface tides (TDE)

### 7.8.1. Tidal constituents

Ocean model component TDE provides the common functionality for tidal forcing and tidal analysis in the model framework. This includes the computation of the gravitational surface forcing, as well as support for lateral forcing at open boundaries (see subsection 9.4.9) and tidal harmonic analysis . The module is activated with `ln_tide=.true.` in namelist `&nam_tide` (namelist 7.8) . It provides the same 34 tidal constituents that are included in the *FES2014 ocean tide model*: Mf, Mm, Ssa, Mtm, Msf, Msqm, Sa, K1, O1, P1, Q1, J1, S1, M2, S2, N2, K2, nu2, mu2, 2N2, L2, T2, eps2, lam2, R2, M3, MKS2, MN4, MS4, M4, N4, S4, M6, and M8; see file *tide.h90* and *tide\_mod.F90* for further information and references\*. Constituents to be included in the tidal forcing (surface and lateral boundaries) are selected by enumerating their respective names in namelist array `sn_tide_cnames` .

### 7.8.2. Surface tidal forcing

Surface tidal forcing can be represented in the model through an additional barotropic force in the momentum equation (equation 1.4a) such that:

$$\frac{\partial \mathbf{U}_h}{\partial t} = \dots + g \nabla (\gamma \Pi_{eq} + \Pi_{sal})$$

where  $\gamma \Pi_{eq}$  stands for the equilibrium tidal forcing scaled by a spatially uniform tilt factor  $\gamma$ , and  $\Pi_{sal}$  is an optional self-attraction and loading term (SAL). These additional terms are enabled when, in addition to `ln_tide=.true.` , the parameter `ln_tide_pot=.true.` .

\*As a legacy option `nn_tide_var` can be set to 0, in which case the 19 tidal constituents (M2, N2, 2N2, S2, K2, K1, O1, Q1, P1, M4, Mf, Mm, Msqm, Mtm, S1, MU2, NU2, L2, and T2; see file *tide.h90*) and associated parameters that have been available in NEMO version 4.0 and earlier are available

```

!-----
&nam_sbc_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_icb    = .false. ! read iceberg flux
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' , -12. , 'socoefr0' , .false. , .true. , 'yearly' , '' ,
sn_s_rnf = 'runoffs' , 24. , 'rosaline' , .true. , .true. , 'yearly' , '' ,
sn_t_rnf = 'runoffs' , 24. , 'rotemper' , .true. , .true. , 'yearly' , '' ,
sn_i_rnf = 'NOT USED' , 24. , 'xxxxxxx' , .true. , .true. , 'yearly' , '' ,
sn_dep_rnf = 'runoffs' , -12. , 'rodepth' , .false. , .true. , 'yearly' , '' ,
/
    
```

namelist 7.9.: &nam\_sbc\_rnf

The equilibrium tidal forcing is expressed as a sum over the subset of constituents listed in `sn_tide_cnames` of `&nam_tide` (namelist 7.8) (e.g.,

```

sn_tide_cnames(1) = 'M2'
sn_tide_cnames(2) = 'K1'
sn_tide_cnames(3) = 'S2'
sn_tide_cnames(4) = 'O1'
    
```

to select the four tidal constituents of strongest equilibrium tidal potential). The tidal tilt factor  $\gamma = 1 + k - h$  includes the Love numbers  $k$  and  $h$  (Love, 1909); this factor is configurable using `rn_tide_gamma` (default value 0.7). Optionally, when `ln_tide_ramp=.true.`, the equilibrium tidal forcing can be ramped up linearly from zero during the initial `rn_tide_ramp_dt` 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). The complex calculations involved in such computations, however, are computationally very expensive. Here, two mutually exclusive simpler variants are available: amplitudes generated by an external model for oscillatory  $\Pi_{sal}$  contributions from each of the selected tidal constituents can be read in (`ln_read_load=.true.`) from the file specified in `cn_tide_load` (the variable names are comprised of the tidal-constituent name and suffixes `z1` and `z2` for the two orthogonal components, respectively); alternatively, a “scalar approximation” can be used (`ln_scal_load=.true.`), where

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

with a spatially uniform coefficient  $\beta$ , which can be configured via `rn_scal_load` (default value 0.094) and is often tuned to minimize tidal prediction errors.

## 7.9. 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 <sup>†</sup>.

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.

The surface runoff is activated via the namelist parameter `ln_rnf` in `&nam_sbc` (namelist 7.1). The specific options that control the runoff are described in `&nam_sbc_rnf` (namelist 7.9) (namelist 7.9). In case of activation, the mandatory fields is the map of surface runoff that need to be specified by the user (`sn_rnf`). All the other parameters described below like the injection depth, the temperature, salinity (...) are optional.

By default the surface runoff is injected in the surface level. However, by activating `ln_rnf_depth`, it is possible to specify a 2D map of depth over which to inject it (`sn_dep_rnf`). The depth variable is expected to be positive with two specific values. A cell value of -1 means the river is added to the surface box only, and a cell value of -999 means the river is added through the entire water column. There is a third way to define the runoff thickness. The depth can be automatically computed (`ln_rnf_depth_ini`). The user simply need to give the depth over which the maximum runoff (`rn_rnf_max`) is spread (`rn_dep_max`). Then the runoff depth is computed simply by scaling the local maximum runoff value over time by the ratio  $\frac{rn\_dep\_max}{rn\_rnf\_max}$ . Once computed, by setting `nn_rnf_depth_file` the file can be outputted for sanity check.

After the depth is defined, the number of grid boxes this corresponds to is calculated and stored in the variable `nk_rnf`. The variable `h_dep` is then updated 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`) simulating a momentum flux. The sea surface height is then 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 tracers, by default (`ln_rnf_tem` and `ln_rnf_sal` set to *false*), the runoff is assumed to be at the river point `sst` and with salinity 0 g/kg. If set to true, the user needs to provide a runoff temperature and/or salinity field (`sn_t_rnf` and/or `sn_s_rnf` respectively). It is worth noting that location with a temperature value of -999 is considered as missing data and the river temperature is taken to be the surface temperature at the river point. Furthermore, the iceberg melt flux forcing (and the associated latent heat flux) can be added as runoff by activating `ln_rnf_icb` instead of using the lagrangian iceberg model (ICB, section 8.2) to simulate it. In this case, the user simply need to specify a map of iceberg melt rate in the file `sn_i_rnf`. In this case, the iceberg fresh water flux is added to the runoff fluxes and the latent heat flux directly to the non solar heat fluxes.

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. These fluxes are then added to the tracer trend in `trasbc.F90` (subsection 6.4.1). This is done in the same way for both linear and non-linear free surface. The temperature and salinity are increased through the specified depth according to the heat and salt content of the river.

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 the corresponding heat and salt (runoff temperature at surface temperature and salinity 0 g/kg by default) 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).

Furthermore, 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

<sup>†</sup>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 7.2.

```

!-----
&namcbc_wave ! External fields from wave model (ln_wave=T)
!-----
ln_sdw = .false. ! get the 2D Surf Stokes Drift & Compute the 3D stokes drift
ln_stcor = .false. ! add Stokes Coriolis and tracer advection terms
ln_cdgw = .false. ! Neutral drag coefficient read from wave model
ln_tauoc = .false. ! ocean stress is modified by wave induced stress
ln_wave_test= .false. ! Test case with constant wave fields
!
ln_charn = .false. ! Charnock coefficient read from wave model (IFS only)
ln_taw = .false. ! ocean stress is modified by wave induced stress (coupled mode)
ln_phioc = .false. ! TKE flux from wave model
ln_bern_srfc= .false. ! wave induced pressure. Bernoulli head J term
ln_breivikFV_2016 = .false. ! breivik 2016 vertical stokes profile
ln_vortex_force = .false. ! Vortex Force term
ln_stshear = .false. ! include stokes shear in EKE computation
!
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_wnum = 'sdw_ecwaves_orca2' , 6. , 'wave_num' , .true. , .true. , 'yearly' , '' ,
'' , ''
sn_tauoc = 'sdw_ecwaves_orca2' , 6. , 'wave_stress' , .true. , .true. , 'yearly' , '' ,
'' , ''
/

```

namelist 7.10.: &amp;namcbc\_wave

boxes.

In addition, there is possibility to increase vertical mixing near river mouths ( `ln_rnf_mouth` ). If activated, the `kz` is increased by `rn_avt_rnf` where the mask file `sn_cnf` is set to 0.5 over the number of level corresponding to the depth `rn_hrnf` . This depth value can be different to the one set in `sn_dep_rnf` .

Finally, it is worth nothing that 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.

## 7.10. Interactions with waves ( *sbcwave.F90* )

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, the vortex-force, the Bernoulli head J term and the Stokes drift impact on mass and tracer advection; moreover the neutral surface drag coefficient or the Charnock parameter from a wave model can be used to evaluate the wind stress. *NEMO* has also been extended to take into account the impact of surface waves on the vertical mixing, via the parameterization of the Langmuir turbulence, the modification of the surface boundary conditions for the Turbulent Kinetic Energy closure scheme, and the inclusion of the Stokes drift contribution to the shear production term in different turbulent closure schemes (TKE, GLS).

Physical processes related to ocean surface waves can be accounted by setting the logical variable `ln_wave=.true.` in `&namcbc` (namelist 7.1) 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** : the neutral drag coefficient, the two components of the surface Stokes drift, the significant wave height, the mean wave period, the mean wave number, and the normalized wave stress going into the ocean can be read from external files. Wave fields should be defined through the `&namsbc_wave` ([namelist 7.10](#)) namelist for external data names, locations, frequency, interpolation and all the miscellaneous options allowed by Input Data generic Interface (see [section 7.2](#)).

**coupled mode** : *NEMO* and an external wave model can be coupled by setting `ln_cpl=.true.` in `&namsbc` ([namelist 7.1](#)) namelist and filling the `&namsbc_cpl` ([namelist 7.6](#)) namelist. *NEMO* can receive the significant wave height, the mean wave period (*TOM1*), the mean wavenumber, the Charnock parameter, the neutral drag coefficient, the two components of the surface Stokes drift and the associated transport, the wave to ocean momentum flux, the net wave-supported stress, the Bernoulli head *J* term and the wave to ocean energy flux term.

The option `ln_wave_test=.true.` enables testing of ocean-wave interactions using an idealized constant wave field. The fields define are the surface Stokes drift (in the x-direction), as well as the significant wave height and mean wave period.

### 7.10.1. Neutral drag coefficient from wave model ( `ln_cdgw` )

The neutral surface drag coefficient provided from an external data source (*i.e.* forced or coupled wave model), can be used by setting the logical variable `ln_cdgw=.true.` in `&namsbc_wave` ([namelist 7.10](#)) namelist. The drag coefficient is computed according to the stable/unstable conditions of the air-sea interface following [Large and Yeager \(2004\)](#), starting from the neutral drag coefficient provided. This option is only available for the `ln_NCAR` and `ln_MFS` bulk formulae.

### 7.10.2. Charnok coefficient from wave model ( `ln_charn` )

The dimensionless Charnok parameter characterising the sea surface roughness provided from an external wave model, can be used by setting the logical variable `ln_charn=.true.` in `&namsbc_wave` ([namelist 7.10](#)) namelist. Then using the routine `sbcblk_algo_ecmwf`, the roughness length that enters the definition of the drag coefficient is computed according to the Charnok parameter depending on the sea state. Note that this option is only available in coupled mode and for `ln_ECMWF=.true.`.

### 7.10.3. 3D Stokes Drift ( `ln_sdw` )

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  $\mathbf{U}_{st}$  in deep water can be computed from the wave spectrum and may be written as:

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

where:  $\theta$  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  $\lambda$  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. Two possible parameterizations for the calculation for the approximate Stokes drift velocity profile are included in the code once provided the surface Stokes drift  $\mathbf{U}_{st}|_{z=0}$  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. To evaluate the 3D Stokes drift, the surface Stokes drift (zonal and meridional components), the Stokes transport or alternatively the significant wave height and the mean wave period should be provided either in forced or coupled mode.

By default `ln_breivikFV_2016=.false.` :

An exponential integral profile parameterization proposed by [Breivik et al. \(2014\)](#) is used:

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

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

$$k_e = \frac{|\mathbf{U}_{st|z=0}|}{5.97|T_{st}|} \quad \text{and} \quad T_{st} = \frac{1}{16} \bar{\omega} H_s^2$$

where  $H_s$  is the significant wave height and  $\bar{\omega}$  is the wave frequency defined as:  $\bar{\omega} = \frac{2\pi}{T_m}$  (being  $T_m$  the mean wave period).

If `ln_breivikFV_2016=.true.` :

A 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) is used:

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

where  $\operatorname{erf}$  is the complementary error function,  $\beta = 1$  and  $k_p$  is the peak wavenumber defined as:

$$k_p = \frac{|\mathbf{U}_{st|z=0}|}{2|T_{st}|} (1 - 2\beta/3)$$

$|T_{st}|$  is estimated from integral wave parameters ( $H_s$  and  $T_m$ ) in forced mode and is provided directly from an external wave model in coupled mode.

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  $\eta$  by including the barotropic Stokes transport horizontal divergence in the term  $D$  of Eq.1.6

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:

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

#### 7.10.4. 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  $\mathbf{U}_{st} \times f$ , 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 2 possible approximations described in subsection 7.10.3), `ln_stcor=.true.` has to be set.

#### 7.10.5. Vortex-force term ( `ln_vortex_force` )

The vortex-force term arises from the interaction of the mean flow vorticity with the Stokes drift. It results in a force equal to  $\mathbf{U}_{st} \times \zeta$ , where  $\zeta$  is the mean flow vorticity. In order to include this term, once evaluated the Stokes drift (using one of the 2 possible approximations described in subsection 7.10.3), `ln_vortex_force=.true.` has to be set.

#### 7.10.6. Wave-induced pressure term ( `ln_bern_srfc` )

An adjustment in the mean pressure arises to accommodate for the presence of waves. The mean pressure is corrected adding a depth-uniform wave-induced kinematic pressure term named Bernoulli head  $J$  term. The Bernoulli head  $J$  term is provided to NEMO from an external wave model where it is defined as:

$$J = g \iint \frac{k}{\sinh(2kd)} S(k, \theta) d\theta dk$$

with  $d$  the water depth.

In order to include this term, `ln_bern_srfc=.true.` has to be defined as well as the Stokes drift option (subsection 7.10.3) and the coupling with an external wave model (section 7.10).



### 7.10.7. Wave modified stress ( `ln_tauoc` & `ln_taw` )

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  $\tau_a$  is the atmospheric surface stress;  $\tau_w$  is the atmospheric stress going into the waves defined as:

$$\tau_w = \rho g \int_0^{2\pi} \int \frac{1}{c_p} (S_{in} + S_{nl} + S_{diss}) dk d\theta$$

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. Note that the  $S_{nl}$  is not always taken into account for the calculation of the atmospheric stress going into the waves, depending on the external wave model. The wave stress derived from an external wave model can be provided either through the normalized wave stress into the ocean by setting `ln_tauoc=.true.`, or through the zonal and meridional stress components by setting `ln_taw=.true.`. In coupled mode both options can be used while in forced mode only the first option is included.

If the normalized wave stress into the ocean ( $\tilde{\tau}$ ) is provided (`ln_tauoc=.true.`) the atmospheric stress felt by the ocean circulation is expressed as:

$$\tau_{oc,a} = \tau_a \times \tilde{\tau}$$

If `ln_taw=.true.`, the zonal and meridional stress fields components from the coupled wave model have to be sent directly to u-grid and v-grid through OASIS.

### 7.10.8. Waves impact vertical mixing ( `ln_phioc` & `ln_stshear` )

The vortex-force vertical term gives rise to extra terms in the turbulent kinetic energy (TKE) prognostic (Couvelard et al., 2020). The first term corresponds to a modification of the shear production term. The Stokes Drift shear contribution can be included, in coupled mode, by setting `ln_stshear=.true.`.

In addition, waves affect the surface boundary condition for the turbulent kinetic energy, the mixing length scale and the dissipative length scale of the TKE closure scheme. The injection of turbulent kinetic energy at the surface can be given by the dissipation of the wave field usually dominated by wave breaking.

In coupled mode, the wave to ocean energy flux term from an external wave model ( $\Phi_o$ ) can be provided to NEMO and then converted into an ocean turbulence source by setting `ln_phioc=.true.`. The boundary condition for the turbulent kinetic energy is implemented in the `zdf_tke` as a Dirichlet or as a Neumann boundary condition (see subsection 11.1.4). The boundary condition for the mixing length scale and the dissipative length scale can also account for surface waves (see subsection 11.1.4)

Some improvements are introduced in the Langmuir turbulence parameterization (see chapter 11 subsection 11.1.4) if wave coupled mode is activated.

## 7.11. Miscellaneous options

### 7.11.1. Diurnal cycle ( `sbc_dcy.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 `&namesbc` (namelist 7.1) namelist variable) when using a bulk formulation (`ln_blk=.true.`) or the flux formulation (`ln_flg=.true.`). The reconstruction is performed in the `sbc_dcy.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



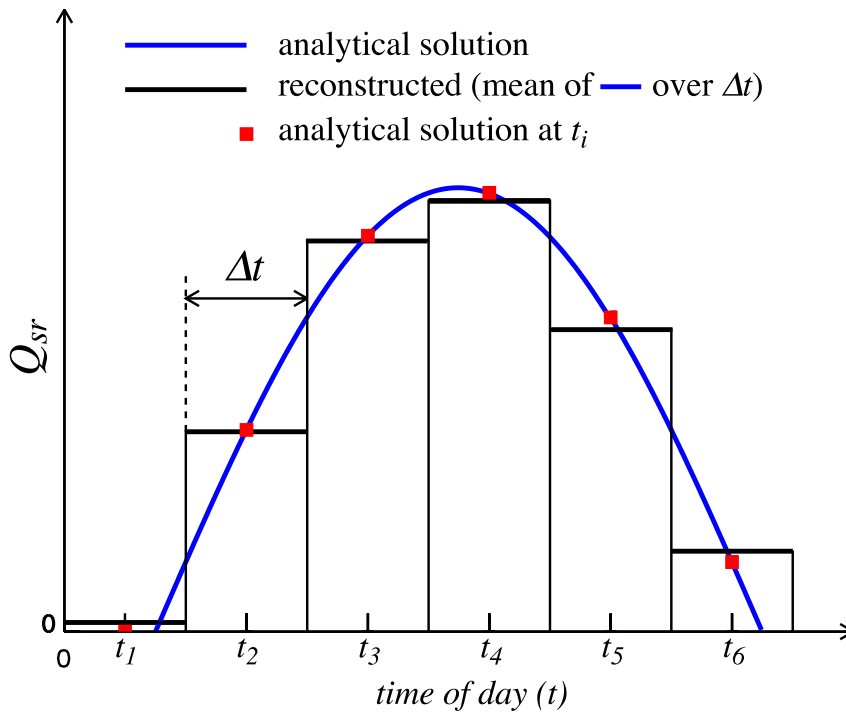


Figure 7.1.: Example of reconstruction of the diurnal cycle variation of short wave flux from daily mean values. The reconstructed diurnal cycle (black line) is chosen as the mean value of the analytical cycle (blue line) over a time step, not as the mid time step value of the analytical cycle (red square). From [Bernie et al. \(2007\)](#).

is the mean value of the analytical cycle over this time step ([figure 7.1](#)). 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  $\Delta t \text{ nn\_fsbc} < 10,800 \text{ s} = 3 \text{ h}$ . An example of reconstructed SWF is given in [figure 7.2](#) for a 12 reconstructed diurnal cycle, one every 2 hours (from 1am to 11pm).

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

### 7.11.2. Rotation of vector pairs onto the model grid directions

When using a flux (`ln_flux=.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 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.

### 7.11.3. Surface restoring to observed SST and/or SSS (`sbcssr.F90`)

The addition of a surface restoring term to observed SST and/or SSS can be activated defining `ln_ssr=.true.` in `&nam_sbc` ([namelist 7.1](#)). Options are defined through the `&nam_sbc_ssr` ([namelist 7.11](#)) namelist variables. On forced mode using a flux formulation (`ln_flux=.true.`), a feedback term *must* be added to the surface heat flux  $Q_{ns}^o$  (`nn_sstr= 1`):

$$Q_{ns} = Q_{ns}^o + \frac{dQ}{dT} (T|_{k=1} - SST_{Obs}) \quad (7.2)$$

where SST is a sea surface temperature field (observed or climatological, `sn_sst`),  $T$  is the model surface layer temperature and  $\frac{dQ}{dT}$  (`rn_dqdt`) is a negative feedback coefficient usually taken equal to  $-40 \text{ W/m}^2/\text{K}$ .

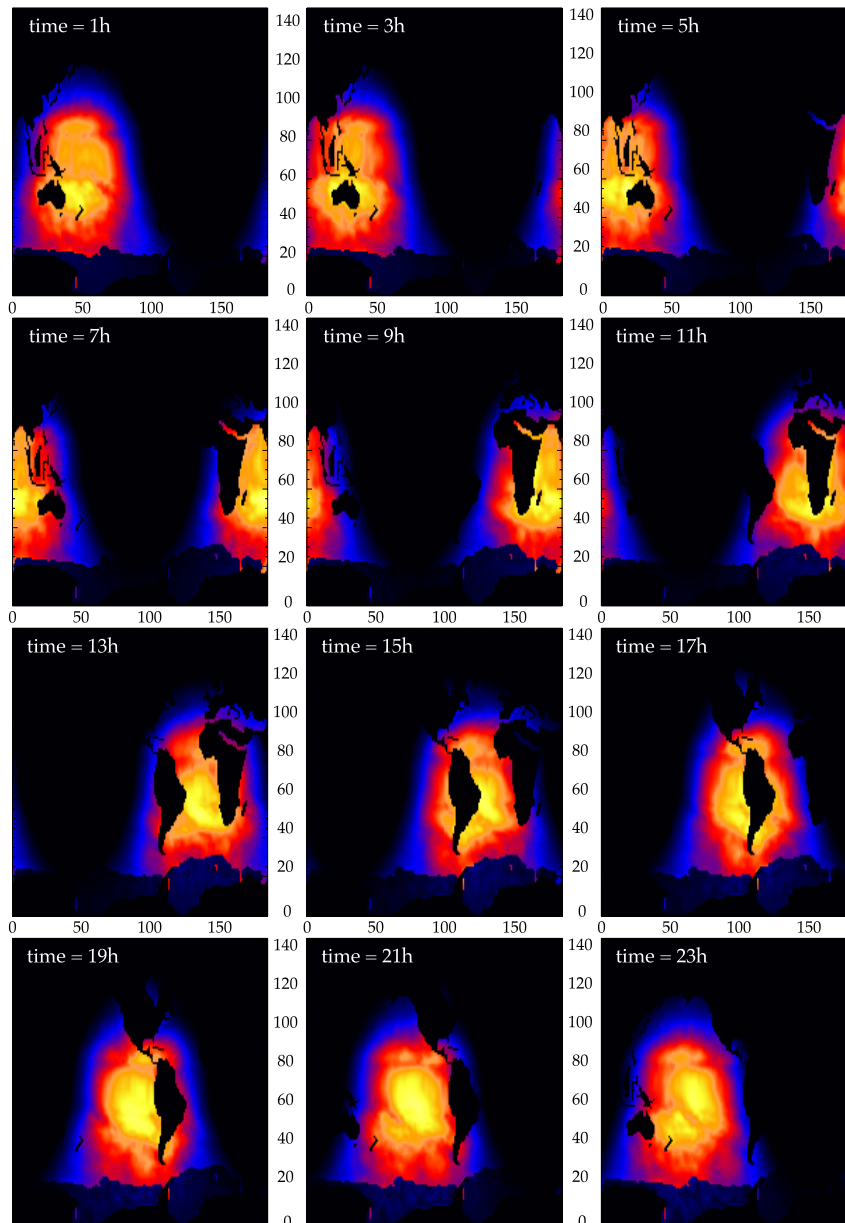


Figure 7.2.: Example of reconstruction of the diurnal cycle variation of short wave flux from daily mean values on an ORCA2 grid with a time sampling of 2 hours (from 1am to 11pm). The display is on (i,j) plane.

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 (`nn_ssr= 1`). Converted into an equivalent freshwater flux, it takes the following expression :

$$emp = emp_o + \gamma_s^{-1} e_{3t} \frac{(S|_{k=1} - SSS_{Obs})}{S|_{k=1}} \quad (7.3)$$

where  $emp_o$  is a net surface fresh water flux (observed, climatological or an atmospheric model product),  $SSS_{Obs}$  is a sea surface salinity estimate (`sn_sss`),  $S|_{k=1}$  is the model surface layer salinity and  $\gamma_s$  is a negative feedback coefficient which is provided as `rn_deds`. At high resolution is is worth bounding the restoring term (`ln_ssr_bnd`) because it can be extremely high in structured not resolved in the sss observation (filament or eddies). If activated the maximum value of the correction term is set to `rn_ssr_bnd`. There is also an option to deactivate the salinity restoring under sea ice (`nn_ssr_ice`). There is 3 options:

**nn\_ssr\_ice = 0** : no restoring under sea ice. This can be justify by the fact that the observation under the sea ice less reliable as in the open ocean.

**nn\_ssr\_ice = 1** : same restoring under sea ice than within the open ocean

```

!-----
&namcbc_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]
nn_sssr_ice = 1 ! control of sea surface restoring under sea-ice
! 0 = no restoration under ice : * (1-icefrac)
! 1 = restoration everywhere
! >1 = enhanced restoration under ice : 1+(nn_icedmp-1)*icefrac

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 7.11.: &namcbc\_ssr

**nn\_sssr\_ice > 1** : strong restoring under the ice. In this case, the user trust more the sss observation that its sea ice model as the restoring will tend to strongly constrain the surface salinity that in reality is strongly driven by the sea ice melt / formation in these area.

If the sea surface salinity estimates do not resolve the river fresh water plumes, the restoring will tend to counter balance the effect of the surface runoff in the model. To avoid such effect, the salinity restoring is not applied at the river mouth mask defined when the runoff parameter `ln_rnf_mouth` is activated.

Unlike heat flux, there is no physical justification for the feedback term in [equation 7.3](#) 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.

#### 7.11.4. 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 `&namcbc` ([namelist 7.1](#)) 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.

**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** : 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`). The activation automatically overwrites the read value of `nn_ice` to its appropriate value (*i.e.* 2 for SI3).

#### 7.11.5. Freshwater budget control (*sbcfwb.F90*)

For global ocean simulations, it can be useful to introduce a control of the mean sea level in order to prevent unrealistic drifting of the sea surface height due to unbalanced freshwater fluxes. In *NEMO*, four options for controlling the freshwater budget are proposed:

```

!-----
&namcbc_fwb ! freshwater-budget adjustment (nn_fwb > 0)
!-----
rn_fwb0 = 0.0 ! Initial freshwater adjustment flux [kg/m2/s] (nn_fwb = 2)
nn_fwb_voltype = 1 ! = 1 : Control ICE+OCEAN volume
! ! = 2 : Control OCEAN volume
ln_hvolg_var = .false. ! = T : Set an analytical variation of volume:
rn_hvolg_amp = 17.e-3 ! Peak to peak seasonal variation (m)
rn_hvolg_trd = 0.0 ! Trend (m/s)
nn_hvolg_mth = 8 ! Month when volume anomaly crosses 0 (1-12)
/

```

namelist 7.12.: &namcbc\_fwb

**nn\_fwb=0** : No control at all; the mean sea level is free to drift, and will certainly do so.

**nn\_fwb=1** : The global mean *emp* is set to zero at each model time step.

**nn\_fwb=2** : *emp* is adjusted by adding a spatially uniform, annual-mean freshwater flux that balances the freshwater budget at the end of the previous year; as the model uses the Boussinesq approximation, the freshwater budget can be evaluated from the change in the mean sea level and in the ice and snow mass after the end of each simulation year; at the start of the model run, an initial adjustment flux can be set using parameter **rn\_fwb0** in namelist &namcbc\_fwb (namelist 7.12) .

**nn\_fwb=3** : volume adjusted at each time step and spread out over >0 erp (sea surface salinity restoring term) area to increase evaporation or spread out over <0 erp area to increase precipitation.

**nn\_fwb=4** : volume adjusted at each time step with a correction on non solar heat flux (*qns*) and salt flux (*sflx*) to avoid any surface buoyancy flux associated with the *emp* correction.

If **nn\_fwb** is set > 0, &namcbc\_fwb (namelist 7.12) block must be filled accordingly:

**rn\_fwb0** : if **nn\_fwb=2** , it defines the initial freshwater adjustment flux.

**nn\_fwb\_voltype** : it refers to the variable considered as the global value of volume (in equivalent liquid height in m) to be conserved by the adjustment process.

**nn\_fwb\_voltype** : if set to 1, the total ocean+equivalent liquid sea ice volume water budget is controled, or if set to 0, only the ocean volume is controled. The former is now the default and recommended, being obviously more accurate on a physical point of view.

**ln\_hvolg\_var** : if set to .true., an analytical variation of the global liquid height can be specified by the user. It is defined as the sum of an annual harmonic signal (with a peak to peak amplitude **rn\_hvolg\_amp** in m, and a zero crossing at the beginning of month **nn\_hvolg\_mth** ) and a linear trend given by **rn\_hvolg\_trd** (in m/s).

## Land Ice Ocean interactions (ISF and ICB)

### Table of contents

8.1. Ice Shelf (ISF)	108
8.1.1. Ocean/Ice shelf fluxes in opened cavities ( <i>isfcav.F90</i> )	108
8.1.2. Ocean/Ice shelf fluxes in parametrised cavities ( <i>isfpar.F90</i> )	112
8.1.3. Available outputs ( <i>isfdiags.F90</i> )	112
8.1.4. Ice sheet coupling ( <i>isfcpl.F90</i> )	113
8.1.5. Ice shelf load ( <i>isfload.F90</i> )	114
8.1.6. Under ice shelf cavity geometry ( <i>domisf.F90</i> )	114
8.2. Icebergs (ICB)	115
8.2.1. Iceberg initialisation ( <i>icbclv.F90</i> )	115
8.2.2. Iceberg dynamics ( <i>icbdyn.F90</i> )	117
8.2.3. Iceberg thermodynamics ( <i>icbthm.F90</i> )	117
8.2.4. Iceberg mpp ( <i>icblbc.F90</i> )	118
8.2.5. Iceberg outputs ( <i>icbdia.F90</i> )	118
8.3. Land Ice Runoff	119

### Changes record

Release	Author(s)	Modifications
5.0	<i>Pierre Mathiot and Katherine Hutchinson</i>	<i>update of the ice shelf iceberg</i>
4.2	<i>Pierre Mathiot</i>	<i>update of the ice shelf section (2019 developments)</i>
4.0	...	...
3.6	...	...

Land ice includes ice sheets, icebergs, and ice-shelves. Land ice / ocean interactions encompasses ice-shelves, glacier termini and icebergs melting, as well as surface and sub-glacial runoff from the ice sheet. Land ice builds up through the accumulation of snowfall over Greenland and Antarctica. It influences the ocean through the melting of ice-shelves or glacier termini at the edge of the continents. This can also be via the calving of icebergs that slowly drift at the ocean surface and also by surface and subsurface subglacial runoff induced by ice sheet surface melting (seasonal variations).

Although land ice and sea ice bear some physical similarities, the modelling components to handle them are usually drastically different due to the differing scales of the problem and the distinctly separate processes at play. In NEMO, basal ice-shelf melting is handled through the ISF module (Mathiot et al., 2017), icebergs are handled through the ICB module (Marsh et al., 2015), and surface runoff is handled through the runoff module. Subglacial runoff is not implemented mostly because there is no consolidated data set to prescribed it.

## 8.1. Interaction with ice shelves (ISF)

The activation or not of the ice shelf, and the accompanying ocean interaction, is controlled by the namelist variable `ln_isf` in `&namisf` (namelist 8.1). The following interactions modes are available:

- representation of the ice shelf/ocean melting/freezing for open cavities (`cav`, `ln_isfcav_mlt`, fig. 8.1b,c).
- parametrisation of the ice shelf/ocean melting/freezing for closed cavities (`par`, `ln_isfpar_mlt`, fig. 8.1d,e).
- coupling with an ice sheet model (`ln_isfcpl`).

The outcomes of the ISF module are the fresh water flux and the associated heat flux. A description and result of sensitivity tests to `ln_isfcav_mlt` and `ln_isfpar_mlt` are presented in Mathiot et al. (2017).

### 8.1.1. Ocean/Ice shelf fluxes in opened cavities (`isfcav.F90`)

`ln_isfcav_mlt = .true.` activates the ocean/ice shelf thermodynamic interactions at the ice shelf/ocean interface. If `ln_isfcav_mlt = .false.`, thermodynamic interactions are deactivated but the ocean dynamics inside the cavity are still active. If `ln_isfcav_mlt` is activated, the user needs to make sure that the `domain_cfg.nc` input file includes ice shelf cavities (subsection 8.1.6).

As part of the `isfcavmlt.F90` module, 3 options are available to represent the ice-shelf/ocean fluxes at the interface:

`cn_isfcav_mlt = 'spe'` : The fresh water flux is specified by forcing fields `sn_isfcav_fwf`. Convention of the input file is: positive toward the ocean (i.e. positive for melting and negative for freezing). The latent heat flux is derived from the fresh water flux. The heat content flux is derived from the fresh water flux assuming a temperature set to freezing point in the top boundary layer (`rn_htbl`).

`cn_isfcav_mlt = 'oasis'` : The `'oasis'` is a prototype of what could be a method to spread precipitation landing on the Antarctic ice sheet as ice shelf melt inside the open cavities when a coupled model (Atmosphere/Ocean) is used. It has been tested in the IPSL model only. Therefore, it is highly recommended that other coupled models use it with caution. Feedbacks are very welcome.

`cn_isfcav_mlt = '2eq'` : The heat flux and the fresh water flux due to ice shelf melting/freezing are parameterized following Grosfeld et al. (1997). This formulation is based on a balance between the vertical diffusive heat flux across the ocean top boundary layer (equation 8.1) and the latent heat due to melting/freezing (equation 8.2):

$$Q_h = \rho c_p \gamma (T_w - T_f) \quad (8.1)$$

$$q = \frac{-Q_h}{L_f} \quad (8.2)$$

where  $Q_h (W.m^{-2})$  is the heat flux,  $q (kg.s^{-1}m^{-2})$  the fresh-water flux,  $L_f$  the specific latent heat,  $T_w$  the temperature averaged over a boundary layer below the ice shelf (explained below),  $T_f$  the freezing point using the pressure at the ice shelf base and the salinity of the water in the boundary layer, and  $\gamma$  the thermal exchange coefficient.

```

!-----
&namisf      ! Top boundary layer (ISF)                               (default: OFF)
!-----
!
!----- ice shelf load -----
!
cn_isfload = 'uniform'      ! scheme to compute ice shelf load (ln_isfcav = .true. in domain_cfg.nc)
    rn_isfload_T = -1.9
    rn_isfload_S = 34.4
!
!----- ice shelf melt formulation -----
!
ln_isf = .false.           ! activate ice shelf module
ln_isfdebug = .false.     ! add debug print in ISF code (global min/max/sum of specific variable)
cn_isfdir = './'          ! directory for all ice shelf input file
!
!----- cavities opened -----
!
ln_isfcav_mlt = .false.   ! ice shelf melting into the cavity (need ln_isfcav = .true. in domain_cfg.nc)
cn_isfcav_mlt = '3eq'    ! ice shelf melting formulation (spe/2eq/3eq/oasis)
!                               ! spe = fwfisf is read from a forcing field (melt > 0; freezing < 0)
!                               ! 2eq = ISOMIP like: 2 equations formulation (Hunter et al., 2006 for a short description)
!                               ! 3eq = ISOMIP+ like: 3 equations formulation (Asay-Davis et al., 2016 for a short description)
!                               ! oasis = fwfisf is given by oasis and pattern by file sn_isfcav_fwf
!                               ! cn_isfcav_mlt = 2eq or 3eq cases:
cn_gammat = 'vel'        ! scheme to compute gammat/s (spe,ad15,hj99)
!                               ! spe = constant transfert velocity (rn_gammat0, rn_gammas0)
!                               ! vel = velocity dependent transfert velocity (u* * gammat/s) (Asay-Davis et al. 2016 for a
↳ short description)
!                               ! vel_stab = velocity and stability dependent transfert coeficient (Holland et al. 1999 for a
↳ complete description)
rn_gammat0 = 1.4e-2      ! gammat coefficient used in spe, vel and vel_stab gamma computation method
rn_gammas0 = 4.0e-4     ! gammas coefficient used in spe, vel and vel_stab gamma computation method
!
rn_htbl = 30.           ! thickness of the top boundary layer (Losh et al. 2008)
!                               ! 0 => thickness of the tbl = thickness of the first wet cell
!
!* 'spe' and 'oasis' case
!-----
!-----!-----!-----!-----!-----!-----!-----!-----!-----!-----!
! 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 !
    sn_isfcav_fwf = 'isfmlt_cav', -12. , 'fwflisf' , .false. , .true. , 'yearly' , '' , '' ,
↳ ''
!
!----- cavities parametrised -----
!
ln_isfpar_mlt = .false.   ! ice shelf melting parametrised
cn_isfpar_mlt = 'spe'    ! ice shelf melting parametrisation (spe/bg03/oasis)
!                               ! spe = fwfisf is read from a forcing field (melt > 0; freezing < 0)
!                               ! bg03 = melt computed using Beckmann and Goosse parametrisation
!                               ! oasis = fwfisf is given by oasis and pattern by file sn_isfpar_fwf
!
!* bg03 case
rn_isfpar_bg03_gt0 = 1.0e-4 ! gamma coeficient used in bg03 paper [m/s]
!
!*** File definition ***
!
!* all cases
!-----
!-----!-----!-----!-----!-----!-----!-----!-----!-----!-----!
! 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 !
    sn_isfpar_zmax = 'isfmlt_par', 0. , 'sozsisfmax', .false. , .true. , 'yearly' , '' , '' ,
↳ ''
    sn_isfpar_zmin = 'isfmlt_par', 0. , 'sozsisfmin', .false. , .true. , 'yearly' , '' , '' ,
↳ ''
!
!* 'spe' and 'oasis' case
↳ sn_isfpar_fwf = 'isfmlt_par' , -12. , 'sofwfisf' , .false. , .true. , 'yearly' , '' , '' ,
!
!* 'bg03' case
!* Leff is in [km]
↳ sn_isfpar_Leff = 'isfmlt_par', 0. , 'Leff' , .false. , .true. , 'yearly' , '' , '' ,
!
!----- ice sheet coupling -----
!
ln_isfcpl = .false.
nn_drown = 10           ! number of iteration of the extrapolation loop (fill the new wet cells)
ln_isfcpl_cons = .false.
/
    
```

namelist 8.1.: &amp;namisf



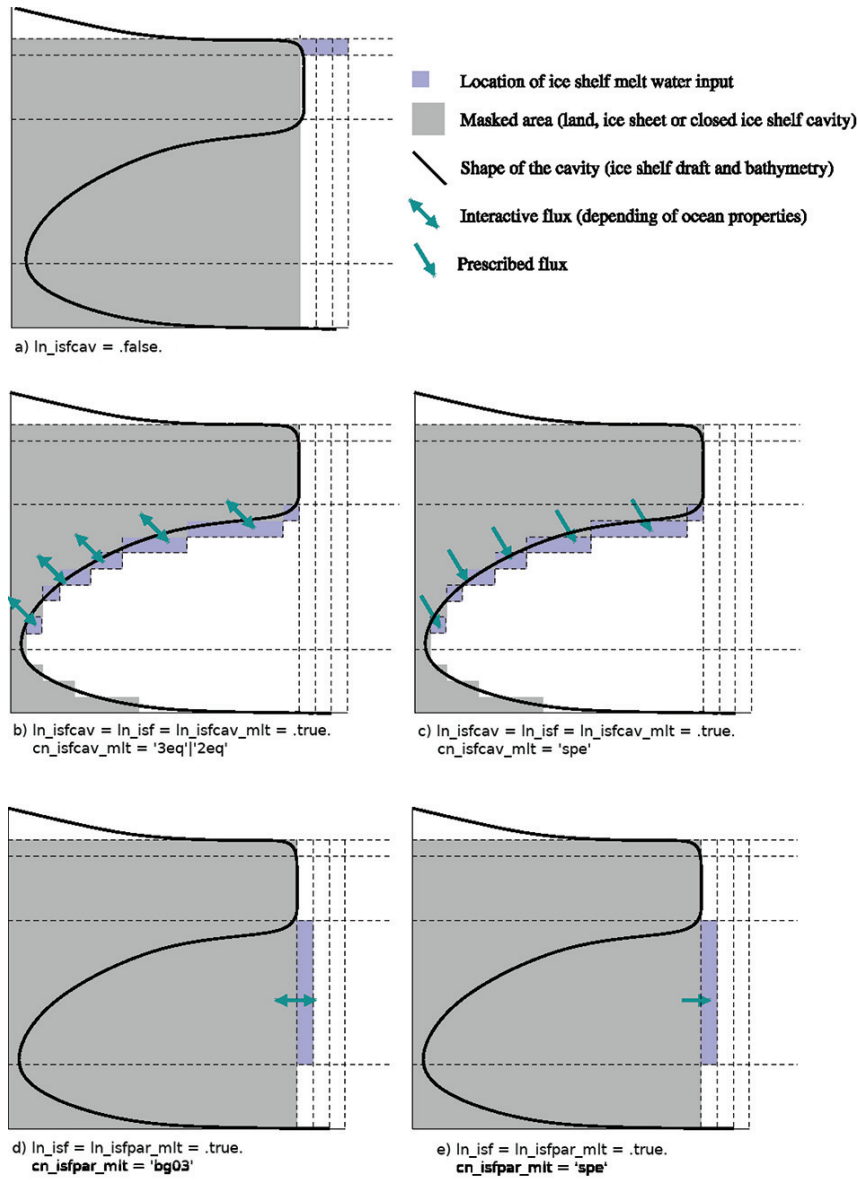


Figure 8.1.: Illustration of the location where the fwf is injected and whether or not the fwf is interactive or not.

`cn_isfcav_mlt = '3eq'` : For realistic studies, the heat and freshwater fluxes are parameterized following (Jenkins et al., 2001). This formulation is based on three equations: a balance between the vertical diffusive heat flux across the boundary layer with variations in latent heat due to melting/freezing of ice and the vertical diffusive heat flux into the ice shelf (equation 8.3); a balance between the vertical diffusive salt flux across the boundary layer with the salt source or sink represented by melting/freezing (equation 8.4); and a linear equation for the freezing temperature of sea water (Asay-Davis et al., 2016, equation 8.5 detailed in the linearisation coefficient of ):

$$c_p \rho \gamma_T (T_w - T_b) = -L_f q - \rho_i c_{p,i} \kappa \frac{T_s - T_b}{h_{isf}} \quad (8.3)$$

$$\rho \gamma_S (S_w - S_b) = (S_i - S_b) q \quad (8.4)$$

$$T_b = \lambda_1 S_b + \lambda_2 + \lambda_3 z_{isf} \quad (8.5)$$

where  $T_b$  is the temperature at the interface,  $S_b$  the salinity at the interface,  $\gamma_T$  and  $\gamma_S$  the exchange velocities for temperature and salt respectively,  $S_i$  the salinity of the ice (assumed to be 0),  $h_{isf}$  the ice shelf thickness,  $z_{isf}$  the ice shelf draft,  $\rho_i$  the density of the iceshell,  $c_{p,i}$  the specific heat capacity of the ice,  $\kappa$  the thermal diffusivity of the ice and  $T_s$  is the atmospheric surface temperature (at the ice/air interface, assumed to be  $-20$  °C). The Liquidus slope ( $\lambda_1$ ), the liquidus intercept ( $\lambda_2$ ) and the Liquidus pressure

coefficient ( $\lambda_3$ ) for TEOS80 and TEOS10 are described in [Asay-Davis et al. \(2016\)](#) and in [Jourdain et al. \(2017\)](#) respectively. The linear system formed by [equation 8.3](#), [equation 8.4](#) and the linearised equation for the freezing temperature of sea water ([equation 8.5](#)) can be solved for  $S_b$  or  $T_b$ . Afterward, the freshwater flux ( $q$ ) and the heat flux ( $Q_h$ ) can be computed.

Table 8.1.: Description of the parameters hard coded into the ISF module

Symbol	Description	Value	Unit
$C_p$	Ocean specific heat	3992	$J.kg^{-1}.K^{-1}$
$L_f$	Ice latent heat of fusion	$3.34 \times 10^5$	$J.kg^{-1}$
$C_{p,i}$	Ice specific heat	2000	$J.kg^{-1}.K^{-1}$
$\kappa$	Heat diffusivity	$1.54 \times 10^{-6}$	$m^2.s^{-1}$
$\rho_i$	Ice density	920	$kg.m^3$

Temperature and salinity used to compute the fluxes in [equation 8.1](#), [equation 8.3](#) and [equation 8.4](#) are the average values for the top boundary layer ([Losch, 2008](#)). The boundary layer thickness is defined by `rn_htbl`. The fluxes and friction velocity are computed using the mean temperature, salinity and velocity in the first `rn_htbl` m (see *isftbl.F90*). Then, the fluxes are spread over the same thickness (ie over one or several cells). If `rn_htbl` is larger than the top  $e3t$ , then there is no more direct feedback between the freezing point at the interface and the temperature of the top cell. This can therefore lead to a super-cool temperature in the top cell under melting conditions. If `rn_htbl` is smaller than the top  $e3t$ , then the top boundary layer thickness is set to the top cell thickness.

For each melt formula '3eq' or '2eq', the turbulent transfer velocities ( $\gamma^{T,S}$ ) between the ocean and the ice can be computed in any of the following 3 ways:

`cn_gammabl` = 'spe' : The salt and heat transfert exchange velocities are constant and defined by:

$$\gamma^T = \Gamma^T$$

$$\gamma^S = \Gamma^S$$

This is the recommended formulation for ISOMIP.

`cn_gammabl` = 'vel' : The salt and heat exchange coefficients are velocity dependent and defined as

$$\gamma^T = \Gamma^T \times u_*$$

$$\gamma^S = \Gamma^S \times u_*$$

where  $u_*$  is the friction velocity in the top boundary layer (ie first `rn_htbl` meters).

See [Jenkins et al. \(2010\)](#) for all the details on this formulation. This is the recommended formulation for realistic applications and ISOMIP+/MISOMIP configurations.

`cn_gammabl` 'vel\_stab' : The salt and heat exchange coefficients are velocity and stability dependent, and are 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_tbl` meters),  $\Gamma_{Turb}$  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 (and is thus not recommended).

In the formulation presented above, the transfert coefficient  $\Gamma^T$  and  $\Gamma^S$  are respectively defined by `rn_gammat0` and `rn_gammas0`. The definition of the exchange velocities  $\gamma^{T,S}$  is done in the *isfcavgam.F90* module.

The ice shelf fresh water fluxes are implemented as a volume flux, same as for the runoff. The fresh water flux addition due to the ice shelf melting is distributed uniformly over the top boundary layer and added to the horizontal divergence (*hdivn*) at each relevant depth level in the subroutine `isf_hdiv`, called from *divhor.F90*. As for the volume flux, the associated heat fluxes (latent heat and heat content) are distributed uniformly vertically over the top boundary layer in *traisf.F90*. See the runoff section [section 7.9](#) for all the details about the divergence correction.

### 8.1.2. Ocean/Ice shelf fluxes in parametrised cavities ( *isfpar.F90* )

For a low resolution model, many ice shelves are too small to be explicitly represented. It is therefore necessary to parametrise or specify the melt from these unresolved cavities ( `ln_isfpar_mlt=.true.` ) in addition of some explicit cavities like Ross ice shelf and Filchner Ronne ice shelf, for exemple. The *isfpar.F90* module offers two options defined in the *isfparmlt.F90* module:

`cn_isfpar_mlt = 'bg03'` : 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 front between the depth of the average grounding line (GL) ( `sn_isfpar_zmax` ) and the base of the ice shelf along the calving front ( `sn_isfpar_zmin` ) as in ( `cn_isfpar_mlt = 'spe'` ). The exchange velocity is specified by `rn_isfpar_bg03_gt0` and the effective melting length ( `sn_isfpar_Leff` ) is read from a file. This parameterisation, however, has not been tested for some time and based on Favier et al. (2019) and more recently Burgard et al. (2022), the advice is that this parameterisation should not be used.

`cn_isfpar_mlt = 'spe'` : The fwf ( $q$ ) is read from `sn_isfpar_fwf` and distributed along the ice shelf front between the average grounding line (GL) depth ( `sn_isfpar_zmax` ) and the depth of the base of the ice shelf at the front edge ( `sn_isfpar_zmin` ). Convention of the input file is positive toward the ocean (i.e. positive for melting and negative for freezing). The heat flux ( $Q_h$ ) is computed as  $Q_h = q \times L_f$ .

`cn_isfpar_mlt = 'oasis'` : The 'oasis' method is a prototype. It spreads the precipitation received by the Antarctic ice sheet as ice shelf melt inside the cavity when a coupled model (Atmosphere/Ocean) is used. It has only been tested in the IPSL model and it is therefore recommended that other coupled models use it with caution. Feedbacks are very welcome.

As for the open cavity case, the ice shelf fresh water fluxes are implemented as a volume flux. In this case the volume and heat fluxes are uniformly distributed between `sn_isfpar_zmin` and `sn_isfpar_zmax`.

To conclude these two sections on ice shelf melt, it is worth noting the following:

`cn_isfcav_mlt = '2eq'`, `cn_isfcav_mlt = '3eq'` and `cn_isfpar_mlt = 'bg03'` computes a melt rate based on the water mass properties, ocean velocities and depth. The resulting fluxes are thus highly dependent of the model resolution (horizontal and vertical) and realism of the water masses onto the shelf.

`cn_isfcav_mlt = 'spe'` and `cn_isfpar_mlt = 'spe'` read the melt rate from a file located in the `cn_isfdir` directory. Via these files the user has total control of the fwf forcing. This can be useful if the water masses on the shelf are not realistic, the resolution (horizontal/vertical) is too coarse to have realistic melting or for studies where one needs to control heat and fw input. Note, however, that if the forcing is not consistent with the dynamics below, an unrealistic low water temperature can manifest.

### 8.1.3. Available outputs ( *isfdiags.F90* )

The following outputs are availables via XIOS:

for parametrised cavities :

```
<field id="isftfrz_par"      long_name="freezing point temperature in the parametrization boundary layer" unit="degC"
↪ />
<field id="fwfisf_par"      long_name="Ice shelf melt rate" unit="kg/m2/s" />
<field id="qoceisf_par"     long_name="Ice shelf ocean heat flux" unit="W/m2" />
<field id="qlatistf_par"    long_name="Ice shelf latent heat flux" unit="W/m2" />
<field id="qhcisf_par"     long_name="Ice shelf heat content flux of injected water" unit="W/m2" />
<field id="fwfisf3d_par"    long_name="Ice shelf melt rate" unit="kg/m2/s"
↪ grid_ref="grid_T_3D" />
<field id="qoceisf3d_par"   long_name="Ice shelf ocean heat flux" unit="W/m2"
↪ grid_ref="grid_T_3D" />
<field id="qlatistf3d_par"  long_name="Ice shelf latent heat flux" unit="W/m2"
↪ grid_ref="grid_T_3D" />
<field id="qhcisf3d_par"    long_name="Ice shelf heat content flux of injected water" unit="W/m2"
↪ grid_ref="grid_T_3D" />
<field id="ttbl_par"       long_name="temperature in the parametrisation boundary layer" unit="degC" />
<field id="isfthermal_d_par" long_name="thermal driving of ice shelf melting" unit="degC" />
```

for open cavities :

```

<field id="isftfrz_cav"      long_name="freezing point temperature at ocean/isf interface"      unit="degC"
↪ />
<field id="fwfisf_cav"      long_name="Ice shelf melt rate"                                     unit="kg/m2/s" />
<field id="qoceisf_cav"     long_name="Ice shelf ocean heat flux"                               unit="W/m2" />
<field id="qlatisf_cav"     long_name="Ice shelf latent heat flux"                              unit="W/m2" />
<field id="qhcisf_cav"     long_name="Ice shelf heat content flux of injected water"          unit="W/m2" />
<field id="fwfisf3d_cav"    long_name="Ice shelf melt rate"                                     unit="kg/m2/s"
↪ grid_ref="grid_T_3D" />
<field id="qoceisf3d_cav"   long_name="Ice shelf ocean heat flux"                               unit="W/m2"
↪ grid_ref="grid_T_3D" />
<field id="qlatisf3d_cav"   long_name="Ice shelf latent heat flux"                              unit="W/m2"
↪ grid_ref="grid_T_3D" />
<field id="qhcisf3d_cav"   long_name="Ice shelf heat content flux of injected water"          unit="W/m2"
↪ grid_ref="grid_T_3D" />
<field id="ttbl_cav"        long_name="temperature in Losch tbl"                                unit="degC" />
<field id="isfthermal_d_cav" long_name="thermal driving of ice shelf melting"                   unit="degC" />
<field id="isfgammat"       long_name="Ice shelf heat-transfert velocity"                       unit="m/s" />
<field id="isfgammas"       long_name="Ice shelf salt-transfert velocity"                       unit="m/s" />
<field id="stbl"           long_name="salinity in the Losh tbl"                                 unit="1e-3" />
<field id="utbl"           long_name="zonal current in the Losh tbl at T point"               unit="m/s" />
<field id="vtbl"           long_name="merid current in the Losh tbl at T point"               unit="m/s" />
<field id="isfustar"        long_name="ustar at T point used in ice shelf melting"             unit="m/s" />
<field id="qconisf"        long_name="Conductive heat flux through the ice shelf"             unit="W/m2" />

```

### 8.1.4. Ice sheet coupling ( *isfcpl.F90* )

A coupling interface with any ice sheet model is available. The coupling is done offline through a file exchange at the restart step. At every coupling step, the user needs to build a new domain\_cfg.nc file using the latest ice shelf draft provided by the ice sheet model. The coupling frequency is usually one year. It is not shown yet in the literature that higher coupling frequency between an ocean model and an ice sheet model is needed.

At each restart step, the procedure is the following:

**Step 1** : the ice sheet model sends a new bathymetry and ice shelf draft netcdf file to NEMO.

**Step 2** : a new domcfg.nc file is built using the DOMAINcfg tools.

**Step 3** : NEMO runs for a specific period and outputs the average melt rate over that period.

**Step 4** : the ice sheet model runs using the melt rate outputted in step 3.

**Step 5** : The process is repeated from step 1.

When the coupling with an ice sheet model is activated ( `ln_iscpl= .true.` ), the isf draft is assumed 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 at the restart phase, is very simple. The 6 different possible cases for the tracers and ssh are:

**Thin a cell** : T/S/ssh are unchanged.

**Enlarge a cell** : T/S/ssh are unchanged.

**Dry a cell** : Mask, T/S, U/V and ssh are set to 0.

**Wet a cell** : Mask is set to 1, T/S is extrapolated from neighbour points, ssh is unchanged. If no neighbours exist, then T/S is extrapolated from old top cell values. If no neighbours exist along i, j and k directions (both previous tests failed), T/S/ssh and mask are set to 0.

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

**Wet a column** : Mask is set to 1, T/S/ssh are extrapolated from neighbours. If no neighbour exists, then T/S/ssh and mask set to 0.

The horizontal extrapolation to fill new cells with realistic values is called `nn_drown` times. It means that if the grounding line retreats by more than `nn_drown` cells between 2 coupling steps, the code will be unable to fill all the new wet cells properly and the model is likely to blow up at initialisation. The default number is set up for the MISOMIP idealised experiments.

The coupling method described above will strongly affect the barotropic transport under an ice shelf when the geometry changes. In order to keep the model stable, an adjustment of the dynamics at initialisation, after the coupling step, is needed. The idea behind this is to keep  $\frac{\partial \eta}{\partial t}$  as it should be without a change in geometry at initialisation. This will prevent any strong velocities due to large pressure gradients. To do so, the local horizontal divergence is corrected to match what it was before the change in geometry. This is done before  $\frac{\partial \eta}{\partial t}$

is computed in the first time step.

This coupling procedure is able to take into account grounding line and calving front migration. Note, however, that it is a non-conservative process. This could thus lead to a trend in heat/salt content and volume.

In order to remove this possible trend and keep the conservation level as close to 0 as possible, a simple conservation scheme is available with `ln_isfcpl_cons = .true.`. The heat/salt/vol. gain/loss are diagnosed, along with the location. A correction increment is computed and applied each time step during the model run. The corrective increment is applied into the cells themselves (if it is a wet cell), the neighbouring cells or the closest wet cell (if the cell is now dry).

### 8.1.5. Ice shelf load ( *isfload.F90* )

Ice shelf impose a load on the ocean. The main hypothesis to compute the ice shelf load is that the ice shelf is in an hydrostatic equilibrium. Therefore, beneath an ice shelf, the total pressure is the sum of the pressure due to the ice shelf load and the pressure due to the ocean load. The ice shelf pressure is computed by integrating a reference density profile  $\rho_{isf}$  from the surface to the base of the ice shelf. The local ocean pressure can thenThis can be formulated like this:

$$p(z) = \int_0^{z_{isf}} \rho_{isf} g dz + \int_{z_{isf}}^z \rho g dz \quad (8.6)$$

where  $p(z)$  is the pressure at depth  $z$ ,  $\rho_{isf}$  a reference density profile,  $\rho$  is the water density at depth  $z$  and  $z_{isf}$  is the ice shelf draft.  $\rho_{isf}$  is assume to be the density of a water parcel at `rn_isfload_s` g/kg and `rn_isfload_t` °C that corresponds to the water replaced by the ice shelf. These values are, for the time being, uniform in space and time ( `cn_isfload='uniform'` ). A detailed description of this method is described in Losch (2008).

In a  $z^*$  coordinate framework, the pressure gradient is formulated as suggested by Adcroft and Campin (2004):

$$\nabla_z p(z) = \nabla_{z^*} p(z) + \rho g \nabla_{z^*} z \quad (8.7)$$

The hydrostatic pressure gradient at a given level,  $k$ , (first term in equation 8.7) is computed by adding the pressure gradient due to the ice shelf load (defined as the first term of equation 8.6) to the vertical integral of the in situ density gradient along the model level from the surface to that level.

The only pressure gradient scheme compatible with the under ice shelf seas is ( `ln_hpg_isf=.true.` ). Outside of the ice shelf cavities, this pressure gradient scheme is exactly the same as traditional sco scheme ( `ln_hpg_sco` , subsection 5.4.2).

### 8.1.6. Under ice shelf cavity geometry ( *domisf.F90* )

To produce a simulation with ice shelf cavities opened, the user needs to make sure the bathymetry in the `main_cfg.nc` input file includes ice shelf cavities (and ice shelf draft information). The logical flag `ln_isfcav` in the `DOMAINcfg` namelist controls whether or not the ice shelf cavities are closed. All the options available to define the shape of the under ice shelf cavities are listed in `&namzgr_isf` (namelist F.3) (`DOMAINcfg` only, namelist F.3).

First of all, the tool makes sure that the ice shelf draft ( $h_{isf}$ ) is compatible with the bathymetry (ice shelf draft being defined by `cn_draft` and bathymetry by `cn_bathy` in appendix F). This is done in 3 steps :

First: the position of the grounding line (separation between the floating ice shelf and the grounded ice sheet) is enforced. Where the difference between the bathymetry and the ice shelf draft is smaller than `rn_glhw_min` , the cell are grounded (ie masked). This step is needed in order to take into account possible small mismatches between the ice shelf draft value and the bathymetry value (sources are coming from different grid, different data processes, rounding errors, ...).

Secondly: unrealistically thin ice shelves are corrected and  $h_{isf}$  is set to a minimum value `rn_isfdep_min` . If `rn_isfdep_min` is smaller than the surface level thickness, `rn_isfdep_min` is set to `e3t_1d(1)`.

Finally: The water column thickness `rn_isfhw_min` is enforced. Where the water column thickness is lower than `rn_isfhw_min` , the ice shelf draft is adjusted to match this criterion. If for any reason this adjustment defies the minimum ice shelf draft allowed ( `rn_isfdep_min` ), the cell is masked.



Once all these adjustments are made, the ice shelf draft and bathymetry are compatible. The user can, in addition, also remove the wet one-cell-wide channels under an ice shelf ( `ln_isfchannel` ).

After the definition of the ice shelf draft, the tool defines the top level. The compulsory criterion is that the water column needs at least 2 wet cells in the water column at U- and V-points. To do so, if there is a one-cell-thick water column, the tools adjust the ice shelf draft to fulfil the requirement by digging into the ice shelf draft.

The process is the following:

step 1: The top level is defined in the same way as the bottom level is defined.

step 2: The isolated grid points in the bathymetry are filled (as it is done in a domain without an ice shelf)

step 3: If the water column at a U- or V- point is one wet cell thick, the ice shelf draft is adjusted. So the actual top cell becomes fully open and the new top cell thickness is set to the minimum cell thickness allowed (following the same logic as for the bottom partial cell). This step is iterated 4 times to ensure the condition is fulfilled along the 4 lateral sides of the water column. In the case of a steep slope and a shallow water column, it is likely that 2 adjacent wet water columns under the same ice shelf are disconnected (bathymetry lies above its neighbouring ice shelf draft). The option `ln_isfconnect` allow the tool to force the connection between these 2 cells. Some limits in meters or levels on the digging allowed by the tool are available (respectively, `rn_zisfmax` or `rn_kisfmax` ). This will force the connection of two parts of an ice shelf at the expense of a long vertical pipe to connect the cells at very different levels.

step 5: If after the previous step, the ice shelf draft is shallower than `rn_isfdep_min` , it means that there is an incompatibility and the water column is closed.

In addition to this process, two extra options are available to the user for easthetic or stability reason. Despite careful setting of your ice shelf draft and bathymetry input file as well as setting described in ??, some situations are unavoidable. For exemple, chimneys within the ice shelf draft (ie wet cell surrounded horizontally by land cell) can be present. The circulation in these cells is not resolved. The user can decide to remove those features by activating `ln_isfcheminey` .

An other exemple is, if you setup your ice shelf draft and bathymetry to do ocean/ice sheet coupling, you may decide to fill the whole antarctic with a bathymetry and an ice shelf draft value (ice/bedrock interface depth when grounded). If you do so, the subglacial lakes will show up (Vostock for exemple). Another possibility is with coarse vertical resolution, some ice shelves could be cut in 2 parts: one connected to the main ocean and the other one closed which can be considered as a subglacial sea be the model. Keeping these features can lead to these situations:

- For subglacial lakes in the case of very weak circulation (often the case), the only heat flux is the conductive heat flux through the ice sheet. This will lead to constant freezing until water reaches -20C. This is one of the deficiencies of the 3 equation melt formulation (for details on this formulation, see: [subsection 8.1.1](#)).
- In the case of coupling with an ice sheet model, the ssh in the subglacial lakes and the main ocean could be very different (ssh initial adjustment for exemple), and so if for any reason both a connected at some point, the model is likely to fail.

The namelist option `ln_isfsubgl` allow you to remove these subglacial lakes.

## 8.2. Interaction with icebergs (ICB)

Icebergs are modelled as lagrangian particles in *NEMO* (Marsh et al., 2015). The activation of the model is controlled by `ln_icebergs=.true.` . The physical behaviour of icebergs 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 the implementation thereof in *NEMO*).

### 8.2.1. Iceberg initialisation ( *icbclv.F90* )

#### Iceberg properties

Icebergs are initially spawned into one of ten classes which have specific mass and thickness as described in the `&namberg` ([namelist 8.2](#)) 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).

```

!-----
&namberg      !  iceberg parameters                                (default: OFF)
!-----
ln_icebergs = .false.      ! activate iceberg floats (force =F with "key_agrif")
!
!                               ! restart
cn_icbrst_in  = "restart_ich" ! suffix of iceberg restart name (input)
cn_icbrst_indir = "./"      ! directory from which to read input ocean restarts
cn_icbrst_out  = "restart_ich" ! suffix of ocean restart name (output)
cn_icbrst_outdir = "./"     ! directory from which to read output ocean restarts
!
!                               ! diagnostics:
ln_bergdia   = .true.      ! Calculate budgets
nn_verbose_level = 0      ! Turn on more verbose output if level > 0
!
! nn_verbose_write and nn_sample_rate need to be a multiple of nn_fsbc
nn_verbose_write = 16     ! Timesteps between verbose messages
nn_sample_rate   = 16     ! 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 (safe value is 0.4, see #2581)
!
ln_M2016 = .false. ! use Merino et al. (2016) modification (use of 3d ocean data instead of only sea surface
↪ data)
ln_icb_grd = .false. ! ground icb when icb bottom level hit oce bottom level (need ln_M2016 to be activated)
!
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 8.2.: &namberg



Each iceberg is generated with a pre determined density ( `rn_rho_bergs` ). The iceberg area is determined based on the iceberg thickness, mass and density. The length and width are determined with a pre defined length over width ratio `rn_LoW_ratio` .

### Iceberg generation

Two initialisation schemes are possible to generate the iceberg particules.

`nn_test_icebergs > 0` ( *icbini.F90* ) This scheme is mostly used for developing, testing and debugging. In this scheme, the value of `nn_test_icebergs` represents the class of iceberg to generate (so between 1 and 10). `rn_test_box` defines a box using four numbers representing the corners of the geographical box: lonmin, lonmax, latmin, latmax. One test icebergs is released in every the cells of this box at the beginning of the run. This happens each time the timestep equals `nn_nit000` .

`nn_test_icebergs=-1` ( *icbclv.F90* ) 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 representing ice accumulation rate at each model point in  $km^3$  of ice / y. These should be ocean points adjacent to land where icebergs are known to calve. Most points in this input grid will have a value of 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). How much of the calving rate is then used to fill each iceberg class before releasing an iceberg is controled by the array `rn_distribution` . The sum of the `rn_distribution` array needs to be 1. 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 to generate a new iceberg of its category is reduced accordingly.

### 8.2.2. Iceberg dynamics ( *icbdyn.F90* )

Iceberg dynamics are affected by multiple factors. The Coriolis force and the ocean, atmosphere and sea ice drags ( $\tau_o/\vec{a}/i$ ) are the main drivers of the iceberg dynamics. In addition, the icebergs are further driven by the wave radiation force ( $\vec{F}_r$ ) and a pressure gradient force ( $\vec{F}_p$ ) proportional to the horizontal ssh gradient.

The momentum balance for an iceberg of mass M is given by:

$$M \frac{d\vec{v}}{dt} = -Mf \times \vec{v} + \vec{\tau}_a + \vec{\tau}_o + \vec{\tau}_i + \vec{F}_p + \vec{F}_r \quad (8.8)$$

See [Martin and Adcroft \(2010\)](#) for the full details on each term presented here. By default, the iceberg dynamics are computed using ocean surface variable (ssu, ssv) and the icebergs are not sensible to the bottom topography (only to land), whatever the iceberg draft may be. [Merino et al. \(2016\)](#) developed an option to use vertical profiles of ocean currents instead ( `ln_M2016` ). Full details on the sensitivity to this parameter are described in [Merino et al. \(2016\)](#).

Icebergs are considered as virtual and levitating over the ocean. There are therefore no interactions between icebergs, and therefore there is no limitation regarding the maximum number of icebergs within a grid cell. Furthermore, generally the simulated icebergs are not affected by a shallow bank. If, however, `ln_M2016` is activated, `ln_icb_grd` engages an option to prevent thick icebergs to move across shallow banks (ie shallower than the iceberg draft). In the case of an interaction between one iceberg and the bedrock or the coastline, the iceberg bounces back to its original position. Therefore, when using `ln_M2016` and `ln_icb_grd` , special care is needed in the calving file to be sure that the bathymetry is deep enough at the calving location so as to prevent the accumulation of grounded icebergs. If this is not possible, then the user needs to build a file describing the `maxclass` to prevent NEMO filling the icebergs classes that are too thick for the local bathymetry.

### 8.2.3. Iceberg thermodynamics ( *icbthm.F90* )

The total iceberg melt and the associated latent heat flux are the only fields that feedback to NEMO. The iceberg mass balance is driven by three components (bottom melt ( $M_b$ ), lateral melt ( $M_v$ ) and erosion by surface waves ( $M_e$ )):

$$\rho \frac{d(LWH)}{dt} = \rho(-LWM_b - H(L+W)(M_e + M_v)) \quad (8.9)$$

where L, W and H are respectively the iceberg length, width and thickness. The various melt formulations are from [Bigg et al. \(1997\)](#) and [Gladstone et al. \(2001\)](#). A complete description is available in Appendix A of

Martin and Adcroft (2010). Only the main characteristics are described below.

The wave erosion depends on the sea state ( $S_s$ ) defined by a fit to the Beaufort scale, the ice concentration ( $A_i$ ) and the sea surface temperature ( $T_{surf}$ ).

$$M_e = \frac{1}{12} S_s (1 + \cos(\pi A_i^3)) (T_{surf} + 2) \quad (8.10)$$

By creating a notch at the floatation line by melting, the wave erosion acts to disintegrate the iceberg by breaking the overhanging slab of ice, thus creating small bits of ice which are assumed to propagate with their larger parent and thus delay flux into the ocean. The proportion of the wave erosion that should be represented as bits is controlled by `rn_bits_erosion_fraction`.

The lateral melt ( $M_v$ ) is mainly driven by the buoyant convection along the side of the icebergs. El-Tahan et al. (1987) empirically estimated to be:

$$M_v = 7.62 \times 10^{-3} T_w + 1.29 \times 10^{-3} T_w^2 \quad (8.11)$$

with  $T_w$  being the mean lateral temperature. By default NEMO uses the sea surface temperature for  $T_w$ .

The bottom melt ( $M_b$ ) is mainly driven by the turbulence created by the relative motion of water with respect to the iceberg velocity. Its empirical estimation is :

$$M_b = 0.58 |\vec{v}_{icb} - \vec{v}_w|^{0.8} \frac{T_w - T_{icb}}{L^{0.2}} \quad (8.12)$$

Where  $\vec{v}_{icb}$  is the iceberg velocity,  $\vec{v}_w$  the basal ocean velocity,  $T_w$  the bottom temperature and  $T_{icb}$  the estimated internal temperature of the icebergs (Loiset, 1993,  $-4^\circ C$ ). By default, NEMO uses the ocean surface properties for  $T_w$ . To avoid melting even under freezing condition, the basal melt is set to 0 if the surface temperature is below the surface freezing point.

As for the iceberg dynamics, ocean vertical temperature profiles and ocean basal properties can be used to compute the lateral and basal melt respectively, instead of the surface temperature, by using `ln_M2016`.

If the iceberg width is too small compared to its draft, the iceberg becomes unstable and capsizes. The iceberg thickness and width are then swapped. Such events are detected using the empirical criterion of Weeks and Mellor (1978) :

$$W < \sqrt{0.92D^2 + 58.32D} \quad (8.13)$$

where  $W$  and  $D$  are the iceberg width and draft.

Finally, as mentioned above, iceberg melt rate is the only NEMO input from the iceberg model. The user can, however, disable it by using `ln_passive_mode`. In this case, NEMO will affect the icebergs, the melt will be computed and outputted, but the icebergs themselves will not have any impact on the NEMO solution.

#### 8.2.4. Iceberg mpp ( *icblbc.F90* )

The management of the communication between the subdomains of the iceberg model is very different to the other components of NEMO, as particules (icebergs) are exchanged between subdomains. This is done every iceberg time step, after the iceberg dynamics. The computation of the iceberg thermodynamics and dynamics are done in the inner domain and over the first half of the overlap region (this is necessary as otherwise the interpolation is not possible due to missing iceberg data). For now, NEMO does not make the most of the halos (2 by default now). This leads to the limitation on the cfl to be 0.4 (`rn_speed_limit`) because of interpolation of the ssh at 0.5 in this case. This is planned to be improve upon for the next version.

#### 8.2.5. Iceberg diagnostics ( *icbdia.F90* )

There are three types of diagnostics available: a text diagnostics for the budget, xios diagnostics for the gridded output and the trajectory for diagnostics per iceberg. `ln_bergdia` activates one (or more) type(s) of diagnostics. If set to false, no diagnostics will be outputted.

## Iceberg text diagnostics

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

`nn_verbose_level` takes a value between one and four. It represents an increasing level of verbosity.

`nn_verbose_write` is the number of time steps between each write.

## XIOS diagnostics

The following outputs are available via XIOS:

```

<field_group id="icbvar" domain_ref="grid_T" >
  <field id="berg_melt" long_name="icb melt rate of icebergs" unit="kg/m2/s"
↪ />
  <field id="berg_melt_hcflx" long_name="icb heat flux to ocean due to melting heat content" unit="J/m2/s"
↪ />
  <field id="berg_melt_qlat" long_name="icb heat flux to ocean due to melting latent heat" unit="J/m2/s"
↪ />
  <field id="berg_buoy_melt" long_name="icb buoyancy component of iceberg melt rate" unit="kg/m2/s"
↪ />
  <field id="berg_eros_melt" long_name="icb erosion component of iceberg melt rate" unit="kg/m2/s"
↪ />
  <field id="berg_conv_melt" long_name="icb convective component of iceberg melt rate" unit="kg/m2/s"
↪ />
  <field id="berg_virtual_area" long_name="icb virtual coverage by icebergs" unit="m2"
↪ />
  <field id="bits_src" long_name="icb mass source of bergy bits" unit="kg/m2/s"
↪ />
  <field id="bits_melt" long_name="icb melt rate of bergy bits" unit="kg/m2/s"
↪ />
  <field id="bits_mass" long_name="icb bergy bit density field" unit="kg/m2"
↪ />
  <field id="berg_mass" long_name="icb iceberg density field" unit="kg/m2"
↪ />
  <field id="calving" long_name="icb calving mass input" unit="kg/s"
↪ />
  <field id="berg_floating_melt" long_name="icb melt rate of icebergs + bits" unit="kg/m2/s"
↪ />
  <field id="berg_real_calving" long_name="icb calving into iceberg class" unit="kg/s" axis_ref="icbcla"
↪ />
  <field id="berg_stored_ice" long_name="icb accumulated ice mass by class" unit="kg" axis_ref="icbcla"
↪ />
</field_group>

```

## Iceberg trajectory ( *icbtrj.F90* )

Iceberg trajectories (position as well as iceberg shapes and ocean properties at iceberg location) can also be outputted, and this is enabled by setting `nn_sample_rate > 0`. A non-zero value represents the frequency of timesteps that information is written to the output file. These output files are in NETCDF format. When running with multiple processes, each output file contains only the icebergs in the corresponding sub-domain. Trajectory points are written 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 their trajectory data may be spread across multiple files. To rebuild it, the user can use a specific python tools located in `tools/MISCELLANEOUS/icb_pp.py`.

## 8.3. Land Ice Runoff

Runoff from the ice sheet is produced by the melt of snow and ice at the ice sheet surface and by the basal melt under the grounded area by friction and geothermal heating. It can be classified in two categories: the surface runoff and the sub-glacial runoff.

The surface melt is originated from the melt at the ice sheet surface that runs directly off of the edge of the ice shelf or glacier termini. NEMO is able to represent it as any other surface river runoff. The usage of this module is described in [section 7.9](#).

The subglacial runoff is produced by the melting at the interface bedrock-ice by geothermal heat and friction of the ice, and also by the percolation of the ice sheet surface melt from the surface to the bedrock through moulins. This kind of runoff joins the ocean at the grounding line of the ice sheet (line where ocean, bedrock and the grounded ice meet). Because of the lack of observations of such runoff, subglacial runoff is not represented

in NEMO.

## Lateral Boundary Condition (LBC)

### Table of contents

9.1.	Boundary condition at the continental interface ( <code>rn_shlat</code> ) . . . . .	122
9.2.	Model-domain boundary condition . . . . .	123
9.2.1.	Closed, cyclic ( <code>l_Iperio, l_Jperio</code> ) . . . . .	124
9.2.2.	North-fold ( <code>l_NFold = .true., c_NFtype = 'T'</code> or <code>c_NFtype = 'F'</code> ) . . . . .	124
9.3.	Exchange with neighbouring processes ( <code>lbclnk.F90, lib_mpp.F90</code> ) . . . . .	124
9.4.	Unstructured open boundary conditions (BDY) . . . . .	127
9.4.1.	Namelists . . . . .	128
9.4.2.	Flow relaxation scheme . . . . .	130
9.4.3.	Flather radiation scheme . . . . .	131
9.4.4.	Orlanski radiation scheme . . . . .	131
9.4.5.	Relaxation at the boundary . . . . .	131
9.4.6.	Boundary geometry . . . . .	132
9.4.7.	Input boundary data files . . . . .	132
9.4.8.	Volume correction . . . . .	133
9.4.9.	Tidal harmonic forcing . . . . .	133

### Changes record

Release	Author(s)	Modifications
5.0	<i>Simon Müller and Sébastien Masson</i>	<i>Review and general revision</i>
4.2	<i>Simon Müller</i>	<i>Minor update of subsection 9.4.9</i>
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.
/
    
```

 namelist 9.1.: `&namlbc`

Four types of lateral boundary conditions (LBC) can be specified or configured: boundary conditions at the continental interface, at the boundaries of the model domain, between sub-domains defined for massively parallel processing (MPP), and at open boundaries (BDY) for regional configurations.

## 9.1. Boundary condition at the continental interface ( `rn_shlat` )

Options are defined through the `&namlbc` (namelist 9.1) 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 the code is greatly improved when computational loops work across whole arrays and boundary conditions are applied through multiplication with a mask array (a mask array contains elements of 1 at ocean locations and 0 elsewhere), rather than by specific computations before or after each computational loop. The simple multiplication of a variable by the relevant mask ensures that it remains zero over land areas; since the majority of the boundary conditions implement a zero flux across solid boundaries, they can be simply applied through multiplication with the mask array associated with the grid on which the flux is evaluated. For example, the diffusive heat flux in the  $i$ -direction is evaluated at  $u$ -points. Evaluating this quantity as,

$$\frac{A^{lT}}{e_1} \frac{\partial T}{\partial i} \equiv \frac{A_u^{lT}}{e_{1u}} \delta_{i+1/2} [T] \text{ mask}_u$$

(where  $\text{mask}_u$  corresponds to the mask array at a  $u$ -point) ensures that the heat flux is zero inside topographic features and at their boundaries, since  $\text{mask}_u$  is zero at solid boundaries, which in this case are defined at  $u$ -points (the normal velocity  $u$  also remains zero at such boundaries) (figure 9.1).

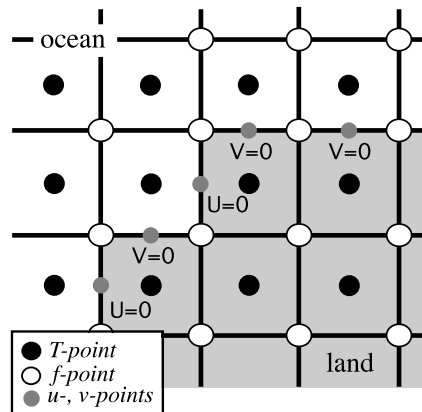


Figure 9.1.: Lateral boundary (thick line) at a T-level; the velocity normal to the boundary is set to zero.

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 at  $f$ -points, and normal velocity points are located between two  $f$ -point neighbours (figure 9.1). The boundary condition on the normal velocity (no flux through solid boundaries) can thus be easily implemented using the mask arrays. 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  $\text{mask}_f$  array along the coastline and bottom topography is set equal to this parameter). These are:

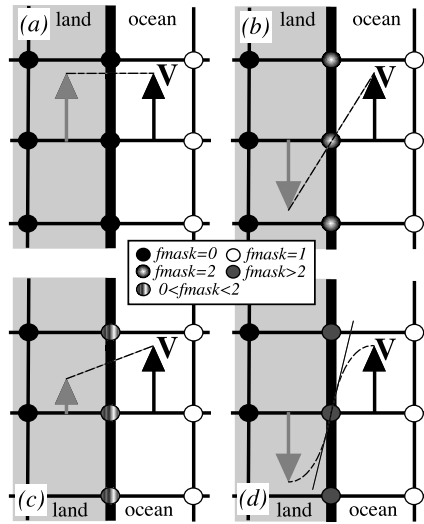


Figure 9.2.: Lateral boundary conditions (a) free-slip ( $rn\_shlat=0$ ); (b) no-slip ( $rn\_shlat=2$ ); (c) "partial" free-slip ( $0 < rn\_shlat < 2$ ) and (d) "strong" no-slip ( $2 < rn\_shlat$ ). Implied "ghost" velocity inside land area is display in grey.

### free-slip boundary condition ( $rn\_shlat=0$ )

the tangential velocity at the boundary is equal to the offshore velocity, *i.e.* the normal derivative of the tangential velocity is zero at the coast, so the vorticity array,  $mask_f$ , is set to zero inside and at continental boundaries (figure 9.2-a).

### no-slip boundary condition ( $rn\_shlat=2$ )

the tangential velocity vanishes at continental boundaries. 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 neighbouring cross-boundary velocity grid-points were of the same magnitude and opposite direction (figure 9.2-b). Therefore, the vorticity along the boundary is given by:

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

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 as:

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

### "partial" free-slip boundary condition ( $0 < rn\_shlat < 2$ )

the tangential velocity at the coastline is smaller than the offshore velocity, *i.e.* there is a lateral friction but not strong enough to make the tangential velocity at the coast vanish (figure 9.2-c). This can be selected by providing a value of  $mask_f$  between 0 and 2.

### "strong" no-slip boundary condition ( $2 < rn\_shlat$ )

the viscous boundary layer is assumed to be smaller than half the grid size (figure 9.2-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 the 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.

## 9.2. Model-domain boundary condition

Several options of global model-domain boundary conditions are available: closed, cyclic east-west, cyclic north-south, cyclic east-west and a north fold, closed and a north fold, or bi-cyclic east-west and north-south. The north-fold boundary condition is associated with the 3-pole ORCA meshes and is available in two variants. The application of these boundary conditions is carried out by calling routine `1bc_1nk` (module `lbclnk.F90`).



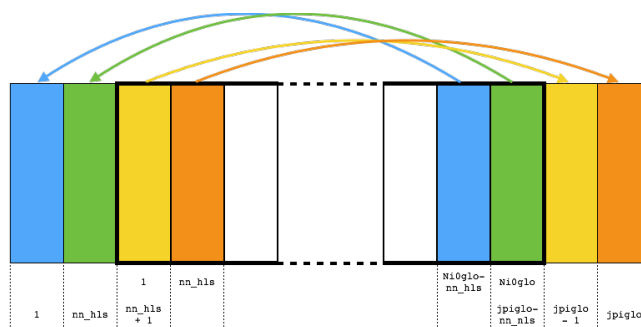


Figure 9.3.: Setting of east-west cyclic boundary conditions for `nm_hls=2`. The original global domain (without halos) is delimited by the bold lines.

### 9.2.1. Closed, cyclic (`l_Iperio,l_Jperio`)

The choice of closed or cyclic model domain boundary condition is controlled by setting the internal code variables `l_Iperio,l_Jperio` to `.true.` or `.false.`. The way these variables are defined will differ according to the value of `ln_read_cfg` parameter in namelist `&namcfg` (namelist 17.1), whose usage is detailed in section 3.2. If `ln_read_cfg=.false.`, the user can define `l_Iperio,l_Jperio` in routine `usrdef_nam` (module `usrdef_nam.F90`). If `ln_read_cfg=.true.`, `l_Iperio,l_Jperio` will be defined according to the values of the global attributes (`Iperio,Jperio = 0` or `1`) in the NetCDF domain configuration file referred to by the `cn_domcfg` parameter in namelist `&namcfg` (namelist 17.1).

**For a fully closed boundary (`l_Iperio = .false.,l_Jperio = .false.`)**, solid walls are imposed at all four model-domain boundaries; the first and last rows and columns of the domain must be set to zero and will be forced to 0 if needed.

**For a cyclic east-west boundary (`l_Iperio = .true.,l_Jperio = .false.`)**, the first and last rows are set to zero (closed); the first `nm_hls` columns (left halo) are defined with the last `nm_hls` columns of the original global domain (without halos); the last `nm_hls` columns (right halo) are defined with the first `nm_hls` columns of the original global domain (without halos); flows out of the eastern (western) boundary re-enter through the western (eastern) boundary.

**For a cyclic north-south boundary (`l_Iperio = .false.,l_Jperio = .true.`)**, the first and last columns are set to zero (closed); analogous to the east-west periodic option, the `nm_hls` first (last) rows are replicated at the last (first) rows of the original global domain (without halos); flows out of the northern (southern) boundary of the domain re-enter through the southern (northern) boundary. Note that the cyclic north-south boundary requires the f-plan approximation so that `f`, the coriolis parameter, remains constant in `j`-direction.

**The bi-cyclic east-west and north-south boundary (`l_Iperio = .true.,l_Jperio = .true.`)** combines the two cyclic options.

### 9.2.2. North-fold (`l_NFold = .true., c_NFtype = 'T'` or `c_NFtype = 'F'`)

The north fold boundary condition has been introduced in order to handle the northern boundary of a three-polar ORCA grid. When mapping these grids onto a sphere, the last grid row of the original global domain (without halo) folds onto itself to connect the model domain between the two poles in the northern hemisphere (figure 17.1), and thus requires a specific treatment to implement such folding at the northern edge of the domain. Further information can be found in appendix E and in the `lbcnfd.F90` module, which contains the subroutine that applies the north-fold boundary condition. The `ln_nnogather` parameter in namelist `&nammpp` (namelist 9.2) must be set to `.true.` to get the best performances. It can be set to `.false.` for debugging purposes.

## 9.3. Exchange with neighbouring processes ( *lbclnk.F90* , *lib\_mpp.F90* )

For massively parallel processing (MPP), a horizontal domain decomposition method is used, see figure 9.4. The basic idea of the method is to split the computational domain of a large numerical experiment horizontally into several smaller subdomains and solve the set of equations by addressing independent local problems. A number of processes, each with its own local memory, that can communicate with each other, are utilised

```

-----
&nammpp      !  Massively Parallel Processing
-----
ln_listonly = .false. ! do nothing else than listing the best domain decompositions (with land domains suppression)
!             ! if T: the largest number of cores tested is defined by max(mppsize, jpni*jpnj)
ln_nnogather = .true. ! activate code to avoid mpi_allgather use at the northfold
ln_mppdelay  = .true. ! activate delayed global communications to go faster
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
nn_hls       = 2      ! halo width (applies to both rows and columns)
nn_comm      = 1      ! comm choice
/
    
```

namelist 9.2.: &nammpp

to compute the model equations in parallel, with each process carrying out the computation restricted to an individual subdomain. The present implementation is largely inspired by Guyon's work [Guyon 1995].

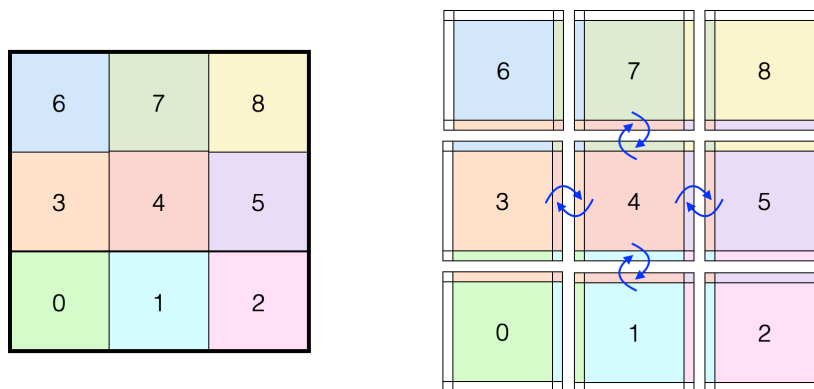


Figure 9.4.: Horizontal domain decomposition in  $3 \times 3$  subdomains. The thick line on the left panel delimits the original domain (without halos). Subdomains numbering starts at 0 from the bottom-left subdomain. Communications of the subdomain 4 with its neighbours are represented by the blue arrows.

Each subdomain consists of an inner region and a boundary. The boundary, also called halo, overlaps with neighbouring subdomains as show in the left panel of figure 9.4. The halo consist of `nn_hls` (namelist `&nammpp` (namelist 9.2) ) rows or columns at each of the sides of the subdomain. It must be set to 1 in *NEMO* version 4.2 and to 2 in *NEMO* version 5.0.

The `lbc_lnk` routine (found in `lbc_lnk.F90` module) is used to fill the halo either by communicating with neighbouring subdomains or locally by applying a cyclic or closed boundary condition. To carry out exchanges with neighbours, the Message Passing Interface (MPI; ) standard is utilised. When communicating, each process sends to the processors associated with its neighbouring subdomains an update of the values of the points that correspond to the interior part of the overlap (*i.e.* the `nn_hls` innermost of the  $2 * \text{nn\_hls}$  rows and columns of the overlap, see blue arrows in figure 9.4). The output file `communication_report.txt` provides an overview of the number of such exchanges made by individual routines during each time step.

Since *NEMO* version 4.2, the MPP approach is activated by default. It can be deactivated (*e.g.* if no MPI implementation is available on the target computer) by defining `key_mpi_off`, and for compatibility with the MPI version 2 standard, `key_mpi2` can be defined. With the *NEMO* version 4.2 release, a new communication strategy, which preserves performance efficiency by reducing communication time, has been introduced by using the neighbourhood collective communications available with the MPI version 3 standard. It provides a way to use sub-communicators to perform collective communications among neighbourhoods: a single MPI message needs to be built for all neighbours before calling the collective operation, instead of four different messages. The new communication approach can be selected by setting `nn_comm=2` in the `&nammpp` (namelist 9.2) namelist record. By default, `nn_comm=1`, which activates the original point-to-point communication of *NEMO* which has been further optimized in *NEMO* version 4.2. Note that other communication strategies are available in the `BENCH` test case. These communication strategies are using non-blocking point-to-point communications

with different approaches to test whether communications have been received: `MPI_Iprobe` (`nn_comm=30`), `MPI_Waitany` (`nn_comm=31`), `MPI_Waitall` (`nn_comm=32`).

In *NEMO* the decomposition of the model domain results in a regular horizontal grid of subdomains, which can be manually configured with the parameters `jpnj` and `jpmj` defined in the `&nammpp` ([namelist 9.2](#)) namelist record to select the number of columns along the i-axis and the number of rows along the j-axis, respectively. If both `jpmj` and `jpnj` are less than 1 (default), they will be automatically set during model initialisation to result in an optimal domain decomposition (see below and ([Irrmann et al., 2022](#))) into a number of subdomains that corresponds to the number of MPI processes allocated to *NEMO* when the model is launched (*i.e.* `mpirun -np <n> ./nemo` will automatically result in a domain decomposition into `<n>` subdomains).

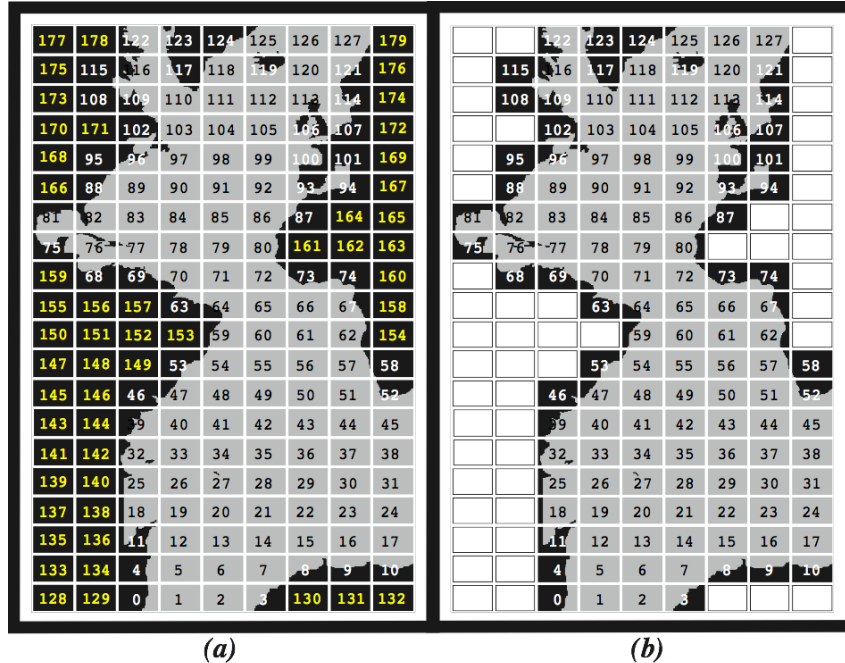


Figure 9.5.: Example of Atlantic domain defined for the CLIPPER projet. Initial grid is composed of 773 x 1236 horizontal points. (a) the domain is split onto 9 *times* 20 subdomains (`jpmj=9`, `jpnj=20`). Subdomains with ocean points are numbered first starting from bottom-left. The 52 subdomains that are land areas are next numbered starting also from bottom-left (in yellow). (b) The 52 subdomains are eliminated (white rectangles) and the resulting number of processors really used during the computation is 128. Note that the subdomains with ocean points have the same number in both cases.

The *NEMO* model computes equation terms with the help of mask arrays (0 on land points and 1 on sea points), see [section 9.1](#). It is therefore possible that an MPI subdomain contains only land points, see [figure 9.5](#). To save ressources, the model initialisation attempts to suppress as many land subdomains as possible from the computational domain. For example if  $N_{mpi}$  processes are allocated to *NEMO* the domain decomposition results in the equality

$$N_{mpi} = jpmj \times jpnj - N_{land} + N_{useless}$$

where  $N_{land}$  is the total number of land subdomains in the domain decomposition into `jpmj` by `jpnj` subdomains.  $N_{useless}$  is the number of land subdomains that are kept in the computational domain in order to ensure that all  $N_{mpi}$  MPI processes are allocated a computational task. The values of  $N_{mpi}$ , `jpmj`, `jpnj`,  $N_{land}$  and  $N_{useless}$  are reported in the output file `ocean.output`.  $N_{useless}$  must, of course, be as small as possible to reduce a wasting of ressources, and therefore a warning is issued in `ocean.output` if  $N_{useless}$  is not zero. Note that a non-zero value of  $N_{useless}$  is usually required when using AGRIF as, up to now, the parent and all child grids must make use of all  $N_{mpi}$  processes.

If the domain decomposition is performed automatically, the variant chosen by the model will both minimise the horizontal subdomain size (defined as  $max_{all\ domains}(subdomainsize)$ ) and maximize the number of eliminated land subdomains. This means that no other decomposition will use fewer processes than  $(jpmj \times jpnj - N_{land})$  while having a smaller subdomain size. In order to tune  $N_{mpi}$  (minimize  $N_{useless}$ ), the model can initially be run with `ln_listonly` activated: the model will start the initialisation phase, print a list of optimal decompositions ( $N_{mpi}$ , `jpmj` and `jpnj`) to `ocean.output`, and then halt. The maximum value of  $N_{mpi}$  tested in this list is  $max(N_{MPI\_tasks}, jpmj \times jpnj)$ . For example, *NEMO* can be run on a single process with `ln_listonly=.true.`, `jpmj=1000` and `jpnj=10` to obtain a list of optimal domain decompositions for the number of processes ranging from 1 to about 10000.

The subdomains are numbered from 0 to  $N_{mpi} - 1$ : first, subdomains containing ocean points are numbered from 0 to  $jpmj * jpnj - N_{land} - 1$ , starting with the bottom-left-most subdomain, see [figure 9.5](#); then, the

remaining  $N_{useless}$  land subdomains are numbered, starting from the bottom-left-most land subdomain (yellow numbers in figure 9.5a). This implies that, for given values of  $jpni$  and  $jpj$ , 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 remain undefined by default, but to avoid missing-number entries in output files, `ln_mskland` can be activated. Note that it may be beneficial to not eliminate land processes when creating the `meshmask` output file (*i.e.* when setting a non-zero value to `nn_msh`).

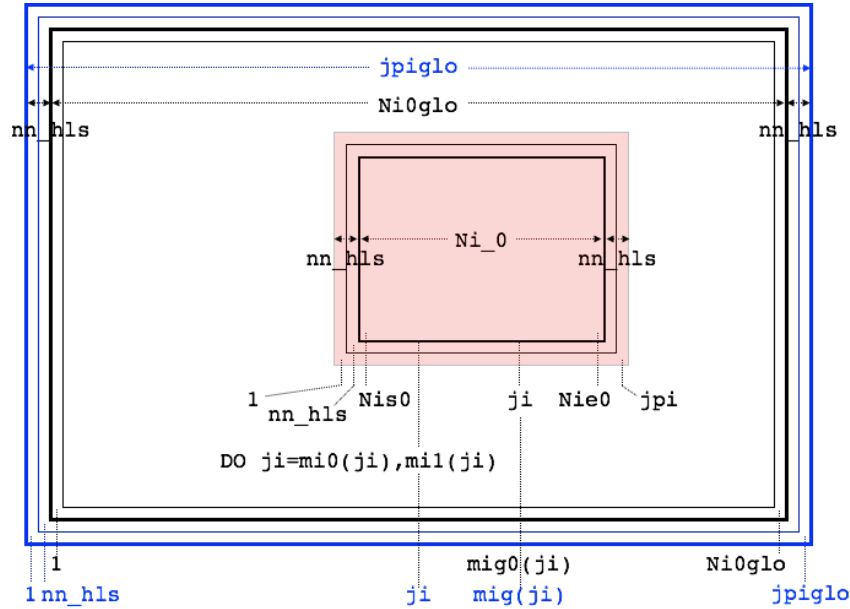


Figure 9.6.: Positioning of a sub-domain when massively parallel processing is used

With the exception of message passing or synchronisation, each process runs independently and accesses solely its own local memory. For this reason, the main model dimensions,  $jp_i$ ,  $jp_j$ , and  $jp_k$ , correspond to that of the local subdomain. As detailed in Irrmann et al. (2022), the value of  $jp_i$  and  $jp_j$  can differ between subdomains. Note that if the configuration requires north folding (`1_NFold = .True.`), the  $jp_j$  of the processes involved in the folding are reduced in order to compensate for the extra cost of the north folding operation.

As shown in figure 9.6, the extents of the subdomain,  $jp_i$  and  $jp_j$ , account for both the inner domain and the halo.  $Nis0$  ( $Njs0$ ) and  $Nie0$  ( $Nje0$ ) are used to specify the start and end of the inner domain along the i-axis (j-axis).

Several variables are available to convert between indices of the local subdomain and the global domain (with or without halos). The dimensions of the whole model domain with a halo are referred to as  $jpiglo$ ,  $jpglo$ , and  $jpgk$ .  $Ni0glo$  and  $Ni0glo$  correspond to the actual domain size as seen in input and output files, without any halo. The 1-d arrays  $mig(1 : jpi)$  and  $mjpg(1 : jpj)$ , defined in the `init_locglo` routine (module `mppini.F90`), can be used to convert local indices to indices of the global domain with halos. For example, an element of  $T_l$ , a local array (subdomain) corresponds to an element of  $T_g$ , a global array (model domain with halos) through the relationship:

$$T_g(mig(i), mjpg(j), k) = T_l(i, j, k),$$

with  $1 \leq i \leq jpi$ ,  $1 \leq j \leq jpj$ , and  $1 \leq k \leq jpgk$ . Similarly, the 1-d arrays  $mig0(1 : jpi)$  and  $mjpg0(1 : jpj)$  can be used to convert local subdomain indices to indices of the global domain without halos. The 1-d arrays  $mi0(1 : jpiglo)$ ,  $mi1(1 : jpiglo)$ ,  $mj0(1 : jpglo)$ , and  $mj1(1 : jpglo)$  have the reverse purpose and can be used as loop loop bounds formulated in terms of global domain indices (see module `dtastd.F90` for examples).

## 9.4. Unstructured open boundary conditions (BDY)

Options are defined through the `&nambdy` (namelist 9.3) and `&nambdy_dta` (namelist 9.4) namelist variables. For regional configurations, the BDY module is the core implementation for the application of open boundary conditions to the ocean temperature, salinity, and barotropic-baroclinic velocities, as well as to ice-snow concentration, thicknesses, temperatures, salinity, and melt-pond 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 open-boundary location is completely flexible

```

!-----
&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 boundaries 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_volctl    = 1         ! = 0, the total water flux across open boundaries is zero
/

```

namelist 9.3.: &amp;nambdy

and facilitates setups from regular boundaries to irregular contours (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 be compatible with the current version. See [subsection 9.4.7](#) for details.

### 9.4.1. 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 with namelist records (`ln_coords_file=.false.`), in which case an additional record must be provided for each set (see the example [namelist records 9.5](#)); or it can be read in from a file (`ln_coords_file=.true.`), in which case a file with the name `cn_coords_file` (the default name, “*coordinates.bdy.nc*”, will be used below to refer to such input files) must be provided for each set (the “*coordinates.bdy*” file is analagous to the usual *NEMO* “*coordinates.nc*” file). For more details of the definition of the boundary geometry and coordinate files see [section subsection 9.4.6](#).

For each boundary set a boundary condition has to be chosen for different field categories: the barotropic solution, “u2d” (sea-surface height and barotropic velocities); the baroclinic velocities, “u3d”; the active tracers \*, “tra”; and sea-ice, “ice”. For each category of variables an algorithm and the boundary data have to be selected (set with `cn_u2d` and `nn_u2d_dta`, `cn_u3d` and `nn_u3d_dta`, `cn_tra` and `nn_tra_dta`, and `cn_ice` and `nn_ice_dta`, respectively).

The choice of algorithm is currently:

**'none'**

no boundary condition is applied, the solution will “see” land points around the edge of the domain;

**'specified'**

the specified boundary condition is applied for baroclinic velocity and tracer variables;

**'neumann'**

the values at the boundary are duplicated (no gradient) for baroclinic velocity and tracer variables;

**'frs'**

the Flow Relaxation Scheme (FRS) for all variables;

**'orlanski'**

the Orlandski radiation scheme (fully oblique) for barotropic, baroclinic and tracer variables;

\*The current version of the BDY module does not extend to passive tracers.

```

!-----
&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
! automatically defined to T if the number of vertical levels in bdy_dta /= jpk
ln_full_vel = .false.    ! 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

!-----
! rotation ! land/sea mask ! frequency (hours) ! variable ! time interp.! clim ! 'yearly' ! weights filename !
!-----
! pairing ! filename ! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' !
bn_ssh        = 'amm12_bdyT_u2d' , 24. , 'sossheig' , .true. , .false. , 'daily' , '' ,
bn_u2d        = 'amm12_bdyU_u2d' , 24. , 'vobtcrtx' , .true. , .false. , 'daily' , '' ,
bn_v2d        = 'amm12_bdyV_u2d' , 24. , 'vobtcrtx' , .true. , .false. , 'daily' , '' ,
bn_u3d        = 'amm12_bdyU_u3d' , 24. , 'vozocrtx' , .true. , .false. , 'daily' , '' ,
bn_v3d        = 'amm12_bdyV_u3d' , 24. , 'vomecrtx' , .true. , .false. , 'daily' , '' ,
bn_tem        = 'amm12_bdyT_tra' , 24. , 'votemper' , .true. , .false. , 'daily' , '' ,
bn_sal        = 'amm12_bdyT_tra' , 24. , 'vosaline' , .true. , .false. , 'daily' , '' ,
!* for si3
bn_a_i        = 'amm12_bdyT_ice' , 24. , 'siconc' , .true. , .false. , 'daily' , '' ,
bn_h_i        = 'amm12_bdyT_ice' , 24. , 'sithic' , .true. , .false. , 'daily' , '' ,
bn_h_s        = 'amm12_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' , '' ,
bn_hil        = 'NOT USED' , 24. , 'sihlid' , .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 --
rn_ice_hlid   = 0.0      ! -- pond lid depth --
/

```

namelist 9.4.: &amp;nambdy\_dta

```

!-----
&nambdy          ! unstructured open boundaries                (default: OFF)
!-----
ln_bdy          = .true.   ! Use unstructured open boundaries
nb_bdy          = 2        ! number of open boundary sets
ln_coords_file  = F F      ! =T : read bdy coordinates from file
nn_rimwidth     = 1 1     ! width of the relaxation zone
/

! BDY segments
&nambdy_index
ctypebdy = 'S'           ! South boundary segment
nbdyind  = 1             ! Index along the constant dimension
nbdybeg  = 2             ! Start point along the varying dimension
nbdyend  = 200          ! Final point along the varying dimension
/

&nambdy_index
ctypebdy = 'E'           ! East boundary segment
nbdyind  = -1           ! Automatic boundary definition: if -1 set boundary to whole side of model domain.
/

```

namelist 9.5.: &amp;nambdy &amp; nambdy\_index



'orlanski\_npo'

the Orlandi radiation scheme (NPO) for barotropic, baroclinic, and tracer variables; and

'flather'

the Flather radiation scheme for the barotropic variables.

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` ). If external boundary data is required then the `&nambdy_dta` (namelist 9.4) namelist must be defined. One `&nambdy_dta` (namelist 9.4) namelist is required for each boundary set, adopting the same order of indexes in which the boundary sets are defined in `&nambdy` (namelist 9.3) . The boundary data is read in from directory `cn_dir` using module `fldread.F90` (see subsection 7.2.1), and entries for each required variable in the `&nambdy_dta` (namelist 9.4) namelist record can be formulated in the corresponding format (filename, file and data frequency, time-interpolation, climatological (time-cyclic) data flag). For sea-ice salinity, temperatures and melt ponds, constant values instead of input data can be selected by specifying the filename as 'NOT USED'.

In case the "u3d" velocity data contains the total velocity (ie, baroclinic and barotropic velocity), the BDY code can derive baroclinic and barotropic velocities by setting `ln_full_vel=.true.` For the barotropic solution there is also the option to use tidal harmonic forcing without ( `nn_dyn2d_dta=2` ) or in addition to external tidal forcing 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 from a file or set to a constant value ( `rn_ice_sal` , `rn_ice_tem` , `rn_ice_apnd` , `rn_ice_hpnd` , and `rn_ice_hlid` ); the ice age is constant and defined by `rn_ice_age` .

There is also an option to vertically interpolate the open boundary data onto the native grid at run-time ( `ln_zinterp=.true.` ). For this to be successful the additional depth and scale-factor variables `gdept`, `gdepu`, `gdepv`, `e3t`, `e3u`, and `e3v` are required to be present in the lateral boundary files, and the scale factors are used to adjustment to velocity fields due to differences in the total water depths between the two vertical grids.

The AMM12 reference configuration, `crgs/AMM12/EXPREF/namelist_cfg`, provides an example with one boundary set of external daily data provided in individual files (from a large-scale model): FRS conditions are applied to temperature and salinity, and Flather conditions to the barotropic variables; no condition is specified for the baroclinic velocities and sea-ice; tidal harmonic forcing is also used.

### 9.4.2. 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  $\Phi$

$$\Phi(d) = \alpha(d)\Phi_e(d) + (1 - \alpha(d))\Phi_m(d) \quad d = 1, N$$

where  $\Phi_m$  is the model solution and  $\Phi_e$  is the specified external field,  $d$  gives the discrete distance from the model boundary and  $\alpha$  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  $\Phi$  of the form:

$$-\frac{1}{\tau}(\Phi - \Phi_e)$$

where the relaxation time scale  $\tau$  is given by a function of  $\alpha$  and the model time step  $\Delta t$ :

$$\tau = \frac{1 - \alpha}{\alpha} \Delta t$$

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 reflections of outgoing signals from the model boundary.

The function  $\alpha$  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 in the range from 8 to 10 (default).



### 9.4.3. 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

$$U = U_e + \frac{c}{h}(\eta - \eta_e), \quad (9.1)$$

where  $U$  is the depth-mean velocity normal to the boundary and  $\eta$  is the sea surface height, both from the model. The subscript  $e$  indicates the corresponding 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 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 difference in [equation 9.1](#) is a spatial gradient across the model boundary, so that  $\eta_e$  is defined on  $T$ -grid points with `nbr=1` (adjacent to the boundary, see [subsection 9.4.6](#)) and  $\eta$  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.

### 9.4.4. 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:

$$\frac{\partial \phi}{\partial t} + c_x \frac{\partial \phi}{\partial x} + c_y \frac{\partial \phi}{\partial y} = -\frac{1}{\tau}(\phi - \phi^{ext}) \quad (9.2)$$

where  $\phi$  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:

$$c_x = -\frac{\partial \phi}{\partial t} \frac{\partial \phi / \partial x}{(\partial \phi / \partial x)^2 + (\partial \phi / \partial y)^2} \quad (9.3)$$

$$c_y = -\frac{\partial \phi}{\partial t} \frac{\partial \phi / \partial y}{(\partial \phi / \partial x)^2 + (\partial \phi / \partial y)^2} \quad (9.4)$$

(As noted by MMS, this is a circular diagnosis of the phase speeds which only makes sense on a discrete grid). Equation ([equation 9.2](#)) is defined adaptively depending on the sign of the phase velocity normal to the boundary  $c_x$ . For  $c_x$  outward, we have

$$\tau = \tau_{out} \quad (9.5)$$

For  $c_x$  inward, the radiation equation is not applied:

$$\tau = \tau_{in} ; c_x = c_y = 0 \quad (9.6)$$

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 9.4.5](#) for details 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 9.2](#)), but including this term in the denominator of equation ([equation 9.3](#)). Both versions of the scheme are options in BDY. Equations ([equation 9.2](#)) - ([equation 9.6](#)) correspond to equations (13) - (15) and (2) - (3) in MMS.

### 9.4.5. Relaxation at the boundary

In addition to a specific boundary condition specified as `cn_tra` and `cn_dyn3d`, relaxation on baroclinic velocities and tracers are available. This option can be selected with the namelist parameters `ln_tra_dmp` and `ln_dyn3d_dmp` for each boundary set.

The relaxation time scale value (`rn_time_dmp` and `rn_time_dmp_out`,  $\tau$ ) are defined at the boundaries itself. This time scale ( $\alpha$ ) 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.

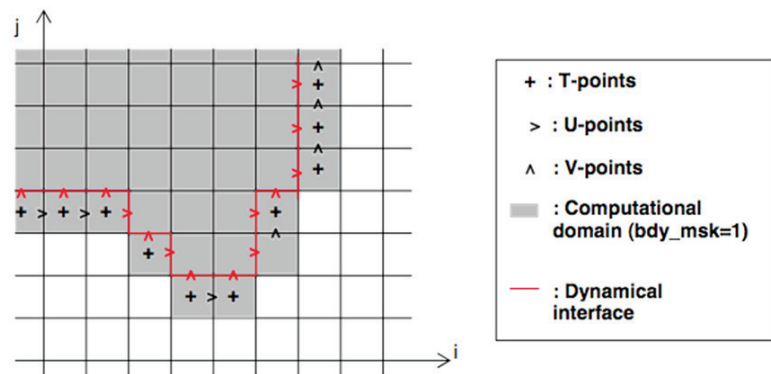


Figure 9.7.: Example of the geometry of unstructured open boundaries

### 9.4.6. 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 9.7 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 record defines straight-line segments for northern, eastern, southern or western boundaries (`ctypebdy` is set to 'N', 'E', 'S', or 'W', respectively), and contains the three values `nbdybeg`, `nbdyend`, and `nbdyind` that specify the first and final index on the axis along, as well as the position on the axis perpendicular to, the segment, respectively. If parameter `nbdyind` is set to -1, the code will automatically define the boundary segment to span the respective domain extent. 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 alternatively be defined through a “*coordinates.bdy.nc*” file. figure 9.8 provides an example of the header information of such a file. The file contains the index arrays for each of the  $T$ -,  $u$ - and  $v$ -grids; prefixes `nbi`, `nbj`, and `nbr` identify the  $i$ -,  $j$ -, and boundary-width-indices relevant for the global domain (the former two are internally converted to local domain indices), respectively; these arrays must be in order of increasing `nbr`. Such files are also used to generate external boundary data via interpolation, and therefore it also contains the latitudes and longitudes of each point as shown; horizontal interpolation is, however, not relevant when running the model.

For some choices of an irregular boundary, the model domain may contain ocean areas 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. Parameter `ln_mask_file` in the `&nambdy` (namelist 9.3) namelist record can be used to activate the input of a mask file of name `cn_mask_file`. Only one mask file can be used, even if multiple boundary sets are defined.

### 9.4.7. 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 have three dimensions: time;  $xb$ , the index of the boundary data point in the horizontal; and  $yb$ , a dummy dimension of extent one that enables the file to be read by the standard *NEMO* I/O routines. 3D fields also have a depth dimension.

From *NEMO* version 3.4 there are additional restrictions on the order in which 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`;
2. all the data for a particular boundary set must be in the same order.

These restrictions mean that data files previously used with a *NEMO* model version lower than 3.4 may not work from version 3.4 onwards.

```

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 gphiv(yb, xbV) ;
        gphiv:units = "degrees_north" ;

// global attributes:
        :file_name = "med12.obc.coordinates.reorder.nc" ;
        :rimwidth = 9 ;
        :NCO = "3.9.9" ;
}

```

Figure 9.8.: Example of the header of a “*coordinates.bdy.nc*” file

#### 9.4.8. Volume correction

A volume correction can be applied to ensure that the total volume of a regional model remains constant. This option can be activated with parameter `ln_vol` of namelist `&nambdy` (namelist 9.3); `ln_vol=.false.` (default) indicates that this option is not used. When active (`ln_vol=.true.`), two options are available with parameter `nn_volctl`:

`nn_volctl=0`

the normal barotropic velocities around the boundary are adjusted at each timestep so that the integrated volume flow through the boundary is zero;

`nn_volctl=1`

in addition to the integrated volume flow through the boundary, the change due to the surface freshwater flux is taken into account to adjust the normal barotropic velocities around the boundary to achieve an integrated volume flow of zero at each timestep.

If more than one boundary set is used, the volume correction is applied to all boundaries simultaneously.

#### 9.4.9. Tidal harmonic forcing

```

!-----
&nambdy_tide  ! tidal forcing at open boundaries                (default: OFF)
!-----
filtide      = 'bdydt/amm12_bdytide_'  ! file name root of tidal forcing files
ln_bdytide_2ddta = .false.             !
/

```

namelist 9.6.: &nambdy\_tide

Tidal forcing at open boundaries requires the activation of surface tides (*i.e.*, in `&nam_tide` (namelist 7.8) , `ln_tide=.true.` with the active tidal constituents listed in the `sn_tide_cnames` array; see section 7.8). The specific options related to the reading in of the complex harmonic amplitudes of elevation (SSH) and barotropic velocity components (u,v) at the open boundaries are defined through the `&nambdy_tide` (namelist 9.6) 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=.true.` or `ln_bdytide_2ddta=.false.` , respectively. In either case, the real and imaginary parts of SSH, u, and v amplitudes associated with each activated tidal constituent `<constituent>` have to be provided separately as fields in input files with names based on `filtide=<input>` . When two-dimensional data is used:

**file `<input>_grid_T.nc`**

is expected to contain variables `<constituent>_z1` and `<constituent>_z2` with the real and imaginary parts of SSH, respectively;

**file `<input>_grid_U.nc`**

variables `<constituent>_u1` and `<constituent>_u2` with the real and imaginary parts of u, respectively; and

**file `<input>_grid_V.nc`**

variables `<constituent>_v1` and `<constituent>_v2` with the real and imaginary parts of v, respectively.

When data along open boundary segments is used:

**file `<input><constituent>_grid_T.nc`**

is expected to contain variables `z1` and `z2` (real and imaginary part of SSH);

**file `<input><constituent>_grid_U.nc`**

variables `u1` and `u2` (real and imaginary part of u); and

**file `<input><constituent>_grid_V.nc`**

variables `v1` and `v2` (real and imaginary part of v).

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

10.1. Lateral mixing operators . . . . .	136
10.2. Direction of lateral mixing ( <i>ldfslp.F90</i> ) . . . . .	136
10.2.1. Slopes for geopotential mixing in the <i>s</i> -coordinate . . . . .	136
10.2.2. Slopes for tracer iso-neutral mixing . . . . .	137
10.2.3. Slopes for momentum iso-neutral mixing . . . . .	138
10.3. Lateral mixing coefficient ( <i>nn_aht_ijk_t</i> & <i>nn_ahm_ijk_t</i> ) . . . . .	139
10.3.1. Mixing coefficients read from file (= -20, -30) . . . . .	139
10.3.2. Constant mixing coefficients (=0) . . . . .	140
10.3.3. Vertically varying mixing coefficients (=10) . . . . .	140
10.3.4. Mesh size dependent mixing coefficients (=20) . . . . .	140
10.3.5. Mesh size and depth dependent mixing coefficients (=30) . . . . .	140
10.3.6. Eddy parameterization ( <i>nn_aht_ijk_t</i> =21) . . . . .	141
10.3.7. Velocity dependent mixing coefficients (=31) . . . . .	141
10.3.8. Deformation rate dependent viscosities ( <i>nn_ahm_ijk_t</i> =32) . . . . .	141
10.3.9. About space and time varying mixing coefficients . . . . .	142
10.4. Eddy induced velocity ( <i>ln_ldfeiv</i> ) . . . . .	142
10.5. Mixed layer eddies ( <i>ln_mle</i> ) . . . . .	143

## 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 6.2](#) 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 6.2](#)) and `&namdyn_ldf` ([namelist 5.4](#)) 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](#)

## 10.1. Lateral mixing operators

We remind here the different lateral mixing operators that can be used. Further details can be found in [subsection 6.2.1](#) and [section 5.6](#).

### 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 UP3 advection scheme on momentum ( `ln_dynadv_up3=.true.` , see [subsection 5.3.2](#)) and can be useful for testing purposes.

### 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.

### 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 Laplacian operator twice. Note that one can not combine Laplacian and Bilaplacian operators for the same variable.

## 10.2. Direction of lateral mixing ( `ldfslp.F90` )

The direction for lateral mixing must be specified and can be oriented along the coordinate levels ( `ln_traldf_lev` ), horizontal surfaces ( `ln_traldf_hor` ), or iso-neutral surfaces ( `ln_traldf_iso` ). The operator does not act along the model levels and must be rotated 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 6.8](#)), while for momentum the slopes are  $r_{1t}$ ,  $r_{1uw}$ ,  $r_{2f}$ ,  $r_{2uw}$  for *u* and  $r_{1f}$ ,  $r_{1vw}$ ,  $r_{2t}$ ,  $r_{2vw}$  for *v*.

### 10.2.1. Slopes for 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 6.8](#) 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.

$$\begin{aligned}
 r_{1u} &= \frac{e_{3u}}{\left( e_{1u} \overline{\overline{\overline{i+1/2, k}}} \right)} \delta_{i+1/2}[z_t] \approx \frac{1}{e_{1u}} \delta_{i+1/2}[z_t] \\
 r_{2v} &= \frac{e_{3v}}{\left( e_{2v} \overline{\overline{\overline{j+1/2, k}}} \right)} \delta_{j+1/2}[z_t] \approx \frac{1}{e_{2v}} \delta_{j+1/2}[z_t] \\
 r_{1w} &= \frac{1}{e_{1w}} \overline{\overline{\overline{\delta_{i+1/2}[z_t]}}}^{i, k+1/2} \approx \frac{1}{e_{1w}} \delta_{i+1/2}[z_{uw}] \\
 r_{2w} &= \frac{1}{e_{2w}} \overline{\overline{\overline{\delta_{j+1/2}[z_t]}}}^{j, k+1/2} \approx \frac{1}{e_{2w}} \delta_{j+1/2}[z_{vw}]
 \end{aligned} \tag{10.1}$$

These slopes are computed once in `ldf_slp_init` when `ln_sco=.true.`, and either `ln_traldf_hor=.true.` or `ln_dynldf_hor=.true.`

### 10.2.2. 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  $\rho$  in [equation 6.8](#) and setting the diffusive fluxes in the three directions to zero leads to the following definition for the neutral slopes:

$$\begin{aligned}
 r_{1u} &= \frac{e_{3u}}{e_{1u}} \frac{\overline{\delta_{i+1/2}[\rho]}}{\overline{\delta_{k+1/2}[\rho]}}^{i+1/2, k} \\
 r_{2v} &= \frac{e_{3v}}{e_{2v}} \frac{\overline{\delta_{j+1/2}[\rho]}}{\overline{\delta_{k+1/2}[\rho]}}^{j+1/2, k} \\
 r_{1w} &= \frac{e_{3w}}{e_{1w}} \frac{\overline{\delta_{i+1/2}[\rho]}}{\overline{\delta_{k+1/2}[\rho]}}^{i, k+1/2} \\
 r_{2w} &= \frac{e_{3w}}{e_{2w}} \frac{\overline{\delta_{j+1/2}[\rho]}}{\overline{\delta_{k+1/2}[\rho]}}^{j, k+1/2}
 \end{aligned} \tag{10.2}$$

As the mixing is performed along neutral surfaces, the gradient of  $\rho$  in [equation 10.2](#) has to be evaluated at the same local pressure (which, in decibars, is approximated by the depth in meters in the model). Therefore [equation 10.2](#) cannot be used as such, but further transformation is needed depending on the vertical coordinate used:

**$z$ -coordinate with full step or partial cell:** in [equation 10.2](#) 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 6.8.2](#)).

**$s$ - or hybrid  $s$ - $z$ - coordinate:** in the current release of NEMO, iso-neutral mixing is only employed for  $s$ -coordinates if the triad operator is used (`ln_traldf_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 10.2](#) 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:

$$\begin{aligned}
 r_{1u} &= \frac{e_{3u}}{e_{1u}} \frac{\alpha_u \overline{\delta_{i+1/2}[T]} - \beta_u \overline{\delta_{i+1/2}[S]}}{\alpha_u \overline{\delta_{k+1/2}[T]} - \beta_u \overline{\delta_{k+1/2}[S]}}^{i+1/2, k} \\
 r_{2v} &= \frac{e_{3v}}{e_{2v}} \frac{\alpha_v \overline{\delta_{j+1/2}[T]} - \beta_v \overline{\delta_{j+1/2}[S]}}{\alpha_v \overline{\delta_{k+1/2}[T]} - \beta_v \overline{\delta_{k+1/2}[S]}}^{j+1/2, k} \\
 r_{1w} &= \frac{e_{3w}}{e_{1w}} \frac{\alpha_w \overline{\delta_{i+1/2}[T]} - \beta_w \overline{\delta_{i+1/2}[S]}}{\alpha_w \overline{\delta_{k+1/2}[T]} - \beta_w \overline{\delta_{k+1/2}[S]}}^{i, k+1/2} \\
 r_{2w} &= \frac{e_{3w}}{e_{2w}} \frac{\alpha_w \overline{\delta_{j+1/2}[T]} - \beta_w \overline{\delta_{j+1/2}[S]}}{\alpha_w \overline{\delta_{k+1/2}[T]} - \beta_w \overline{\delta_{k+1/2}[S]}}^{j, k+1/2}
 \end{aligned}$$



where  $\alpha$  and  $\beta$ , the thermal expansion and saline contraction coefficients introduced in subsection 6.8.2, 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  $\alpha$ :  $\alpha_u = \overline{\alpha_T}^{i+1/2}$ ,  $\alpha_v = \overline{\alpha_T}^{j+1/2}$  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 (Guilyardi et al., 2001): 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 10.1). 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.

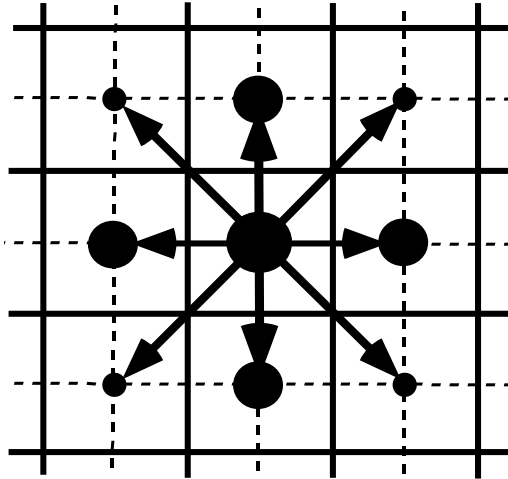


Figure 10.1.: Averaging procedure for isopycnal slope computation

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 (see figure 10.2), 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).

### 10.2.3. 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 5.18 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 10.1 and equation 10.2:

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

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 9.1).

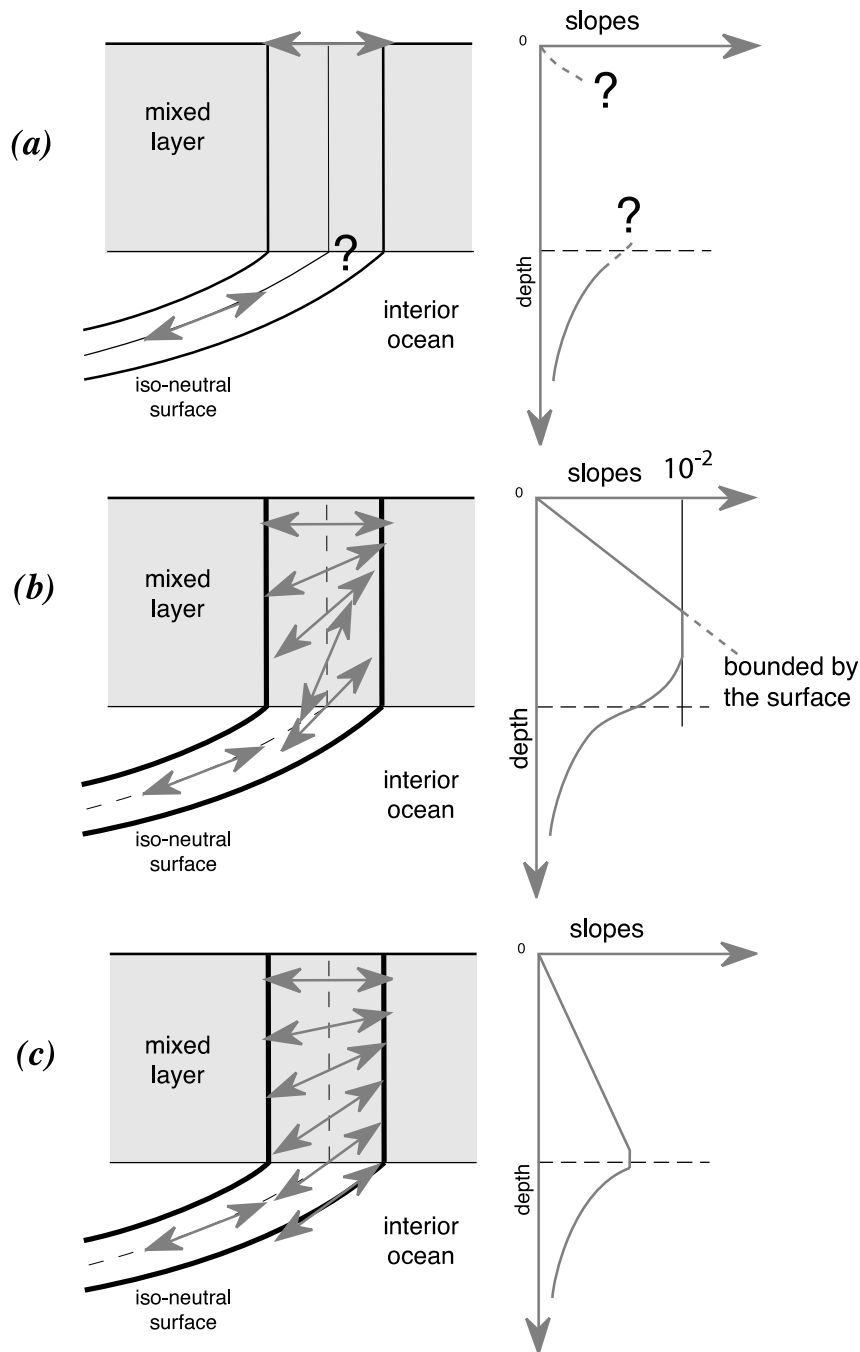


Figure 10.2.: Vertical profile of the slope used for lateral mixing in the mixed layer: (a) in the real ocean the slope is the iso-neutral slope in the ocean interior, which has to be adjusted at the surface boundary *i.e.* it must tend to zero at the surface since there is no mixing across the air-sea interface: wall boundary condition. Nevertheless, the profile between the surface zero value and the interior iso-neutral one is unknown, and especially the value at the base of the mixed layer; (b) profile of slope using a linear tapering of the slope near the surface and imposing a maximum slope of  $1/100$ ; (c) profile of slope actually used in *NEMO*: a linear decrease of the slope from zero at the surface to its ocean interior value computed just below the mixed layer. Note the huge change in the slope at the base of the mixed layer between (b) and (c).

### 10.3. 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:

#### 10.3.1. 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  $20^\circ$  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 10.1).

Namelist parameter	Input filename	dimensions	variable names
<code>nn_ahm_ijk_t=-20</code>	<code>eddy_viscosity_2D.nc</code>	$(i, j)$	<code>ahmt_2d</code> , <code>ahmf_2d</code>
<code>nn_aht_ijk_t=-20</code>	<code>eddy_diffusivity_2D.nc</code>	$(i, j)$	<code>ahtu_2d</code> , <code>ahtv_2d</code>
<code>nn_ahm_ijk_t=-30</code>	<code>eddy_viscosity_3D.nc</code>	$(i, j, k)$	<code>ahmt_3d</code> , <code>ahmf_3d</code>
<code>nn_aht_ijk_t=-30</code>	<code>eddy_diffusivity_3D.nc</code>	$(i, j, k)$	<code>ahtu_3d</code> , <code>ahtv_3d</code>

Table 10.1.: Description of expected input files if mixing coefficients are read from NetCDF files

### 10.3.2. 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:

$$A_o^l = \begin{cases} \frac{1}{2} U_{scl} L_{scl} & \text{for laplacian operator} \\ \frac{1}{12} U_{scl} L_{scl}^3 & \text{for bilaplacian operator} \end{cases} \quad (10.3)$$

$U_{scl}$  and  $L_{scl}$  are given by the namelist parameters `rn_Ud` , `rn_Uv` , `rn_Ld` and `rn_Lv` .

### 10.3.3. 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 10.3, 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.

### 10.3.4. 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:

$$A^l = \begin{cases} \frac{1}{2} U_{scl} \max(e_1, e_2) & \text{for laplacian operator} \\ \frac{1}{12} U_{scl} \max(e_1, e_2)^3 & \text{for bilaplacian operator} \end{cases} \quad (10.4)$$

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 subsection 2.2.2), especially when using a bilaplacian operator.

### 10.3.5. 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.

### 10.3.6. Eddy parameterization ( `nn_ah_t_ijk_t=21` )

This parameterization is designed to represent the tracer lateral mixing induced by (unresolved) quasi-geostrophic eddies. It is originally combined with the [Gent and McWilliams \(1990\)](#) advective formulation but is also available here for (Laplacian only) diffusion.

[Tréguier et al. \(1997\)](#) express tracer diffusivities as the product of a characteristic inverse timescale,  $T_{eff}^{-1}$ , by a wavenumber of the "energy-containing scale",  $k_0$ :

$$A^{lT} = k_0^{-2} T_{eff}^{-1} \quad (10.5)$$

$T_{eff}^{-1}$  is taken as the maximum growth rate of baroclinic waves, which reads:

$$T_{eff}^{-1} = \left( \frac{1}{H + \eta} \int_0^{H+\eta} \frac{\nabla_h b \cdot \nabla_h b}{\partial_z b} dz \right)^{\frac{1}{2}} \quad (10.6)$$

where  $b = -g\rho/\rho_0$  is the buoyancy. Taking advantage of the slope computation described previously, this is translated as follows in the code:

$$T_{eff}^{-1} = \left( \frac{1}{H + \eta} \int_0^{H+\eta} N^2 (r_1^2 + r_2^2) dz \right)^{\frac{1}{2}} \quad (10.7)$$

The exact formulation of the lengthscale is left open by [Tréguier et al. \(1997\)](#) but they suggest the first Rossby radius of deformation as a possible choice, which is the case in NEMO. Hence, we have:

$$k_0^{-1} \approx Ro \approx \frac{1}{\pi f} \int_0^{H+\eta} N dz \quad (10.8)$$

where  $Ro$  is constrained to be  $> 2km$  and  $< 40km$ .  $A^{lT}$  is, in addition, progressively tapered to 0 equatorward of  $20^\circ N/S$ .

### 10.3.7. Flow dependent mixing coefficients ( `nn_ah_t_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 2 for the Laplacian and 12 for the Bilaplacian):

$$A^l = \begin{cases} \frac{1}{2}|U|e & \text{for laplacian operator} \\ \frac{1}{12}|U|e^3 & \text{for bilaplacian operator} \end{cases} \quad (10.9)$$

### 10.3.8. 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.:

$$\begin{aligned} T_{smag}^{-1} &= \sqrt{(\partial_x u - \partial_y v)^2 + (\partial_y u + \partial_x v)^2} \\ L_{smag} &= \frac{1}{\pi} \frac{e_1 e_2}{e_1 + e_2} \end{aligned} \quad (10.10)$$

Introducing a user defined constant  $C$  (given in the namelist as `rn_csmc`), one can deduce the mixing coefficients as follows:

$$A_{smag} = \begin{cases} C^2 T_{smag}^{-1} L_{smag}^2 & \text{for laplacian operator} \\ \frac{C^2}{8} T_{smag}^{-1} L_{smag}^4 & \text{for bilaplacian operator} \end{cases} \quad (10.11)$$

For stability reasons, upper and lower limits are applied on the resulting coefficient (see [subsection 2.2.2](#)) so that:

$$\begin{aligned} C_{min} \frac{1}{2} |U|e < A_{smag} < C_{max} \frac{e^2}{8\Delta t} & \quad \text{for laplacian operator} \\ C_{min} \frac{1}{12} |U|e^3 < A_{smag} < C_{max} \frac{e^4}{64\Delta t} & \quad \text{for bilaplacian operator} \end{aligned} \quad (10.12)$$

```

!-----
&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
!                   ! = 32 F(i,j,t) = GEOMETRIC parameterization      (=> fill namldf_eke)
!
! 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_eke_equ  = .false. ! switch on update of GEOMETRIC eddy energy equation      (=> fill namldf_eke)
!                   ! forced to be true if nn_aei_ijk_t = 32
/

```

namelist 10.1.: `&namtra_eiv`

where  $C_{min}$  and  $C_{max}$  are adimensional namelist parameters given by `rn_minfac` and `rn_maxfac` respectively.

### 10.3.9. 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](#)).

## 10.4. 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 10.1](#) is used in  $z$ -coordinates, and the sum [equation 10.1](#) + [equation 10.2](#) 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:

$$\begin{aligned}
u^* &= \frac{1}{e_{2u}e_{3u}} \delta_k \left[ e_{2u} A_{uw}^{eiv} \overline{r_{1w}}^{i+1/2} \right] \\
v^* &= \frac{1}{e_{1u}e_{3v}} \delta_k \left[ e_{1v} A_{vw}^{eiv} \overline{r_{2w}}^{j+1/2} \right] \\
w^* &= \frac{1}{e_{1w}e_{2w}} \left\{ \delta_i \left[ e_{2u} A_{uw}^{eiv} \overline{r_{1w}}^{i+1/2} \right] + \delta_j \left[ e_{1v} A_{vw}^{eiv} \overline{r_{2w}}^{j+1/2} \right] \right\}
\end{aligned} \tag{10.13}$$

where  $A^{eiv}$  is the eddy induced velocity coefficient whose value is set through `nn_aei_ijk_t` `&namtra_eiv` ([namelist 10.1](#)) 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 6.1](#)) and not just the 2<sup>nd</sup> 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).

In the case of `nn_aei_ijk_t = 32`, the GEOMETRIC scaling for the eddy induced velocity coefficient from [Mak et al. \(2022b\)](#)

$$A^{eiv} = \alpha \frac{\hat{E}}{\int sN \Gamma(z) dz} \Gamma(z), \tag{10.14}$$

```

!-----
&namldf_eke !   GEOMETRIC param. (total EKE equation)                               (nn_aei_ijk_t = 32)
!-----
rn_ekedis   = 100.    ! dissipation time scale of EKE [days]
nn_eke_dis  = 0       ! dissipation option
!           ! = 0 constant in space
!           ! =-20 read in geom.diss_2D.nc file
rn_geom     = 0.04    ! geometric parameterization master coefficient (>0 & <1)
rn_eke_init = 1.e-1   ! initial total EKE value
rn_eke_min  = 1.e+0   ! background value of total EKE
rn_ross_min = 4.e+3   ! tapering of aeiv based on min Rossby radius [m]
!           ! set to zero to not taper it
rn_eke_lap  = 500.    ! Laplacian diffusion coefficient of EKE
!           ! this is in all options below, so set it to zero and nothing is done
rn_aeiv_min = 1.e+1   ! minimum bound of eiv coefficient
rn_aeiv_max = 1.5e+4  ! maximum bound of eiv coefficient
rn_SFmin    = 1.0     ! minimum bound of Structure Function
rn_SFmax    = 1.0     ! maximum bound of Structure Function
nn_eke_opt  = 1       ! options for terms to include in EKE budget
!           ! = 0 PE->EKE conversion, dissipation only
!           ! = 1 as 0 but with advection
!           ! = 2 as 1 but with additional KE->EKE conversion
!           ! for testing purposes:
!           ! = 88 only advection by depth-averaged flow
!           ! = 99 only Laplacian diffusion
ln_adv_wav  = .false. ! include advection at long Rossby speed
ln_beta_plane = .false. ! beta plane option for computing long Rossby speed (default: sphere option)
/
    
```

namelist 10.2.: &namldf\_eke

```

!-----
&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 10.3.: &namtra\_mle

is used, where  $\alpha$  ( `rn_geom` ) is a non-dimensional factor bounded in magnitude by 1,  $\Gamma(z) = N^2/N_{ref}^2$  (controlled by `rn_SFmin` and `rn_SFmax`, switch off by setting them equal to 1) is a vertical structure function based on [Ferreira et al. \(2005\)](#), and  $s$  is the isopycnal slope ( $s^2 = r_{1w}^2 + r_{2w}^2$ ). The parameterized depth-integrated eddy energy  $\hat{E}$  is calculated from

$$\frac{d\hat{E}}{dt} + \underbrace{\nabla_H \cdot ((\tilde{u}^z - c) \hat{E})}_{\text{advection}} = \underbrace{\int A^{eiv} s^2 N^2 dz}_{\text{source}} - \underbrace{\lambda(\hat{E} - \hat{E}_0)}_{\text{dissipation}} + \underbrace{\eta_E \nabla_H^2 \hat{E}}_{\text{diffusion}}, \quad (10.15)$$

where  $\nabla_H$  is the horizontal gradient operator,  $\tilde{u}^z$  is the depth-averaged velocity in the 1, 2 direction,  $c$  is the long Rossby phase velocity pointing into the  $i$  direction with speed  $|c| = \pi^{-1} \int |N| dz$  via a WKB-type approximation,  $\lambda$  ( `rn_eke_dis` ) is a linear dissipation time-scale in units of days (converted to per second in NEMO),  $\hat{E}_0$  ( `rn_eke_min` ) is a stabilizer for oscillations in  $\hat{E}$ , and  $\eta_E$  ( `rn_eke_lap` ) is a diffusion coefficient. Various options controlling the calculation of  $A^{eiv}$  or  $\hat{E}$  may be made through namelist parameters in `&namldf_eke` ([namelist 10.2](#)).

An option is provided to read in a bespoke spatially varying but constant in time  $\lambda^{-1}$  in units of days ( `rn_eke_dis` = -20). See [Mak et al. \(2022a\)](#) and associated data repository for an estimate and some scripts to regenerate the estimates and/or sample this on various ORCA grids.

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](#).

## 10.5. Mixed layer eddies ( `ln_mle` )

Submesoscale dynamics are primarily driven by the formation of fronts and the associated ageostrophic circulations. The parameterization introduced by [Fox-Kemper et al. \(2008\)](#), referred to here as the mixed-layer

eddy (MLE) parameterization, represents mixed-layer restratification caused by frontal instabilities and frontogenesis. This parameterization incorporates an eddy-induced overturning streamfunction, scaled to account for the MLE-induced vertical buoyancy flux, which acts to release potential energy. When `ln_mle=.true.` in `&namtra_mle` (`namelist 10.3`) namelist, this parameterization of the mixing due to unresolved mixed layer instabilities is activated. Additional transport derived from this streamfunction is computed in `ldf_mle_trp` and added to the eulerian transport in `tra_adv` as done for eddy induced advection. The specific formulation of the streamfunction depends on the chosen configuration:

`nn_mle=0`

Following Fox-Kemper et al. (2008), the streamfunction is defined as

$$\Psi = C_e \frac{H^2 \nabla_h \bar{b}^z \times \mathbf{k}}{|f|} \mu(z) \quad (10.16)$$

where  $H$  is the mixed layer depth,  $\bar{b}^z$  is the buoyancy averaged over the mixed layer depth  $f$  is the Coriolis parameter and  $\mu(z)$  the structure function  $\mu(z) = \max\left(0, \left[1 - \left(\frac{2z}{H} + 1\right)^2\right], \left[1 + \frac{5}{21} \left(\frac{2z}{H} + 1\right)^2\right]\right)$

`nn_mle=1`

Following Fox-Kemper et al. (2011), the streamfunction is defined as

$$\Psi = C_e \frac{\Delta s}{L_f} \frac{H^2 \nabla_h \bar{b}^z \times \mathbf{k}}{\sqrt{|f_2 + \tau^{-2}}}} \mu(z) \quad (10.17)$$

where  $\tau$  is the time scale over which submesoscale eddies develop and restratify the mixed layer,  $\Delta s$  is the grid resolution,  $L_f$  is the typical scale of mixed layer front.  $L_f$  is known to vary from 20 km near the Equator to 1 km or less at high latitude, taking a value of 5 km at mid latitudes (see Capet et al. XXX). We choose  $L_f$  to be inversely proportional to  $f$  with a coefficient that is chosen to fit the above  $L_f$  values.  $L_f = L_0 \frac{f_0}{|f|}$  The introduction of the timescale  $\tau$  to overcome the Equator singularity is not required anymore.

MLE tends to restratify the mixed layer. The action of the parameterization is most pronounced where mixed layers are deep and horizontal buoyancy gradients are large.



## Table of contents

11.1. Vertical mixing	146
11.1.1. Background values	146
11.1.2. Constant ( <code>ln_zdfcst</code> )	147
11.1.3. Richardson number dependent ( <code>ln_zdfric</code> )	147
11.1.4. TKE turbulent closure scheme ( <code>ln_zdftke</code> )	148
11.1.5. GLS: Generic Length Scale ( <code>ln_zdfgls</code> )	152
11.1.6. OSMOSIS boundary layer scheme ( <code>ln_zdfosm = .true.</code> )	154
11.1.7. Discrete energy conservation for TKE and GLS schemes	157
11.2. Convection	159
11.2.1. Non-penetrative convective adjustment ( <code>ln_tranpc</code> )	159
11.2.2. Enhanced vertical diffusion ( <code>ln_zdfevd</code> )	160
11.2.3. Mass Flux Convection ( <code>ln_zdfmfc</code> )	160
11.2.4. Handling convection with turbulent closure schemes ( <code>ln_zdf_{tke,gls,osm}</code> )	161
11.3. Double diffusion mixing ( <code>ln_zdfddm</code> )	161
11.4. Bottom and top friction ( <code>zdfdrg.F90</code> )	162
11.4.1. Free-slip boundary conditions ( <code>ln_drg_OFF</code> )	163
11.4.2. Linear top/bottom friction ( <code>ln_lin</code> )	163
11.4.3. Non-linear top/bottom friction ( <code>ln_non_lin</code> )	164
11.4.4. Log-layer top/bottom friction ( <code>ln_loglayer</code> )	164
11.4.5. Explicit top/bottom friction ( <code>ln_drgimp=.false.</code> )	164
11.4.6. Implicit top/bottom friction ( <code>ln_drgimp=.true.</code> )	165
11.4.7. Bottom friction with split-explicit free surface	165
11.5. Internal wave-driven mixing ( <code>ln_zdfiwm</code> )	166
11.6. Surface wave-induced mixing ( <code>ln_zdfswm</code> )	168

## Changes record

Release	Author(s)	Modifications
5.0	<i>D. Calvert</i> <i>C. de Lavergne, K. Hutchinson</i> <i>R. Bourdalle-Badie, S. Techene</i> <i>A. Moulin, E. Clementi</i>	<i>General updates</i> <i>Update Internal Wave Mixing section</i> <i>Add Mass Flux Correction scheme section</i> <i>Update of subsection 11.1.4 in for wave coupling</i>
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_zdfric   = .false.      ! local Richardson dependent formulation (T => fill namzdf_ric)
ln_zdf_tke  = .false.      ! Turbulent Kinetic Energy closure (T => fill namzdf_tke)
ln_zdf_gls  = .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
ln_zdfmfc   = .false.      ! Mass Flux Convection
!
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_zdfswm   = .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 11.1.: &namzdf

## 11.1. Vertical mixing

The discrete form of the ocean subgrid scale physics has been presented in [section 6.3](#) 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 7](#)), while at the bottom they are set to zero for heat and salt, unless a geothermal flux forcing is prescribed as a bottom boundary condition (*i.e.* `ln_trabbc` defined, see [subsection 6.4.3](#)), and specified through a bottom friction parameterisation for momentum (see [section 11.4](#)).

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 6.3](#) and [section 5.7](#)). These coefficients can be defined as constant, or a function of the local Richardson number, or computed from a turbulent closure model (either TKE or GLS or OSMOSIS formulation). This choice is specified via the appropriate namelist parameter in `&namzdf` ([namelist 11.1](#)).

The computation of these coefficients is initialized in the `zdfphy.F90` module and performed in the `zdf_fric.F90`, `zdf_tke.F90` or `zdf_gls.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.

### 11.1.1. Background values

To avoid numerical instabilities associated with weak vertical diffusion, all methods of calculating the eddy viscosity and diffusivity coefficients will enforce a minimum background value on their final values:  $A_b^{vm}$  and  $A_b^{vT}$  respectively. These background values are set by the namelist parameters `rn_avm0` and `rn_avt0` respectively, which should be at least as large as molecular values (see [subsection 11.1.2](#)).

Vertical and horizontal profiles may be applied to  $A_b^{vT}$  via the `nn_avb` and `nn_havtb` namelist parameters respectively. When these parameters are set to 0, no profile is applied and  $A_b^{vT}$  is constant everywhere. When setting `nn_avb=1`, a theoretical vertical profile will be applied to  $A_b^{vT}$  (Kraus, 1990). When setting `nn_havtb=1`,  $A_b^{vT}$  will be reduced in the tropics, decreasing linearly from  $\pm 15^\circ$  latitude to 10% of its nominal value (`rn_avt0`) at  $\pm 5^\circ$  latitude.

```

!-----
&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 11.2.: &namzdf\_ric

### 11.1.2. Constant ( ln\_zdfcst )

When `ln_zdfcst=.true.`, 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:

$$A_u^{vm} = A_v^{vm} = 1.2 \cdot 10^{-4} \text{ m}^2 \cdot \text{s}^{-1}$$

$$A^{vT} = A^{vS} = 1.2 \cdot 10^{-5} \text{ m}^2 \cdot \text{s}^{-1}$$

The coefficient values are set to their background values; see [subsection 11.1.1](#) and the parameters described within. 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.

### 11.1.3. Richardson number dependent ( ln\_zdf\_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 11.2](#)) 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:

$$\begin{cases} A^{vm} = \frac{A_{ric}^{vm}}{(1 + a Ri)^n} + A_b^{vm} \\ A^{vT} = \frac{A^{vm}}{(1 + a Ri)} + A_b^{vT} \end{cases}$$

where  $Ri = N^2 / (\partial_z \mathbf{U}_h)^2$  is the local Richardson number,  $N$  is the local Brunt-Vaisälä frequency (see [subsection 6.8.2](#)),  $A_b^{vT}$  and  $A_b^{vm}$  are the constant background values (see [subsection 11.1.1](#)), and  $A_{ric}^{vT} = 10^{-4} \text{ m}^2 \cdot \text{s}^{-1}$  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 `ln_mldw=.true.` in the namelist. In this case, the local depth of turbulent wind-mixing or "Ekman depth" ( $h_e$ ) 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 set by the namelist parameter `rn_ekmfc`,  $u^*$  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  $\rho_o$ . The minimum and maximum value of  $h_e$  is constrained by the namelist parameters `rn_mldmin` and `rn_mldmax` respectively. Once  $h_e$  is computed, the minimum values of the vertical eddy coefficients  $A^{vT}$  and  $A^{vm}$  within  $h_e$  are constrained by the namelist parameters `rn_wtmix` and `rn_wvmix` respectively ([Lermusiaux, 2001](#)).

```

!-----
&namzdf_tke ! turbulent eddy kinetic dependent vertical diffusion (ln_zdftke =T)
!-----
rn_ediff = 0.1 ! coef. for vertical eddy coef. (aut=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, aut=pdl(Ri)*aum) or not (=0, aut=aum)
nn_mxl = 3 ! mixing length: = 0 bounded by the distance to surface and bottom
! ! = 1 bounded by the local vertical scale factor
! ! = 2 first vertical derivative of mixing length bounded by 1
! ! = 3 as =2 with distinct dissipative an mixing length scale
ln_mxl0 = .true. ! surface mixing length scale = F(wind stress) (T) or not (F)
nn_mxlice = 0 ! type of scaling under sea-ice
! ! = 0 no scaling under sea-ice
! ! = 1 scaling with constant sea-ice thickness
! ! = 2 scaling with mean sea-ice thickness ( only with SI3 sea-ice model )
! ! = 3 scaling with maximum sea-ice thickness
rn_mxlice = 10. ! max constant ice thickness value when scaling under sea-ice ( nn_mxlice=1)
rn_mxl0 = 0.04 ! surface buoyancy lenght scale minimum value
ln_mxshw = .false. ! surface mixing length scale = F(wave height)
ln_lc = .true. ! Langmuir cell parameterisation (Azell 2002)
rn_lc = 0.15 ! coef. associated to Langmuir cells
nn_etau = 1 ! penetration of tke below the mixed layer (ML) due to NIWs
! ! = 0 none ; = 1 add a tke source below the ML
! ! = 2 add a tke source just at the base of the ML
! ! = 3 as = 1 applied on HF part of the stress (ln_cpl=T)
rn_efr = 0.05 ! fraction of surface tke value which penetrates below the ML (nn_etau=1 or 2)
nn_htau = 1 ! type of exponential decrease of the penetration below the ML
! ! = 0 constant 10 m length scale
! ! = 1 0.5m at the equator to 30m poleward of 40 degrees
nn_eice = 1 ! attenuation of langmuir & surface wave breaking under ice
! ! = 0 no impact of ice cover on langmuir & surface wave breaking
! ! = 1 weighed by 1-TANH(10*fr_i)
! ! = 2 weighed by 1-fr_i
! ! = 3 weighed by 1-MIN(1,4*fr_i)
nn_bc_surf = 1 ! surface condition (0/1=Dir/Neum) ! Only applicable for wave coupling (ln_cplwave=1)
nn_bc_bot = 1 ! bottom condition (0/1=Dir/Neum) ! Only applicable for wave coupling (ln_cplwave=1)
/

```

namelist 11.3.: &namzdf\_tke

#### 11.1.4. TKE turbulent closure scheme ( ln\_zdftke )

When `ln_zdftke=.true.`, 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:

$$\frac{\partial \bar{e}}{\partial t} = \frac{K_m}{e_3^2} \left[ \left( \frac{\partial u}{\partial k} \right)^2 + \left( \frac{\partial v}{\partial k} \right)^2 \right] - K_\rho N^2 + \frac{1}{e_3} \frac{\partial}{\partial k} \left[ \frac{A^{vm}}{e_3} \frac{\partial \bar{e}}{\partial k} \right] - c_\epsilon \frac{\bar{e}^{3/2}}{l_\epsilon} \quad (11.1)$$

$$K_m = C_k l_k \sqrt{\bar{e}}$$

$$K_\rho = A^{vm} / P_{rt}$$

where  $N$  is the local Brunt-Vaisälä frequency (see subsection 6.8.2),  $l_\epsilon$  and  $l_k$  are the dissipation and mixing length scales,  $P_{rt}$  is the Prandtl number,  $K_m$  and  $K_\rho$  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`.

The definition used for  $P_{rt}$  is controlled by the `nn_pdl` namelist parameter. If `nn_pdl=0`, then  $P_{rt} = 1$ . If `nn_pdl=1`, then  $P_{rt}$  is a function of the local Richardson number ( $R_i$ ) following Blanke and Delécluse (1993):

$$P_{rt} = \begin{cases} 1 & \text{if } R_i \leq 0.2 \\ 5 R_i & \text{if } 0.2 \leq R_i \leq 2 \end{cases}$$

At the sea surface, the value of  $\bar{e}$  is prescribed from the wind stress field as  $\bar{e}_o = e_{bb} |\tau| / \rho_o$ , where  $e_{bb}$  is set by the `rn_ebb` namelist parameter. The default value of  $e_{bb}$  is 3.75 (Gaspar et al., 1990), however a much larger

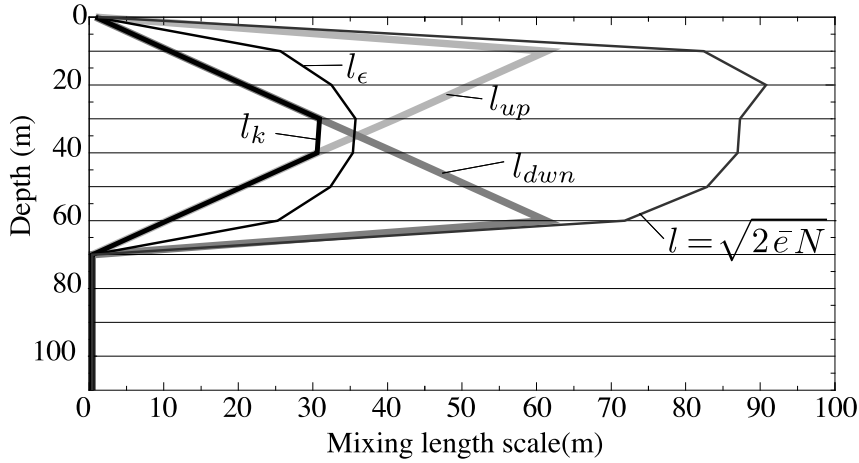


Figure 11.1.: Illustration of the mixing length computation

value can be used when taking into account the surface wave breaking (see below [equation 11.4](#)). The bottom value of  $\bar{\epsilon}$  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  $10^{-6} \text{ m}^2 \text{ s}^{-2}$ . This allows the subsequent formulations to match that of [Gargett \(1984\)](#) for the diffusion in the thermocline and deep ocean:  $K_\rho = 10^{-3}/N$ .

A separate minimum value is applied to the surface value of  $\bar{\epsilon}$ , set by the `rn_emin0` namelist parameter. This is typically larger; here it is set to  $10^{-4} \text{ m}^2 \text{ s}^{-2}$  by default.

### 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_mx1` namelist parameter.

The first two are based on the following first order approximation ([Blanke and Delécluse, 1993](#)):

$$l_k = l_\epsilon = \sqrt{2\bar{\epsilon}}/N \quad (11.2)$$

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_mx1=0`) or by the local vertical scale factor (`nn_mx1=1`). [Blanke and Delécluse \(1993\)](#) notice that this simplification has two major drawbacks: it makes no sense for locally unstable stratification and the computation no longer uses all the information contained in the vertical density profile.

To overcome these drawbacks, [Madec et al. \(1998\)](#) introduces the `nn_mx1=2, 3` cases, which add an extra assumption concerning the vertical gradient of the computed length scale. The length scales are first evaluated as in [equation 11.2](#) and then bounded such that the vertical variations of the length scale cannot be larger than the variations of depth:

$$\frac{1}{e_3} \left| \frac{\partial l}{\partial k} \right| \leq 1 \quad \text{with } l = l_k = l_\epsilon \quad (11.3)$$

This 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 11.1](#)).

In order to impose the [equation 11.3](#) 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):

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

where  $l^{(k)}$  is computed using [equation 11.2](#), i.e.  $l^{(k)} = \sqrt{2\bar{\epsilon}^{(k)}/N^{2(k)}}$ .

In the `nn_mx1=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_mx1=3` case, the dissipation and mixing turbulent length scales are as given in Gaspar et al. (1990):

$$\begin{aligned} l_\epsilon &= \sqrt{l_{up} l_{down}} \\ l_k &= \min(l_{up}, l_{down}) \end{aligned}$$

At the ocean surface, a non zero length scale is set through the `rn_mx10` 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 default value for `rn_mx10` of 0.04 m. In the ocean interior, a minimum length scale is set to recover the molecular viscosity when  $\bar{e}$  reaches its minimum value (such that  $1.10^{-6} = C_k l_{min} \sqrt{\bar{e}_{min}}$ ).

### Surface wave breaking parameterization (no information from an external wave model)

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 act on values of the surface length scale, TKE and the 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 :

$$\bar{e}_o = \frac{1}{2} (15.8 \alpha_{CB})^{2/3} \frac{|\tau|}{\rho_o} \quad (11.4)$$

where  $\alpha_{CB}$  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). Mellor and Blumberg (2004) suggest  $\alpha_{CB} = 100$  which, as the surface boundary condition on TKE in NEMO is prescribed through  $\bar{e}_o = e_{bb} |\tau| / \rho_o$ , corresponds to setting `rn_ebb=67.83`.

The namelist parameter `ln_mx10` determines the surface boundary condition on the turbulent length scale,  $l_o$ . When `ln_mx10=.false.`,  $l_o$  is set to the value specified by the namelist parameter `rn_mx10` (see previous subsection). When `ln_mx10=.true.`,  $l_o$  follows Charnock's relation:

$$l_o = \kappa \beta \frac{|\tau|}{g \rho_o} \quad (11.5)$$

where  $\kappa = 0.40$  is the von Karman constant, and  $\beta = 2.10^5$  is Charnock's constant set to the value chosen by Stacey (1999).

### Surface wave breaking parameterization (using information from an external wave model)

Surface boundary conditions for the turbulent kinetic energy, the mixing length scale and the dissipative length scale can be defined using wave fields provided from an external wave model (see chapter 7, section 7.10). The injection of turbulent kinetic energy at the surface can be given by the dissipation of the wave field usually dominated by wave breaking. In coupled mode, the wave to ocean energy flux term ( $\Phi_o$ ) from an external wave model can be provided and then converted into an ocean turbulence source by setting `ln_phioc=.true.`

The surface TKE can be defined by a Dirichlet boundary condition by setting `nn_bc_surf=0` in the `&namzdf` (namelist 11.1) namelist:

$$\bar{e}_o = \frac{1}{2} \left( 15.8 \frac{\Phi_o}{\rho_o} \right)^{2/3} \quad (11.6)$$

Due to the definition of the computational grid, the TKE flux is not applied at the free surface but at the centre of the topmost grid cell ( $z = z1$ ).

To be more accurate, a Neumann boundary condition equivalent to interpreting the half-grid cell at the top as a constant flux layer (consistent with the surface layer Monin-Obukhov theory) can be applied by instead setting `nn_bc_surf=1` (Couvelard et al., 2020):

$$\left( \frac{Km}{e_3} \partial_k e \right)_{z=z1} = \frac{\Phi_o}{\rho_o} \quad (11.7)$$

The mixing length scale surface value  $l_o$  can be estimated from the surface roughness length  $z0$ :

$$l_o = \kappa \frac{(C_k C_\epsilon)^{1/4}}{C_k} z0 \quad (11.8)$$

where  $z0$  is directly estimated from the significant wave height ( $H_s$ ) provided by the external wave model as  $z0 = 1.6H_s$ . To use this option, `ln_mxhsw`, `ln_wave` and `ln_sdw` have to be set to `.true.`



## Langmuir cells

Langmuir circulations (LC) can be described as organised 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 are 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 simulations, includes the whole effect of LC in an extra source term of TKE,  $P_{LC}$ . The presence of  $P_{LC}$  in [equation 11.1](#), the TKE equation, is controlled by setting `ln_lc=.true.` in the `&namzdf_tke` ([namelist 11.3](#)) namelist.

By making an analogy with the characteristic convective velocity scale (*e.g.*, [D'Alessio et al. \(1998\)](#)),  $P_{LC}$  is assumed to be :

$$P_{LC}(z) = (1 - F_i) \frac{w_{LC}^3(z)}{H_{LC}}$$

where  $w_{LC}(z)$  is the vertical velocity profile of LC,  $H_{LC}$  is the LC depth and  $F_i$  is a function of sea ice concentration ( $f_i$ ) representing the attenuation of wind-driven mixing under sea ice.

$F_i$  has several possible definitions, chosen via the `nn_eice` namelist parameter:

`nn_eice=0`

No attenuation of mixing under sea ice ( $F_i = 0$ )

`nn_eice=1`

TANH profile with no mixing at 100% ice concentration ( $F_i = \tanh(10f_i)$ )

`nn_eice=2`

Linear profile with no mixing at 100% ice concentration ( $F_i = f_i$ )

`nn_eice=3`

Linear profile with no mixing at 25% ice concentration ( $F_i = \min(4f_i, 1)$ )

$w_{LC}$  is assumed to be zero at the surface and 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 :

$$w_{LC} = \begin{cases} c_{LC} \|u_s^{LC}\| \sin(-\pi z/H_{LC}) & \text{if } -z \leq H_{LC} \\ 0 & \text{otherwise} \end{cases}$$

In the absence of information about the wave field,  $w_{LC}$  is assumed to be proportional to the surface Stokes drift ( $u_s^{LC} = u_{s0}$ ) empirically estimated by  $u_{s0} = 0.377 |\tau|^{1/2}$ , where  $|\tau|$  is the surface wind stress module \*.

In the case of online coupling with an external wave model (see [chapter 7, section 7.10](#)),  $w_{LC}$  is proportional to the component of the Stokes drift aligned with the wind ([Couvelard et al., 2020](#)) and  $u_s^{LC} = \max(u_{s0} \cdot e_\tau, 0)$  where  $e_\tau$  is the unit vector in the wind stress direction and  $u_{s0}$  is the surface Stokes drift provided by the external wave model.

$c_{LC} = 0.15$  has been chosen by [Axell \(2002\)](#) as a good compromise to fit large-eddy simulation data and yields maximum vertical velocities  $w_{LC}$  of the order of a few centimetres per second. The value of  $c_{LC}$  is set through the `rn_lc` namelist parameter and should have a value of between 0.15 and 0.54 ([Axell, 2002](#)).

$H_{LC}$  is estimated in a similar way to the turbulent length scale of TKE equations: it is the depth to which a water parcel with kinetic energy due to Stokes drift can reach on its own by converting its kinetic energy to potential energy, according to

$$-\int_{-H_{LC}}^0 N^2 z dz = \frac{1}{2} \|u_s^{LC}\|^2$$

## 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

\*Following [Li and Garrett \(1993\)](#), the surface Stoke drift velocity may be expressed as  $u_{s0} = 0.016 |U_{10m}|$ . Assuming an air density of  $\rho_a = 1.22 \text{ Kg/m}^3$  and a drag coefficient of  $1.5 \cdot 10^{-3}$  allows  $u_{s0}$  to be expressed 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_hsri      = 0.03 ! Ice-ocean 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_z0_ice    = 1 ! attenuation of surface wave breaking under ice
!           ! = 0 no impact of ice cover
!           ! = 1 roughness uses rn_hsri and is weighed by 1-TANH(10*fr_i)
!           ! = 2 roughness uses rn_hsri and is weighed by 1-fr_i
!           ! = 3 roughness uses rn_hsri and is weighed by 1-MIN(1,4*fr_i)
nn_mxlice    = 0 ! mixing under sea ice
!           ! = 0 No scaling under sea-ice
!           ! = 1 scaling with constant Ice-ocean roughness (rn_hsri)
!           ! = 2 scaling with mean sea-ice thickness
!           ! = 3 scaling with max sea-ice thickness
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 11.4.: &namzdf\_gls

of the ocean's planetary boundary layer that are not explicitly captured by 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 11.4](#), the standard TKE surface boundary condition, plus a depth dependance given by:

$$S = (1 - F_i) f_r e_s e^{-z/h_\tau} \quad (11.9)$$

where  $z$  is the depth,  $e_s$  is TKE surface boundary condition,  $f_r$  is the fraction of the surface TKE that penetrates into the ocean,  $h_\tau$  is a vertical mixing length scale that controls the exponential shape of the penetration, and  $F_i$  is a function of sea ice concentration ( $f_i$ ) representing the attenuation of wind-driven mixing under sea ice.

The value of  $f_r$ , usually a few percent, is specified through the `rn_efr` namelist parameter. The vertical mixing length scale,  $h_\tau$ , can be set as a 10 m uniform value (`nn_htau=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_htau=1`). As for the parameterisation of Langmuir Circulations,  $F_i$  has several possible definitions chosen via the `nn_eice` namelist parameter (see [subsubsection 11.1.4](#)).

Note that two other options exist, `nn_etau=2, 3`. They correspond to applying [equation 11.9](#) 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.

### 11.1.5. GLS: Generic Length Scale ( `ln_zdfgls` )

When `ln_zdfgls=.true.`, the vertical eddy viscosity and diffusivity coefficients are computed using the Generic Length Scale (GLS) scheme. The GLS scheme is a turbulent closure model based on two prognostic equations: one for the turbulent kinetic energy  $\bar{e}$ , and another for the generic length scale,  $\psi$  (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 [table 11.1](#) allows to recover a number of well-known turbulent closures including  $k$ - $kl$  (Mellor and Yamada, 1982),  $k$ - $\epsilon$  (Rodi, 1987) and  $k$ - $\omega$  (Wilcox, 1988) among others (Umlauf and Burchard, 2003; Kantha and Carniel, 2003).

The GLS scheme is given by the following set of equations:

$$\frac{\partial \bar{e}}{\partial t} = \frac{K_m}{\sigma_e e_3} \left[ \left( \frac{\partial u}{\partial k} \right)^2 + \left( \frac{\partial v}{\partial k} \right)^2 \right] - K_\rho N^2 + \frac{1}{e_3} \frac{\partial}{\partial k} \left[ \frac{K_m}{e_3} \frac{\partial \bar{e}}{\partial k} \right] - \epsilon \quad (11.10)$$

$$\frac{\partial \psi}{\partial t} = \frac{\psi}{\bar{\epsilon}} \left\{ \frac{C_1 K_m}{\sigma_\psi e_3} \left[ \left( \frac{\partial u}{\partial k} \right)^2 + \left( \frac{\partial v}{\partial k} \right)^2 \right] - C_3 K_\rho N^2 - C_2 \epsilon Fw \right\} + \frac{1}{e_3} \frac{\partial}{\partial k} \left[ \frac{K_m}{e_3} \frac{\partial \psi}{\partial k} \right]$$

$$K_m = C_\mu \sqrt{\bar{\epsilon}} l$$

$$K_\rho = C_{\mu'} \sqrt{\bar{\epsilon}} l$$

$$\epsilon = C_{0\mu} \frac{\bar{\epsilon}^{3/2}}{l}$$

where  $N$  is the local Brunt-Vaisälä frequency (see [subsection 6.8.2](#)) and  $\epsilon$  the dissipation rate.

The constants  $C_1$ ,  $C_2$ ,  $C_3$ ,  $\sigma_e$ ,  $\sigma_\psi$  and the wall function ( $Fw$ ) depend on the choice of the turbulence model. Four different turbulent models are pre-defined ([table 11.1](#)). They are made available through the `nn_clo` namelist parameter.

	$k - kl$	$k - \epsilon$	$k - \omega$	generic
<code>nn_clo</code>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>
$(p, n, m)$	( 0 , 1 , 1 )	( 3 , 1.5 , -1 )	( -1 , 0.5 , -1 )	( 2 , 1 , -0.67 )
$\sigma_k$	2.44	1.	2.	0.8
$\sigma_\psi$	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 11.1.: Set of predefined GLS parameters, or equivalently predefined turbulence models available with `ln_zdfgls=.true.` and controlled by the `nn_clo` namelist variable in `&namzdf_gls` ([namelist 11.4](#)).

In the Mellor-Yamada model, the negativity of  $n$  requires the use of a wall function to force the convergence of the mixing length towards  $\kappa z_b$  (where  $\kappa$  is the Von Karman constant and  $z_b$  the rugosity length scale) value near physical boundaries (logarithmic boundary layer law).

The stability functions  $C_\mu$  and  $C_{\mu'}$  have several definitions, chosen via the namelist parameter `nn_stab_func`:

`nn_stab_func=0`

Galperin et al. (1988) functions

`nn_stab_func=1`

Kantha and Clayson (1994) functions

`nn_stab_func=2`

Canuto et al. (2001) "model A" functions

`nn_stab_func=3`

Canuto et al. (2001) "model B" functions

The value of  $C_{0\mu}$  depends on this choice of the stability function.

The surface and bottom boundary conditions on both  $\bar{\epsilon}$  and  $\psi$  are chosen via the `nn_bc_surf` and `nn_bc_bot` namelist parameters respectively. They can be calculated using a Dirichlet (= 0) or Neumann (= 1) condition.

As for the TKE turbulent closure scheme ([subsection 11.1.4](#)), the wave effect on the mixing is parameterised following Mellor and Blumberg (2004) and Craig and Banner (1994). The namelist parameters `rn_crban` and `rn_charn` correspond to  $\alpha_{CB}$  in [equation 11.4](#) and  $\beta$  in [equation 11.5](#). Setting `rn_crban=0.` will disable this parameterisation.

The  $\psi$  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  $l_{max} = c_{lim} \sqrt{2\bar{\epsilon}}/N$  where 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

```

!-----
&namzdf_osm      !   OSM vertical diffusion                               (ln_zdfosm =T)
!-----
ln_use_osm_la = .false.      ! Use  rn_osm_la
rn_osm_la      = 0.3         ! Turbulent Langmuir number
rn_zdfosm_adjust_sd = 1.0    ! Stokes drift reduction factor
rn_osm_hblfrac = 0.1        ! specify top part of hbl for nn_osm_wave = 3 or 4
rn_osm_bl_thresh = 5.e-5    ! Threshold buoyancy for deepening of OSBL base
nn_ave = 0              ! choice of horizontal averaging on avt, avmu, avmu
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. 10.01 !1. ! diffusivity when unstable below BL (m2/s)
rn_osm_dstokes = 5.      ! Depth scale of Stokes drift (m)
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
nn_osm_SD_reduce = 0    ! Method used to get active Stokes drift from surface value
!                       ! = 0: No reduction
!                       ! = 1: use SD avged over top 10% hbl
!                       ! = 2: use surface value of SD fit to slope at rn_osm_hblfrac*hbl below surface
ln_zdfosm_ice_shelter = .true. ! reduce surface SD and depth scale under ice
ln_osm_mle = .true.     ! Use integrated FK-OSM model
/

```

namelist 11.5.: &namzdf\_osm

model for the turbulent fluxes. This clipping is activated by setting `ln_length_lim=.true.` and  $c_{lim}$  is set to the value of `rn_clim_galp`.

The time and space discretization of the GLS equations follows the same energetic consideration as for the TKE case described in subsection 11.1.7 (Burchard, 2002). An evaluation of the 4 GLS turbulent closure schemes can be found in Warner et al. (2005) for the ROMS model and in Reffray. et al. (2015) for the NEMO model.

### 11.1.6. OSMOSIS boundary layer scheme ( `ln_zdfosm` )

When `ln_zdfosm=.true.`, the vertical eddy viscosity and diffusivity coefficients are computed using the OSMOSIS scheme.

**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 calculated by the NEMO surface wave module (see section 7.10), though only the component parallel to the wind stress is retained. Significant wave height and wave-mean period are used to calculate the Stokes penetration depth, again following Breivik et al. (2014).

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  $1 \text{ m s}^{-2}$ ).

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` 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 `zdfmzl.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 [section 11.2](#) 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  $\Delta h$ ; [figure 11.2](#) shows typical (a) buoyancy structure and (b) turbulent buoyancy flux profiles 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:

$$w_{*L} = (u_*^2 u_{s0})^{1/3}; \quad (11.11)$$

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:

$$\nu_* = \{u_*^3 [1 - \exp(-.5La_t^2)] + w_{*L}^3\}^{1/3}. \quad (11.12)$$

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:

$$\omega_* = (\nu_*^3 + 0.5w_{*C}^3)^{1/3}. \quad (11.13)$$

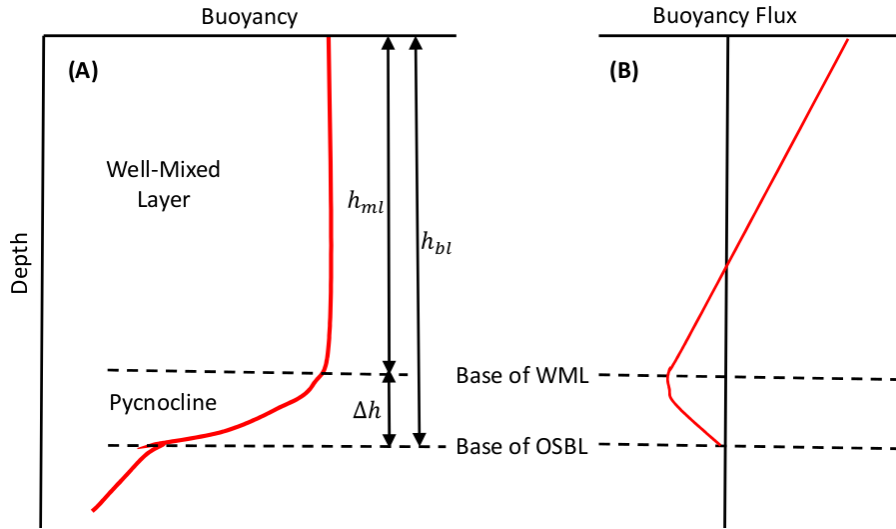


Figure 11.2.: The structure of the entraining boundary layer. (a) Mean buoyancy profile. (b) Profile of the buoyancy flux.

### The flux gradient model

The flux-gradient relationships used in the OSMOSIS scheme take the form:

$$\overline{w'\chi'} = -K \frac{\partial \overline{\chi}}{\partial z} + N_{\chi,s} + N_{\chi,b} + N_{\chi,t}, \quad (11.14)$$

where  $\chi$  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  $\chi$ .  $N_{\chi,s}$  represents the effects that the Stokes shear has on the transport of  $\chi$ ,  $N_{\chi,b}$  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

$$\sigma_{ml} = -z/h_{ml}; \quad \sigma_{bl} = -z/h_{bl}, \quad (11.15)$$

in unstable conditions the eddy diffusivity ( $K_d$ ) and eddy viscosity ( $K_\nu$ ) profiles are parametrized as:

$$K_d = 0.8 \omega_* h_{ml} \sigma_{ml} (1 - \beta_d \sigma_{ml})^{3/2} \quad (11.16)$$

$$K_\nu = 0.3 \omega_* h_{ml} \sigma_{ml} (1 - \beta_\nu \sigma_{ml}) (1 - \frac{1}{2} \sigma_{ml}^2) \quad (11.17)$$

where  $\beta_d$  and  $\beta_\nu$  are parameters that are determined by matching equation 11.16 and equation 11.17 to the eddy diffusivity and viscosity at the base of the well-mixed layer, given by

$$K_{d,ml} = K_{\nu,ml} = 0.16 \omega_* \Delta h. \quad (11.18)$$

For stable conditions the eddy diffusivity/viscosity profiles are given by:

$$K_d = 0.75 \nu_* h_{ml} \exp \left[ -2.8 (h_{bl}/L_L)^2 \right] \sigma_{ml} (1 - \sigma_{ml})^{3/2} \quad (11.19)$$

$$K_\nu = 0.375 \nu_* h_{ml} \exp \left[ -2.8 (h_{bl}/L_L)^2 \right] \sigma_{ml} (1 - \sigma_{ml}) (1 - \frac{1}{2} \sigma_{ml}^2). \quad (11.20)$$

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:

$$L_L = -w_{*L}^3 / \langle \overline{w'b'} \rangle_L, \quad (11.21)$$

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

$$\langle \overline{w'b'} \rangle_L = \frac{1}{2} \overline{w'b'}_0 - g \alpha_E \left[ \frac{1}{2} (I(0) + I(-h)) - \langle I \rangle \right]. \quad (11.22)$$

In unstable conditions the eddy diffusivity and viscosity depend on stability through the velocity scale  $\omega_*$ , which depends on the two velocity scales  $\nu_*$  and  $w_{*C}$ .

Details of the non-gradient terms in [equation 11.14](#) 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

$$\frac{\partial h_{bl}}{\partial t} = W_b - \frac{\overline{w'b'}_{ent}}{\Delta B_{bl}} \quad (11.23)$$

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  $\Delta B_{bl}$  is the difference between the buoyancy 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

$$\overline{w'b'}_{ent} = -\alpha_B \overline{w'b'}_0 - \alpha_S \frac{u_*^3}{h_{ml}} + G(\delta/h_{ml}) \left[ \alpha_S e^{-1.5 La_t} - \alpha_L \frac{w_{*L}^3}{h_{ml}} \right] \quad (11.24)$$

where the factor  $G \equiv 1 - e^{-25\delta/h_{bl}}(1 - 4\delta/h_{bl})$  models the lesser efficiency of Langmuir mixing when the boundary-layer depth is much greater than the Stokes depth, and  $\alpha_B$ ,  $\alpha_S$  and  $\alpha_L$  depend on the ratio of the appropriate eddy turnover time to the inertial timescale  $f^{-1}$ . Results from the LES suggest  $\alpha_B = 0.18F(fh_{bl}/w_{*C})$ ,  $\alpha_S = 0.15F(fh_{bl}/u_*)$  and  $\alpha_L = 0.035F(fh_{bl}/u_{*L})$ , where  $F(x) \equiv \tanh(x^{-1})^{0.69}$ .

For the stable boundary layer, the equation for the depth of the OSBL is:

$$\max \left( \Delta B_{bl}, \frac{w_{*L}^2}{h_{bl}} \right) \frac{\partial h_{bl}}{\partial t} = \left( 0.06 + 0.52 \frac{h_{bl}}{L_L} \right) \frac{w_{*L}^3}{h_{bl}} + \langle \overline{w'b'} \rangle_L. \quad (11.25)$$

[equation 11.23](#) 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 [equation 11.25](#) 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  $\Delta B_{bl}$  which will be  $\approx 0$  for the collapsing OSBL.

#### 11.1.7. 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 11.1](#) and [equation 11.10](#)) 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 11.3](#) 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:

$$\begin{aligned} & \int_{-H}^{\eta} u^t \partial_z (K_m^t (\partial_z u)^{t+\Delta t}) dz \\ &= \left[ u^t K_m^t (\partial_z u)^{t+\Delta t} \right]_{-H}^{\eta} - \int_{-H}^{\eta} K_m^t \partial_z u^t \partial_z u^{t+\Delta t} dz \end{aligned} \quad (11.26)$$

Here, the vertical diffusion of momentum is discretized backward in time with a coefficient,  $K_m$ , known at time  $t$  ([figure 11.3](#)), as it is required when using the TKE scheme (see [subsection 2.2.2](#)). The first term of the right hand side of [equation 11.26](#) 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 11.26](#) 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 11.1](#) and [equation 11.10](#)). This term must balance the input of potential energy resulting from

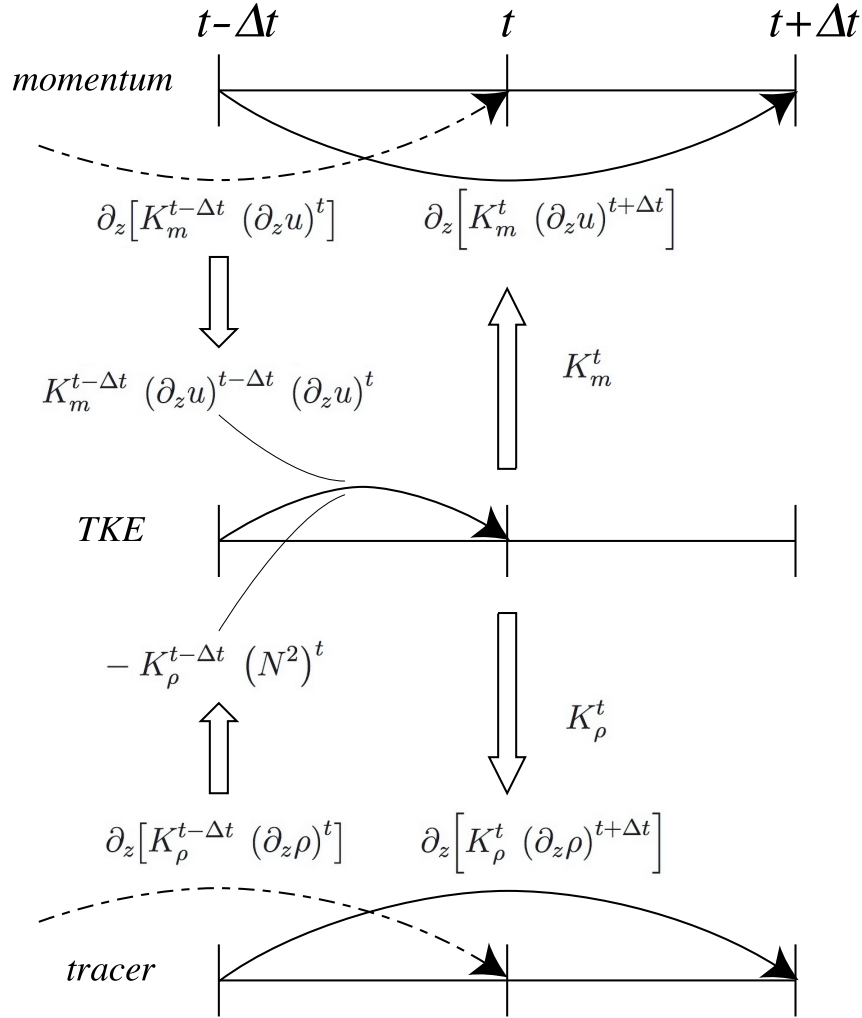


Figure 11.3.: Illustration of the subgrid kinetic energy integration in GLS and TKE schemes and its links to the momentum and tracer time integration.

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:

$$\begin{aligned}
 & \int_{-H}^{\eta} g z \partial_z (K_\rho^t (\partial_k \rho)^{t+\Delta t}) dz \\
 &= \left[ g z K_\rho^t (\partial_z \rho)^{t+\Delta t} \right]_{-H}^{\eta} - \int_{-H}^{\eta} g K_\rho^t (\partial_k \rho)^{t+\Delta t} dz \\
 &= - \left[ z K_\rho^t (N^2)^{t+\Delta t} \right]_{-H}^{\eta} + \int_{-H}^{\eta} \rho^{t+\Delta t} K_\rho^t (N^2)^{t+\Delta t} dz
 \end{aligned} \tag{11.27}$$

where we use  $N^2 = -g \partial_k \rho / (e_{3\rho})$ . The first term of the right hand side of [equation 11.27](#) 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 11.26](#) implies that, to be energetically consistent, the product  $K_\rho^{t-\Delta t} (N^2)^t$  should be used in [equation 11.1](#) and [equation 11.10](#).

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 11.26](#) 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 to be taken into account.



The above energetic considerations lead to the following final discrete form for the TKE equation:

$$\begin{aligned}
 \frac{(\bar{e})^t - (\bar{e})^{t-\Delta t}}{\Delta t} \equiv & \left\{ \left( \overline{\left( \overline{K_m}^{i+1/2} \right)^{t-\Delta t} \frac{\delta_{k+1/2}[u^{t+\Delta t}]}{e_3 u^{t+\Delta t}} \frac{\delta_{k+1/2}[u^t]}{e_3 u^t}} \right)^i \right. \\
 & \left. + \left( \overline{\left( \overline{K_m}^{j+1/2} \right)^{t-\Delta t} \frac{\delta_{k+1/2}[v^{t+\Delta t}]}{e_3 v^{t+\Delta t}} \frac{\delta_{k+1/2}[v^t]}{e_3 v^t}} \right)^j \right\} \\
 & - K_\rho^{t-\Delta t} (N^2)^t \\
 & + \frac{1}{e_3 w^{t+\Delta t}} \delta_{k+1/2} \left[ K_m^{t-\Delta t} \frac{\delta_k[(\bar{e})^{t+\Delta t}]}{e_3 w^{t+\Delta t}} \right] \\
 & - c_\epsilon \left( \frac{\sqrt{\bar{e}}}{l_\epsilon} \right)^{t-\Delta t} (\bar{e})^{t+\Delta t}
 \end{aligned} \tag{11.28}$$

where the last two terms in [equation 11.28](#) (vertical diffusion and Kolmogorov dissipation) are time stepped using a backward scheme (see [subsection 2.2.2](#)). Note that the Kolmogorov term has been linearized in time in order to render the implicit computation possible.

## 11.2. 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.

Two parameterisations are available to deal specifically with convective processes: a non-penetrative convective adjustment ([subsection 11.2.1](#)) or an enhanced vertical diffusion ([subsection 11.2.2](#)). It is recommended that one of these parameterisations be enabled when using either the TKE or GLS turbulent closure scheme, but not when using the OSMOSIS turbulent closure scheme (see [subsection 11.2.4](#)).

### 11.2.1. Non-penetrative convective adjustment ( `ln_tranpc` )

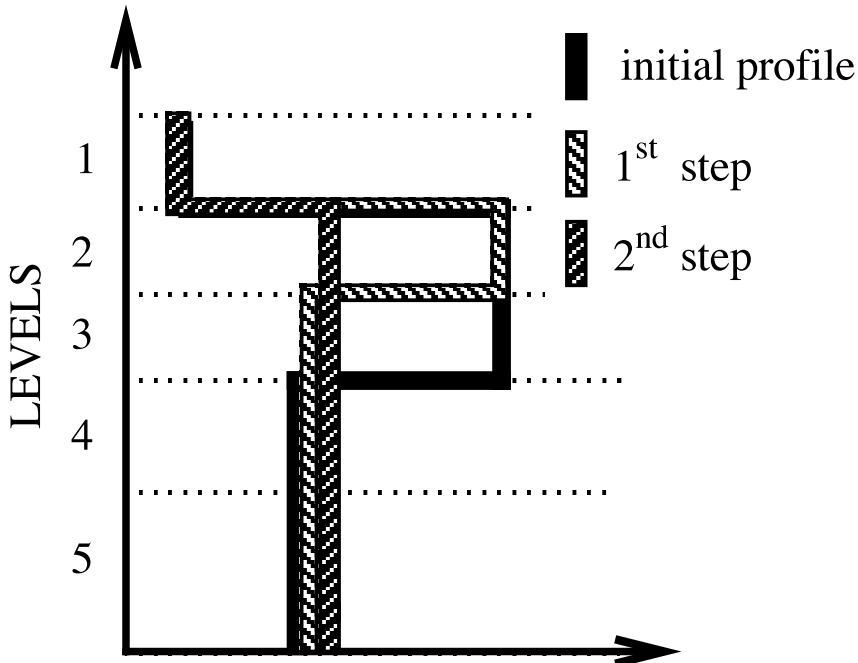


Figure 11.4.: Example of an unstable density profile treated by the non penetrative convective adjustment algorithm.  $1^{st}$  step: the initial profile is checked from the surface to the bottom. It is found to be unstable between levels 3 and 4. They are mixed. The resulting  $\rho$  is still larger than  $\rho(5)$ : levels 3 to 5 are mixed. The resulting  $\rho$  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  $\rho$  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 11.1](#)) 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 11.4): 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.

### 11.2.2. Enhanced vertical diffusion ( `ln_zdfevd` )

Options are defined through the `&namzdf` (namelist 11.1) 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 (`nn_evdm=1`).

In practice, where  $N^2 \leq 10^{-12}$ ,  $A_T^{vT}$  and  $A_T^{vS}$ , and if `nn_evdm=1`, the four neighbouring  $A_u^{vm}$  and  $A_v^{vm}$  values also, are set equal to the namelist parameter `rn_evd`. A typical value for `rn_evd` is between 1 and  $100 \text{ m}^2 \text{ s}^{-1}$ . 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 subsection 2.2.3).

### 11.2.3. Mass Flux Convection ( `ln_zdfmfc` )

The `ln_zdfmfc` option offers a new, coherent way to simultaneously parameterize local and non-local transport within the oceanic convective mixing layer (Giordani et al., 2020). This approach, initially developed for atmospheric models (Grant (2001); Soares et al. (2004); Pergaud et al. (2009)), assumes that subgrid turbulent fluxes in the convective boundary layer result from two distinct mixing scales: the local scale of small eddies, represented by vertical diffusion (TKE, GLS, etc.), and the scale of large eddies or convective thermals, represented by a non-local mass flux approach. The combination of both diffusive and convective schemes operating simultaneously is called as Eddy Diffusivity Mass Flux (EDMF).

The mass flux scheme is designed to represent all convection regimes, *i.e.*, from moderate to strong, and from intermediate to deep convection. Its goal is to calculate the effects of a population of convective plumes occupying a fraction of an ocean model grid cell on its prognostic variables. This scheme thus parameterizes the subgrid-scale effects of very fine-scale convective plumes on the resolved variables at the model grid scale. This phenomenon is a one-dimensional vertical process. Since convection is a non-local phenomenon, this scheme complements mixing schemes such as the TKE and Richardson schemes. It serves as an alternative to the "Enhanced Vertical Diffusion" (`ln_zdfevd`) or "Non-Penetrative Convection" (`ln_zdfnpc`) schemes currently implemented in *NEMO*. The scheme is active at every model time step, alongside diffusion. It is broken down into several steps, corresponding to the evolution equations for tracers within convective plumes and the mass flux resulting from the work of buoyancy forces in the vertical direction and the conservation of energy. The system of equations, along with its implementation in *NEMO*, is described in (Giordani et al., 2020), specifically in equations 12 and 13.

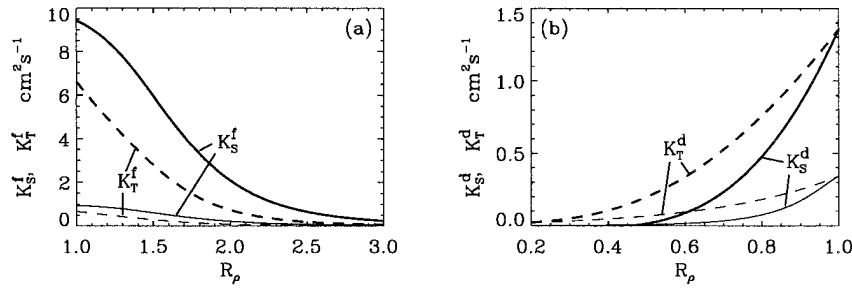


Figure 11.5.: From Merryfield et al. (1999): (a) Diapycnal diffusivities  $A_f^{vT}$  and  $A_f^{vS}$  for temperature and salt in regions of salt fingering. Heavy curves denote  $A^{*v} = 10^{-3} \text{ m}^2.\text{s}^{-1}$  and thin curves  $A^{*v} = 10^{-4} \text{ m}^2.\text{s}^{-1}$ ; (b) diapycnal diffusivities  $A_d^{vT}$  and  $A_d^{vS}$  for temperature and salt in regions of diffusive convection. Heavy curves denote the Fedorov parameterisation and thin curves the Kelley parameterisation. The latter is not implemented in NEMO.

### 11.2.4. Handling convection with turbulent closure schemes

(`ln_zdf`{`tke`,`gls`,`osm`})

The TKE and GLS turbulent closure schemes presented in subsection 11.1.4 and subsection 11.1.5 are, in theory, able to handle statically unstable density profiles. In such a case, the term corresponding to the destruction of turbulent kinetic energy through stratification in equation 11.1 or equation 11.10 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  $1 \text{ m}^2.\text{s}^{-1}$ ). These large values restore the static stability of the water column in a way similar to that of the enhanced vertical diffusion parameterisation (subsection 11.2.2). However, in the vicinity of the sea surface (first ocean layer), the eddy coefficients computed by the turbulent closure scheme do not usually exceed  $10^{-2} \text{ m}^2.\text{s}^{-1}$ , because the mixing length scale is bounded by the distance to the sea surface.

When using either the TKE (`ln_zdf``tke`=`.true.`) or GLS (`ln_zdf``gls`=`.true.`) turbulent closure scheme, it is therefore recommended to also enable the enhanced vertical diffusion parameterisation (`ln_zdf``evd`=`.true.`). This should not be done when using the OSMOSIS turbulent closure scheme (`ln_zdf``osm`=`.true.`), as this already includes enhanced vertical diffusion in the case of convection (see subsection 11.1.6).

## 11.3. Double diffusion mixing ( `ln_zdf``ddm` )

This parameterisation has been introduced in `zdfddm.F90` module and is controlled by the namelist parameter `ln_zdfddm` in `&namzdf` (namelist 11.1). 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  $\alpha$  and  $\beta$  are coefficients of thermal expansion and saline contraction (see subsection 6.8.1). To represent mixing of S and T by salt fingering, we adopt the diapycnal diffusivities suggested by Schmitt (1981):

$$A_f^{vS} = \begin{cases} \frac{A^{*v}}{1+(R_\rho/R_c)^n} & \text{if } R_\rho > 1 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \quad (11.29)$$

$$A_f^{vT} = 0.7 A_f^{vS} / R_\rho \quad (11.30)$$

The factor 0.7 in equation 11.30 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  $A^{*v} = 10^{-4} \text{ m}^2.\text{s}^{-1}$ .

To represent mixing of S and T by diffusive layering, the diapycnal diffusivities suggested by Fedorov (1988)

```

!-----
&namdrg      !  top/bottom drag coefficient                               (default: NO selection)
!-----
ln_drg_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 = vkarmm/log(z/z0) |U|
!
ln_drgimp    = .true.   !  implicit top/bottom friction flag
ln_drgice_imp = .true.  !  implicit ice-ocean drag
/

```

namelist 11.6.: `&namdrg`

```

!-----
&namdrg_top  !  TOP friction                                           (ln_drg_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 11.7.: `&namdrg_top`

are used:

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

$$A_d^{vS} = \begin{cases} A_d^{vT} (1.85 R_\rho - 0.85) & \text{if } 0.5 \leq R_\rho < 1 \text{ and } N^2 > 0 \\ A_d^{vT} 0.15 R_\rho & \text{if } 0 < R_\rho < 0.5 \text{ and } N^2 > 0 \\ 0 & \text{otherwise} \end{cases} \quad (11.31)$$

The dependencies of [equation 11.29](#) to [equation 11.31](#) on  $R_\rho$  are illustrated in [figure 11.5](#). Implementing this requires computing  $R_\rho$  at each grid point on every time step. This is done in *eosbn2.F90* ([section 6.8](#)) at the same time as  $N^2$  is computed, avoiding duplication in the computation of  $\alpha$  and  $\beta$  (which is usually quite expensive).

## 11.4. Bottom and top friction ( *zdfdrng.F90* )

Options to define the top and bottom friction are defined via parameters in the `&namdrg` ([namelist 11.6](#)) namelist, and, for top and bottom friction specifically, in the `&namdrg_top` ([namelist 11.7](#)) and `&namdrg_bot` ([namelist 11.8](#)) namelists respectively. 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:

$$A^{vm} (\partial \mathbf{U}_h / \partial z) = \mathcal{F}_h^U$$

```

!-----
&namdrg_bot  !  BOTTOM friction                                           (ln_drg_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 11.8.: `&namdrg_bot`

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  $d = \sqrt{2 A^{vm}/f} = 50$  m, one needs a vertical diffusion coefficient  $A^{vm} = 0.125 \text{ m}^2\text{s}^{-1}$  (for a Coriolis frequency  $f = 10^{-4} \text{ m}^2\text{s}^{-1}$ ). With a background value,  $A^{vm} = 10^{-4} \text{ m}^2\text{s}^{-1}$ , 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:

$$\frac{\partial u_k}{\partial t} = \frac{1}{e_{3u}} \left[ \frac{A^{vm}}{e_{3uw}} \delta_{k+1/2} [u] - \mathcal{F}_h^u \right] \approx -\frac{\mathcal{F}_h^u}{e_{3u}} \quad (11.32)$$

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* (section 5.7). 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:

$$\frac{\partial \mathbf{U}_h}{\partial t} = -\frac{\mathcal{F}_h^U}{e_{3u}} = \frac{c_b^U}{e_{3u}} \mathbf{U}_h^b \quad (11.33)$$

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 are referred to as  $c_b^T$  in the following. These are then linearly interpolated in space to get  $c_b^U$  at velocity points.

#### 11.4.1. Free-slip boundary conditions ( `ln_drg_OFF` )

When setting `ln_drg_OFF=.true.` free-slip conditions are used at the top and bottom boundaries, *i.e.* the drag coefficient used in equation 11.33 is set to  $c_b^T = 0$ .

#### 11.4.2. Linear top/bottom friction ( `ln_lin` )

The linear friction parameterisation (including the special case of a free-slip condition, subsection 11.4.1) 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  $\text{ms}^{-1}$ . The drag coefficient used in the general expression (equation 11.33) is therefore:

$$c_b^T = -r$$

$r$  is generally estimated as  $H/\tau$ , where  $\tau$  is a typical decay time in the deep ocean and  $H$  is the ocean depth. Commonly accepted values of  $\tau$  are of the order of 100 to 200 days (Weatherly, 1984). A value of  $\tau^{-1} = 10^{-7} \text{ s}^{-1}$ , 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).

In *NEMO*, linear friction is enabled by setting `ln_lin=.true.` .  $c_b^T$  is calculated in *zdfdrg.F90* and the trend due to the friction is added to the general momentum trend in *dynzdf.F90* .  $r$  is calculated as `rn_Cd0 * rn_Uc0`, where `rn_Cd0` and `rn_Uc0` are namelist parameters corresponding to the drag coefficient  $C_D$  and velocity scale  $U_{av}$  respectively. Their default values (0.001 and  $0.4 \text{ m s}^{-1}$  respectively) result in a friction coefficient of  $r = 4 \cdot 10^{-4} \text{ m s}^{-1}$ , corresponding to a decay time scale of 115 days when assuming an ocean depth of  $H = 4000 \text{ m}$ .

Local enhancements may be applied to the values of  $c_b^T$  by setting `ln_boost=.true.` and providing a 2D mask array (with values  $0 \leq M_b \leq 1$ ) via a NetCDF file. For bottom (top) friction, the array and file are named `bfr_coef` and `bfr_coef.nc` (`tfr_coef` and `tfr_coef.nc`) respectively. Locations with a non-zero mask value will have the friction coefficient increased by  $M_b * \text{rn\_boost} * \text{rn\_Cd0}$ .

### 11.4.3. Non-linear top/bottom friction ( `ln_non_lin` )

The non-linear bottom friction parameterisation assumes that the top/bottom friction is quadratic:

$$\mathcal{F}_h^U = \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} = C_D \sqrt{u_b^2 + v_b^2 + e_b} \mathbf{U}_h^b$$

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. The drag coefficient used in the general expression ([equation 11.33](#)) is therefore:

$$c_b^T = - C_D \left[ (\bar{u}_b^i)^2 + (\bar{v}_b^j)^2 + e_b \right]^{1/2}$$

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  $e_b = 2.5 \cdot 10^{-3} \text{m}^2 \text{s}^{-2}$ .

In *NEMO*, non-linear friction is enabled by setting `ln_non_lin=.true.`. As for linear friction,  $c_b^T$  is calculated in *zdfdr.F90* and the trend due to the friction is added to the general momentum trend in *dynzdf.F90*.  $C_D$  and  $e_b$  correspond to the namelist parameters `rn_Cd0` and `rn_ke0` respectively, with their default values set to those of the CME experiment. Note that for applications which consider tides explicitly, a low or even zero value of `rn_ke0` is recommended.

As for linear friction, local enhancements may be applied to the values of  $c_b^T$  by setting `ln_boost=.true.` (see [subsection 11.4.2](#)).

### 11.4.4. 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.  $C_D$  is then no longer constant, but instead related to the distance to the wall (or equivalently, to the half of the top/bottom layer thickness):

$$C_D = \left( \frac{\kappa}{\log(0.5 e_{3b}/z_0)} \right)^2$$

where  $\kappa$  is the von-Karman constant and  $z_0$  is a roughness length. This assumes that the model vertical resolution can capture the logarithmic layer, which typically occurs for layers thinner than 1 m or so.

This special case of the non-linear friction is enabled by setting `ln_loglayer=.true.` instead of `ln_non_lin=.true.`.  $z_0$  corresponds to the namelist parameter `rn_z0`, while  $C_D$  is bounded by the namelist parameters `rn_Cd0` and `rn_Cdmax` such that `rn_Cd0`  $\leq C_D \leq$  `rn_Cdmax`. The lower bound of `rn_Cd0` covers large layer thicknesses where logarithmic layers are presumably not resolved, while the upper bound of `rn_Cdmax` is applied for stability reasons.

The log-layer enhancement can also be applied to the top boundary friction if ice-shelf cavities are activated (`ln_isfcav=.true.`).

### 11.4.5. 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 11.4.7](#)). 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):

$$\left( \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_t = c_t^U \mathbf{u}_t^{n-1}$$

At the bottom (above the sea floor):

$$\left( \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_b = c_b^U \mathbf{u}_b^{n-1}$$

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 11.32](#) is:

$$\begin{aligned}\Delta u &= -\frac{\mathcal{F}_h^u}{e_{3u}} 2\Delta t \\ &= -\frac{ru}{e_{3u}} 2\Delta t\end{aligned}\tag{11.34}$$

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 11.34](#), 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  $1 \text{ m.s}^{-1}$  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  $|u| = 1 \text{ ms}^{-1}$ ,  $\Delta t = 1800 \text{ s}$ ,  $r = 10^{-3}$  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 `zdfdrv.F90` (assuming a  $1 \text{ ms}^{-1}$  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 11.4.6](#)). The number of potential breaches of the explicit stability criterion are still reported for information purposes.

#### 11.4.6. 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 11.6](#)) 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):

$$\left( \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_t = c_t^{\mathbf{U}} \mathbf{u}_t^{n+1}$$

At the bottom (above the sea floor):

$$\left( \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \Big|_b = c_b^{\mathbf{U}} \mathbf{u}_b^{n+1}$$

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.

#### 11.4.7. 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 4.1.2](#)), 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 it difficult to obtain 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:





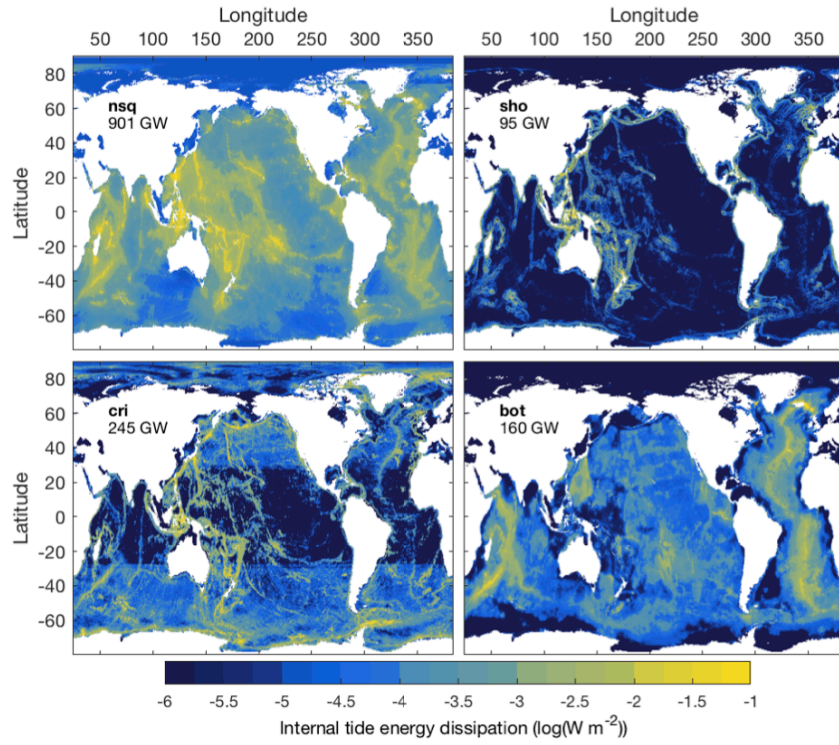


Figure 11.6.: Four power maps (in  $\log_{10}(W m^{-2})$ ), as estimated by de Lavergne et al. (2024), that enter the parameterization of internal wave-driven mixing. The globally integrated power (in GW) within each field is indicated at the top left of each panel. The overall power amounts to 1.40 TW.

- $E_{cri}(i, j)$ : bottom-intensified dissipation at critical slopes;
- $E_{bot}(i, j)$ : bottom-intensified dissipation above abyssal hills.

These power fields are read in a NetCDF forcing file whose name is set in the namelist section `&namzdf` (namelist 11.1). This forcing file also contains two maps of decay scales [m] necessary for the vertical distribution of  $E_{cri}$  and  $E_{bot}$ :

- $H_{cri}(i, j)$ : related to the height difference of the critical slope.
- $H_{bot}(i, j)$ : related to the wavelength of abyssal hills and to the energy flux  $E_{bot}$ .

Each power field goes with a specific vertical structure. The local turbulence production  $\epsilon(i, j, k)$  is thus obtained as the sum of four contributions,

$$\epsilon = \epsilon_{nsq} + \epsilon_{sho} + \epsilon_{cri} + \epsilon_{bot},$$

where, within each  $(i, j)$  water column,

- $\epsilon_{nsq} \propto N^2$
- $\epsilon_{sho} \propto N$
- $\epsilon_{cri} \propto \exp(-h_{ab}/H_{cri})$
- $\epsilon_{bot} \propto 1/(1 + h_{ab}/H_{bot})^2$

with  $h_{ab}$  the height above bottom and  $N$  the simulated buoyancy frequency. These vertical structures are interactive in the sense that they depend on the simulated stratification and sea surface height. Vertical distributions are thus computed every time step and are normalized so that

$$\int \rho \epsilon_{nsq} dz = E_{nsq}, \quad \int \rho \epsilon_{sho} dz = E_{sho}, \quad \int \rho \epsilon_{cri} dz = E_{cri}, \quad \int \rho \epsilon_{bot} dz = E_{bot}.$$

As of now, the parameterization only accounts for mixing powered by internal tides, thought to be the dominant supply of small-scale turbulence in the ocean interior (de Lavergne et al., 2020). It is recommended to use the forcing fields available at <https://doi.org/10.17882/103233>. These fields are state-of-the-art estimates of internal tide energy dissipation, including subinertial tides (de Lavergne et al., 2024). Details on the implementation of this parameterization and impacts in long NEMO eORCA1 experiments can be found in (de Lavergne et al., 2024).

## 11.6. Surface wave-induced mixing ( `ln_zdfswm` )

Surface waves produce an enhanced mixing through wave-turbulence interaction. In addition to breaking waves induced turbulence ([subsection 11.1.4](#)), the influence of non-breaking waves can be accounted for by 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:

$$B_v = \alpha A U_{st} \exp(3kz) \quad (11.35)$$

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  $\alpha$  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.

This parameterisation is enabled by setting `ln_zdfswm=.true.`. Additionally, both wave interaction (`ln_wave=.true.`) and calculation of the Stokes Drift (`ln_sdw=.true.`) must be enabled. The required wave fields (significant wave height and mean wave number) can be provided either in forced or coupled mode. For more information on wave parameters and settings, see [section 7.10](#).

## Output and Diagnostics (IOM, DIA, TRD)

### Table of contents

12.1. Model outputs . . . . .	170
12.2. Standard model diagnostic output with XIOS ( <i>iom_put</i> , <i>key_xios</i> ) . . . . .	170
12.2.1. Main XIOS configuration file ( <i>iodef.xml</i> ) . . . . .	171
12.2.2. Practical issues . . . . .	172
12.2.3. XML fundamentals . . . . .	173
12.2.4. Detailed functionalities . . . . .	176
12.2.5. CF metadata standard compliance . . . . .	181
12.2.6. Enabling NetCDF4 compression with XIOS . . . . .	182
12.3. Reading and writing restart files . . . . .	182
12.4. NetCDF4 support (legacy output file method) . . . . .	183
12.5. Tracer/Dynamics trends ( <i>namtrd</i> , <i>namtrc_trd</i> ) . . . . .	183
12.6. Transports across sections . . . . .	185
12.7. Diagnosing the steric effect on sea surface height . . . . .	187
12.8. Tidal harmonic and generic multiple-linear-regression analysis ( <i>diamlr.F90</i> ) . . . . .	189
12.8.1. Configuration of the multiple-linear-regression analysis . . . . .	189
12.8.2. The intermediate output and its post-processing . . . . .	190
12.9. Other diagnostics . . . . .	190
12.9.1. Depth of various quantities ( <i>diahth.F90</i> ) . . . . .	190
12.9.2. CMIP-specific and poleward transport diagnostics ( <i>diaar5.F90</i> , <i>diaptr.F90</i> ) . . . . .	191
12.9.3. De-tided diagnostic output from tidal models ( <i>dia25h.F90</i> , <i>diadetide.F90</i> ) . . . . .	191
12.9.4. Courant numbers . . . . .	192

### Changes record

Release	Author(s)	Modifications
5.0	<i>D. Calvert, A. Coward and S. Müller</i>	<i>General revisions; addition of section 12.8; expansion of subsection 12.9.3</i>
4.0	...	...
3.6	...	...
3.4	...	...
<=3.4	...	...

## 12.1. Model outputs

The model outputs are of three types: the output log/progress listings; the diagnostic output file(s); and the restart file(s).

The output log and progress listings are output in the *ocean.output* file(s), which contains information 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. Model errors that are caught by *NEMO* (via the `ctl_stop` subroutine) will issue a return code of 123 and information on the errors will be written to the *ocean.output* file. Additional progress information can be requested using the options explained in [subsection 16.4.1](#).

Diagnostic output files are written in NetCDF4 format and are generated by one of two available methods. With the legacy method (used when `key_xios` is not specified), output files have a predefined structure and contain time averaged diagnostics. If `key_diainstant` is specified, instantaneous diagnostics are instead output. With the standard method (used when `key_xios` is specified), *NEMO* can employ the full capability of the XIOS I/O server, which provides flexibility in the choice of the fields to be written as well as how the writing tasks are distributed over the processors in a massively parallel computing environment. A complete description of the use of this I/O server is presented in the next section.

The restart file is used by the code when the user wants to start the model with initial conditions defined by a previous simulation. Restart files are NetCDF files containing 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 stops and restarts and the same run performed in one continuous integration step. It should be noted that this requires that the restart file contains two consecutive time steps for all the prognostic variables.

Two methods are available to read and write restart files. The default method is for *NEMO* to perform these tasks: *NEMO* will generate a restart file for each MPP subdomain, which will then be read by the same subdomain on restarting. Therefore if a change in MPP decomposition is required between runs, then the individual restart files must first be combined into a single restart file for the full domain. This can be done using the `REBUILD_NEMO` tool. The alternative method is to use XIOS to read and/or write restart files (see [section 12.3](#)). This functionality was introduced in *NEMO* v4.2 and includes the ability to write restart data directly to a single file for the full domain.

## 12.2. Standard model diagnostic output with XIOS ( `iom_put` , `key_xios` )

The standard *NEMO* diagnostic output method (activated when `key_xios` is specified) uses an external I/O library and server named `XIOS` with the *NEMO* subroutine `iom_put` serving as the main interface to this library.

XIOS is developed by Yann Meurdesoif and his team at IPSL, and has its own [repository and support pages](#). *NEMO* v5 can be used with either [version 2](#) or [version 3](#) of XIOS. *NEMO* expects XIOS v2 by default and requires at least SVN revision 2131 of this version of the library. The use of XIOS v3 requires that the *NEMO* key `key_xios3` be specified. Further details are available in the [NEMO user guide](#). XIOS will create output files in NetCDF4 format, which is incompatible with the older NetCDF3 libraries. Post-processing and visualization tools must therefore be linked to NetCDF4 libraries to be able to handle the NetCDF files created by XIOS.// XIOS has been designed to be simple to use, flexible and efficient. Its two main purposes 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.

Table 12.1.: "xios" context variables typically used in the *iodef.xml* configuration files used by *NEMO*

variable name	description	example
<code>info_level</code>	verbosity level (0 to 100)	0
<code>using_server</code>	activate attached (false) or detached(true) mode	true
<code>using_oasis</code>	XIOS is used with OASIS (true) or not (false)	false
<code>oasis_codes_id</code>	<b>[XIOS 2 only]</b> when using oasis, define the identifier of <i>NEMO</i> in the nam-couple. Note that the identifier of XIOS is "xios.x"	oceanx

- Control over metadata via a large XML "database" of possible output fields.
- Control over the compression and/or precision of output fields (subject to certain conditions)

In addition, the `iom_put` interface allows the user to add in the *NEMO* code the output of any new variable (scalar, 1D, 2D or 3D) in a very easy way. The functionalities of XIOS and the `iom_put` interface are listed in the following subsections.

The second functionality targets output performance when running in parallel. XIOS provides the possibility to specify  $N$  dedicated I/O servers (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.

XIOS can take advantage of the parallel I/O functionality of NetCDF4\* to have each XIOS server write to a single output file. This facility is ideal for small to moderate size configurations but can be problematic with large models due to the large memory requirements and the inability to use NetCDF4's compression capabilities in this "one\_file" mode.

XIOS2 has the option of using two levels of I/O servers so it may be possible, in some circumstances, to use a single I/O server at the second level to enable compression. In many cases though, it is often more robust to use "multiple\_file" mode (where each XIOS server writes to a separate file) and to recombine these files as a post-processing step. The `REBUILD_NEMO` tool is provided for this purpose. As the number of XIOS servers is typically much less than the number of *NEMO* processes, significantly fewer output files will be generated compared to the legacy method (which outputs one file per *NEMO* process), reducing the overhead of this post-processing step. For smaller configurations this post-processing step can be avoided entirely, even without a parallel-enabled NetCDF4 library, by using only one XIOS server.

XIOS3 provides a more versatile approach with the concept of "pools and services" where different "pools" of XIOS processes can be assigned to perform different services. Chief amongst these services are "gatherers" and "writers" which are similar to the two-level server capabilities of XIOS2, but, crucially, allow the user better control over the assignment of resources. With care, it is possible to achieve sustained "one\_file" output even with large models. These capabilities are relatively new at the time of the 5.0 release, but the approach is documented in the [XIOS3 demonstrator](#) of the on-line user guide.

### 12.2.1. Main XIOS configuration file (*iodef.xml*)

The behaviour of XIOS is controlled by settings in external XML configuration files, with settings for different applications (or components of one) split into separate "contexts". These settings are specified via the top-level *iodef.xml* file (see for example `./cfgs/ORCA2_ICE_PISCES/EXPREF/iodef.xml`). Basic details on XML syntax and rules can be found in [subsection 12.2.3](#).

In *NEMO*, the *iodef.xml* file typically contains settings for the "xios" context (controlling the overall functionality of XIOS) and for one or more "nemo" contexts (defining the fields and grids used, as well as the output files to be generated, by *NEMO* and any AGRIF child grids). Further information on these contexts can be found in [subsubsection 12.2.3](#).

#### The "xios" context

"xios" context settings that might commonly be configured are shown in [table 12.1](#).

#### The "nemo" context

"nemo" context settings are usually separated into several XML files, each handling a different component of the configuration. These files are included in *iodef.xml* via a nested set of `src` directives (see [subsubsection 12.2.3](#)),

\*This requires that your NetCDF4 library is linked to an HDF5 library that has been correctly compiled (*i.e.* with the configure option `--enable-parallel`)



usually via an intermediate file `context_nemo.xml`. e.g. for the ORCA2\_ICE\_PISCES reference configuration, this hierarchy of files can be represented as:

```
iodef.xml <----- <context id="nemo" src="./context_nemo.xml"/>
context_nemo.xml <--+
    |
    +-- <field_definition src="./field_def_nemo-oce.xml"/>
    +-- <field_definition src="./field_def_nemo-ice.xml"/>
    +-- <field_definition src="./field_def_nemo-pisces.xml"/>
    |
    +-- <file_definition src="./file_def_nemo-oce.xml"/>
    +-- <file_definition src="./file_def_nemo-ice.xml"/>
    +-- <file_definition src="./file_def_nemo-pisces.xml"/>
    |
    +-- <axis_definition src="./axis_def_nemo.xml"/>
    |
    +-- <domain_definition src="./domain_def_nemo.xml"/>
    |
    +-- <grid_definition src="./grid_def_nemo.xml"/>
```

The purposes and contents of these XML files will be explained further in later sections.

## 12.2.2. 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 then need to link to the compiled XIOS library- see the [NEMO user guide](#).

### Attached or detached mode?

For both XIOS2 and XIOS3, a key setting in the "xios" context (`iodef.xml`) is:

```
<variable id="using_server" type="bool"></variable>
```

which 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 output produced will also depend on the type of each file requested in the `file_definition` sections. The type can be either "multiple\_file" or "one\_file" (explained more fully in later sections).

In *attached mode* and if the type of file is "multiple\_file", then each *NEMO* process will also act as an I/O server and produce its own set of output files. Superficially, this emulates the standard behaviour of *NEMO* without XIOS. However, the subdomain written out by each process does not correspond to the `jpj x jpj` 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.

### Number of cpus used by XIOS in detached mode

The number of cores used by the XIOS servers is specified when launching the model. This number should be from  $1/10$  to  $1/50$  of the number of cores dedicated to *NEMO*. Some manufacturers suggest using  $O(\sqrt{N})$  dedicated I/O 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 using 2 cpus for XIOS servers and 62 cpus for *NEMO* with `mpirun`:



```
mpirun -np 62 ./nemo.exe : -np 2 ./xios_server.exe
```

## Add your own outputs

It is very easy to add your own outputs with XIOS and the *NEMO* `iom_put` interface. 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 (step 4). To add new diagnostics, all 4 of the following steps must be taken.

1. In the *NEMO* code, add a `"CALL iom_put( 'identifier', array )"` for the array that is to be output as a diagnostic. In most cases, this will be in a part of the code which is executed only once per timestep and after the array has been updated for that timestep.

Adding this call simply exposes the array to the XIOS workflow; whether or not (and at which frequency) the corresponding diagnostic is actually output by XIOS will be determined by the contents of the file definition (see step 4).

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 appropriate `./cfgs/SHARED/field_def_nemo-...xml` files, add a definition for your diagnostic to the field definition using the same identifier you used in the *NEMO* Fortran code (see subsequent sections for details of the XML syntax and rules). For example:

```
<field_definition>
  <field_group id="grid_T" grid_ref="grid_T_3D"> <!-- T grid -->
    <field id="identifier" long_name="blabla" />
  </field_group>
</field_definition>
```

This definition must be added to the `field_group` whose reference grid (`grid_ref`) is consistent with the size of the array passed to `iom_put`. The `grid_ref` attribute refers to definitions set in `grid_def_nemo.xml` which, in turn, reference domains and axes defined either in the code (`iom_set_domain_attr` and `iom_set_axis_attr` in `iom.F90`) or in the XML configuration files (`domain_def_nemo.xml` and `axis_def_nemo.xml`). *e.g.* :

```
<grid_definition>
  <grid id="grid_T_3D" >
    <domain domain_ref="grid_T" />
    <axis axis_ref="deptht" />
  </grid>
</grid_definition>
```

Note that if the array passed to `iom_put` is computed within the Surface Boundary Condition module (chapter 7), then the corresponding field definition must be added within the SBC `field_group`, `<field_group id="SBC" ...>`. This is because the array is updated every `nn_fsbc` time steps and the frequency of operations in the SBC `field_group` has been defined accordingly (see `iom_set_field_attr` in `iom.F90`).

4. Finally, add your field to one or more file definitions defined in `file_def_nemo-*.xml` (each corresponding to an output file- again, see the subsequent sections for XML syntax and rules)

```
<file_definition>
  <file_group id="5d" output_freq="5d" output_level="10" enabled=".TRUE."> <!-- 5d files -->
    <file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
      <field field_ref="identifier" />
    </file>
  </file_group>
</file_definition>
```

### 12.2.3. XML fundamentals

This subsection discusses some basic aspects of the XML syntax used by XIOS. Further information can be found in the XIOS reference and user guides available [here](#).

Table 12.2.: Hierarchy of scopes used by tags in the XIOS XML configuration files

Scope	description	example
root	declaration of the root element that can contain element groups or elements	<code>&lt;file_definition ... &gt;</code>
group	declaration of a group element that can contain element groups or elements	<code>&lt;file_group ... &gt;</code>
element	declaration of an element that can contain elements	<code>&lt;file ... &gt;</code>

Table 12.3.: XIOS contexts used by NEMO

context	description	example
xios	context containing information for XIOS	<code>&lt;context id="xios" ... &gt;</code>
nemo	context containing I/O information for NEMO (mother grid when using AGRIF)	<code>&lt;context id="nemo" ... &gt;</code>
n_nemo	context containing I/O information for NEMO child grid n (when using AGRIF)	<code>&lt;context id="n_nemo" ... &gt;</code>

## 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: `<a b="x" c="y" d="z"> ... </a>`. See [here](#) for more details.

## XML tags

The XML tags used by XIOS are organised into 7 families: `context`, `axis`, `domain`, `grid`, `field`, `file` and `variable`. Each tag family has a hierarchy of three scopes (except for `context`), shown in [table 12.2](#).

Each element may have several attributes. Some attributes are mandatory, some are optional with a default value, and others are completely optional. A special attribute, `id`, is used to identify an element (or a group of elements) and must have a unique value within each element family. This attribute is optional, but the corresponding element cannot be referenced if this is not defined.

XIOS "contexts" (definitions and settings for different applications or components of one) are separated by the `context` tag. No interference is possible between 2 different contexts. Each context has its own calendar and an associated timestep. The contexts used by NEMO (which can be defined in any order) are shown in [table 12.3](#). The "xios" context uses only 1 tag ([table 12.4](#)), while the other contexts related to NEMO use 5 tags ([table 12.5](#)).

## Nesting XML files

The main XML file (`iodef.xml`) can be split into different parts to improve its readability. These other XML files can then be included in the `iodef.xml` file via the `src` attribute:

```
<context id="nemo" src="./context_nemo.xml"/>
```

In the NEMO reference configurations, the field, file and grid definitions are typically split over several XML files in this manner. *e.g.* the `context_nemo.xml` file for AGRIF\_DEMO contains:

```
<!-- Fields definition -->
<field_definition src="./field_def_nemo-oce.xml" /> <!-- NEMO ocean dynamics -->
<field_definition src="./field_def_nemo-ice.xml" /> <!-- NEMO ocean sea ice -->
<field_definition src="./field_def_nemo-pisces.xml" /> <!-- NEMO ocean biogeochemical -->
```

Table 12.4.: XIOS tags used by the xios context

context tag	description	example
<code>variable_definition</code>	define variables needed by XIOS. This can be seen as a kind of namelist for XIOS.	<code>&lt;variable_definition ... &gt;</code>

Table 12.5.: XIOS tags used by the nemo contexts (both mother and child grids when using AGRIF)

context tag	description	example
field_definition	define all variables that can potentially be outputted	<code>&lt;field_definition ... &gt;</code>
file_definition	define the netcdf files to be created and the variables they will contain	<code>&lt;file_definition ... &gt;</code>
axis_definition	define vertical axis	<code>&lt;axis_definition ... &gt;</code>
domain_definition	define the horizontal grids	<code>&lt;domain_definition ... &gt;</code>
grid_definition	define the 2D and 3D grids (association of an axis and a domain)	<code>&lt;grid_definition ... &gt;</code>

```

<field_definition src="./field_def_nemo-innerttrc.xml"/> <!-- NEMO ocean inert passive tracer -->

<!-- Files definition -->
<file_definition src="./file_def_nemo-oce.xml"/> <!-- NEMO ocean dynamics -->
<file_definition src="./file_def_nemo-ice.xml"/> <!-- NEMO ocean sea ice -->
<file_definition src="./file_def_nemo-innerttrc.xml"/> <!-- NEMO ocean inert passive tracer -->

<!-- Grids/domains/axes definition -->
<axis_definition src="./axis_def_nemo.xml"/> <!-- Axis definition -->
<domain_definition src="./domain_def_nemo.xml"/> <!-- Domain definition -->
<grid_definition src="./grid_def_nemo.xml"/> <!-- Grids definition -->

```

## 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 their parent, and an attribute defined in a child replaces the inherited attribute value. Note that the special attribute `id` is never inherited.

Example 1: direct inheritance.

```

<field_definition operation="average" >
  <field_group id="grid_T" grid_ref="grid_T_2D"> <!-- T grid -->
    <field id="sst" /> <!-- averaged sst -->
    <field id="sss" operation="instant"/> <!-- instantaneous sss -->
  </field_group>
</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: inherit (and overwrite, if needed) the attributes of a tag you are referring to:

```

<field_definition>
  <field_group id="grid_T" grid_ref="grid_T_2D"> <!-- T grid -->
    <field id="sst" long_name="sea surface temperature" />
    <field id="sss" long_name="sea surface salinity" />
  </field_group>
</field_definition>

<file_definition>
  <file id="myfile" output_freq="1d" />
  <field field_ref="sst" /> <!-- default -->
  <field field_ref="sss" long_name="my description" /> <!-- overwritten -->
</file>
</file_definition>

```

## Use of groups

Groups can be used for 2 purposes. Firstly, a 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 2D and 3D fields on the T grid:

```

<field_definition>
  <field_group id="grid_T" grid_ref="grid_T_2D">
    <field id="toce" long_name="temperature" unit="degC" grid_ref="grid_T_3D"/>
    <field id="sst" long_name="sea surface temperature" unit="degC" />
    <field id="sss" long_name="sea surface salinity" unit="psu" />
    <field id="ssh" long_name="sea surface height" unit="m" />
  </field_group>
</field_definition>

```

Most of the fields are 2D, so the 2D grid definition ("grid\_T\_2D") is used by the group. Field "toce" is 3D, so the 2D grid definition inherited from the group is overwritten by that of the 3D grid ("grid\_T\_3D").

Secondly, a group can be used to refer to multiple elements with a single reference. Several examples of groups of fields are included at the end of the field definition XML configuration 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 shortlist of variables on the U grid:

```

<field_definition>
  <field_group id="groupU" >
    <field field_ref="uoce" />
    <field field_ref="ssu" />
    <field field_ref="utau" />
  </field_group>
</field_definition>

```

can be included in a file definition via the `group_ref` attribute:

```

<file_definition>
  <file id="myfile_U" output_freq="1d" />
  <field_group group_ref="groupU" />
  <field field_ref="uocetr_eff" /> <!-- add another field -->
</file>
</file_definition>

```

## 12.2.4. Detailed functionalities

This subsection discusses some of the functionality offered by XIOS, several examples of which can be found within the files ./cfgs/ORCA2\_ICE\_PISCES/EXPREF/\*.xml. Again, refer to the XIOS reference and user guides, available [here](#), for more information.

### Define horizontal subdomains/zooms

Horizontal subdomains ("zooms") are defined through the attributes `ibegin`, `jbegin`, `ni`, `nj` of the `zoom_domain` tag. This must appear within a `domain` tag, and must therefore be placed in the domain definition part of the XML (*i.e.* between the `domain_definition` tags in ./cfgs/SHARED/domain\_def\_nemo.xml).

Note that **zoom\_domain is deprecated in XIOS3** and will eventually be removed; `extract_domain` should be used instead. XIOS3 still supports the use of `zoom_domain`, but will generate warnings stating that this has been renamed to `extract_domain`.

For example, a 5 by 5 box with the bottom left corner at point (10,10) would be defined as:

```

<domain_definition>
  <domain id="myzoomT" domain_ref="grid_T">
    <zoom_domain ibegin="10" jbegin="10" ni="5" nj="5" />
  </domain>
</domain_definition>

```

and would then be used for diagnostic output via the `domain_ref` attribute of the `field` tag family, *e.g.*

```

<file_definition>
  <file id="myfile_zoom" output_freq="1d" >
    <field field_ref="toce" domain_ref="myzoomT"/>
  </file>
</file_definition>

```

However, only `grid_ref` or a `domain_ref/axis_ref` pair may be specified, not both. In the example above, field "toce" has likely been defined with `grid_ref="grid_T_3D"` in the field definition XML configuration file. This will be inherited, so we must override `grid_ref` instead of `domain_ref` by defining a new grid (a copy of "grid\_T\_3D" in ./cfgs/SHARED/grid\_def\_nemo.xml):

```
<grid_definition>
  <grid id="grid_T_3D_myzoomT">
    <domain domain_ref="myzoomT" />
    <axis axis_ref="deptht" />
  </grid>
</grid_definition>
```

and then referencing this in the `field` tag:

```
<file_definition>
  <file id="myfile_zoom" output_freq="1d" >
    <field field_ref="toce" grid_ref="grid_T_3D_myzoomT"/>
  </file>
</file_definition>
```

Moorings are seen as an extreme case corresponding to a 1 by 1 subdomain. The Equatorial section, the TAO, RAMA and PIRATA moorings are already defined in the code and can therefore be used without needing to specify their (i,j) position in the grid. These predefined zooms can be activated by the use of a specific `domain_ref`: "EqT", "EqU" or "EqW" for the equatorial sections and the mooring position for TAO, RAMA and PIRATA followed by "T", *e.g.*

```
<file_definition>
  <file id="myfile_zoom" output_freq="1d" >
    <field field_ref="sst" domain_ref="0n180wT"/>
  </file>
</file_definition>
```

A full list of these section and mooring domains can be found in `./cfigs/SHARED/domain_def_nemo.xml`.

As noted in section 12.2, using "multiple\_file" type output will produce one file per XIOS server with each file containing a different part of the full domain, which may split the subdomain across several files. In this case, tools like `REBUILD_NEMO` should be used to combine these files.

## Define vertical zooms

Vertical zooms are defined through the attributes `begin` and `n` of the `zoom_axis` tag. This must appear within an `axis` tag, and must therefore be placed in the axis definition part of the XML (*i.e.* between the `axis_definition` tags in `./cfigs/SHARED/axis_def_nemo.xml`). Note that as for `zoom_domain`, **zoom\_axis is deprecated in XIOS3** and `extract_axis` should be used instead.

For example, a zoom corresponding to the top 300m of the ocean would be defined as:

```
<axis_definition>
  <!-- Vertical zoom for a 31-levels ORCA2 grid. For eORCA1 300m corresponds to n=35 -->
  <axis id="deptht300" axis_ref="deptht" >
    <zoom_axis begin="0" n="19" />
  </axis>
</axis_definition>
```

and would then be used for diagnostic output via the `axis_ref` attribute of the `field` tag family, *e.g.*

```
<file_definition>
  <file id="myfile_zoom" output_freq="1d" >
    <field field_ref="diag_id" axis_ref="deptht300"/>
  </file>
</file_definition>
```

As noted in the previous section, only `grid_ref` or a `domain_ref/axis_ref` pair may be specified, not both. Therefore in the case of a 3D diagnostic, we must override `grid_ref` instead of `axis_ref` by defining a new grid (a copy of `grid_T_3D` in `./cfigs/SHARED/grid_def_nemo.xml`):

```
<grid_definition>
  <grid id="grid_T_3D_0_300m">
    <domain domain_ref="grid_T" />
    <axis axis_ref="deptht300" />
  </grid>
</grid_definition>
```

and then referencing this in the `field` tag:

```
<file_definition>
  <file id="myfile_zoom" output_freq="1d" >
    <field field_ref="toce" grid_ref="grid_T_3D_0_300m"/>
  </file>
</file_definition>
```

Table 12.6.: Placeholder strings for the names of diagnostic output files generated by XIOS and the strings they are substituted for, when the file id has the form "fileN"

Placeholder string	Automatically replaced by
@expname@	The experiment name (from <code>cn_exp</code> in the namelist)
@freq@	Output frequency (from XML attribute <code>output_freq</code> )
@startdate@	Starting date of the simulation (from <code>nn_date0</code> in the restart or the namelist). yyyymmdd format
@startdatefull@	Starting date of the simulation (from <code>nn_date0</code> in the restart or the namelist). yyyymmdd_hh:mm:ss format
@enddate@	Ending date of the simulation (from <code>nn_date0</code> and <code>nn_itend</code> in the namelist). yyyymmdd format
@enddatefull@	Ending date of the simulation (from <code>nn_date0</code> and <code>nn_itend</code> in the namelist). yyyymmdd_hh:mm:ss format

### Changes to the names of output files applied by *NEMO*

The output file names are defined by the attributes `name` and `name_suffix` of the `file` tag family. For example:

```
<file_definition>
  <file_group id="1d" output_freq="1d" name="myfile_1d" >
    <file id="myfileA" name_suffix="AAA" > <!-- will create file "myfile_1d_AAA" -->
    ...
  </file>
  <file id="myfileB" name_suffix="BBB" > <!-- will create file "myfile_1d_BBB" -->
  ...
  </file>
</file_group>
</file_definition>
```

However it is often very convenient to include the name of the experiment, the output file frequency and the start/end dates of the simulation in the file name, which are stored either in the namelist or in the XML file. To achieve this, we added the following rule: if the `id` of the `file` tag is "fileN" (where N = 1 to 999 on 1 to 3 digits) or one of the predefined sections or moorings (see next subsection), parts of the `name` and `name_suffix` attributes (which can be inherited) will be automatically replaced if they correspond to any of the placeholders in table 12.6.

For example,

```
<file_definition>
  <file id="file66" name="myfile_@expname@_@startdate@_freq@freq@" output_freq="1d" >
</file_definition>
```

with the namelist:

```
cn_exp    = "ORCA2"
nn_date0  = 19891231
ln_rstart = .false.
```

will give the following file name radical: `myfile_ORCA2_19891231_freq1d`

### Other XML attributes set by *NEMO*

The values of some XML attributes (including `name_suffix`, discussed in the previous subsection) are automatically set by the `set_xmlatt` subroutine in *NEMO* (`iom.F90`). These attributes and their values are given in table 12.7. Any definition of these attributes in the XML files will be overwritten; by convention their values are set to "auto" (for strings) or "0000" (for integers), although this is not necessary.

### Advanced use of XIOS functionalities

XIOS can do far more than just gather and write output. Importantly, it can perform computations with the fields it receives providing opportunities to create derived quantities without burdening the model simulation. This section provides a few illustrations of the possibilities:

1. Using algebraic expressions

A new diagnostic can be derived from existing diagnostics, either in the file definition:

Table 12.7.: XIOS XML attributes that are set automatically by *NEMO*, excluding *name* and *name\_suffix*

Tag family and id affected by automatic definition of some of their attributes	Attribute name	Attribute value
field_definition	freq_op	rn_rdt
field: SBC, SBC_scalar, ABL	freq_op	rn_rdt × nn_fsbc
field: trendT_even	freq_op	2× rn_rdt
field: trendT_odd	freq_op	2× rn_rdt
zoom_domain: EqT, EqU, EqW	freq_offset	-1
zoom_domain: TAO, RAMA and PIRATA moorings	jbegin, ni,	set according to the grid
	ibegin, jbegin,	set according to the grid

```

<file_definition>
  <file id="derived_vars" output_freq="1d" >
    <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" > taum * taum </field>
    <field field_ref="qt" name="stupid_check" > qt - qsr - qns </field>
  </file>
</file_definition>
    
```

or in the field definition:

```

<field_definition>
  <field id="sst2" field_ref="sst" long_name="square of sea surface temperature" unit="degC2" > sst * sst </field>
</field_definition>
    
```

and then referenced in the file definition:

```

<file_definition>
  <file id="derived_vars" output_freq="1d" >
    <field field_ref="sst2" > sst2 </field>
  </file>
</file_definition>
    
```

Note that in this case, simply adding "`<field field_ref="sst2" />`" to the file definition would not work since "sst2" would not be evaluated.

## 2. Use of the "@" function: example 1, weighted temporal average

The "@" function can be used in algebraic expressions to chain temporal operations. In this example, it is used to output a weighted temporal average of the temperature (with the time-varying layer thickness as the weight).

The product of the two quantities is first added as a new variable in the field definition:

```

<field_definition operation="average" freq_op="1ts">
  <field id="toce_e3t" long_name="temperature * e3t" unit="degC*m" grid_ref="grid_T_3D" >toce * e3t</field>
</field_definition>
    
```

The `operation="average"` and `freq_op="1ts"` attributes specify the temporal operation and its sampling frequency- `toce` and `e3t` will be used to calculate `toce_e3t` for every timestep, which will then be averaged over a time period set by the `output_freq` or `freq_op` attributes (the latter is given priority) in the file definition. For example:

```

<file_definition>
  <file_group id="5d" output_freq="5d" output_level="10" enabled=".true." > <!-- 5d files -->
    <file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
      <!-- 5-day averages -->
      <field field_ref="toce" />
      <!-- 1-day averages, output once every 5 days -->
      <field field_ref="toce" freq_op="1d" name="toce_1d" />
    </file>
  </file_group>
</file_definition>
    
```

To produce a 5-day weighted average, the 5-day average of the weighted temperature (`@toce_e3t`) must be divided by that of the layer thickness (`@e3t`):



```

<file_definition>
  <file_group id="5d" output_freq="5d" output_level="10" enabled=".true." > <!-- 5d files -->
    <file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
      <field field_ref="toce" operation="instant" freq_op="5d" > @toce_e3t / @e3t </field>
    </file>
  </file_group>
</file_definition>

```

Normally, the `operation="average"` and `freq_op="1ts"` attributes inherited from the field definition would be overwritten by the `operation="instant"` and `freq_op="5d"` attributes in the file definition. This would result in instantaneous output (data for one timestep) every 5 days.

The “@” function overrides this behaviour so that instead, the temporal operations are applied separately. Specifically, it indicates that the temporal operation for the adjacent field should be performed before evaluating the algebraic expression it is part of. This results in 2 chained temporal operations:

- Temporal operation 1: the operation type and sampling frequency are set by the `operation` and `freq_op` attributes in the field definition, while the temporal period of the operation is set by the `freq_op` attribute in the file definition.

- Temporal operation 2: the operation type, sampling frequency and temporal period are specified by the `operation`, `freq_op` and `output_freq` attributes in the file definition.

For the above thickness-weighted temperature example, the following operations occur in order:

- `toce_e3t` is calculated from `toce` and `e3t` for every timestep
- 5-day averages of `toce_e3t` and `e3t` are calculated (temporal operation 1)
- The weighted average, `toce_e3t / e3t`, is calculated using these 5-day averages
- The instantaneous value of this expression is output every 5 days (temporal operation 2)

The last of these (the 2nd temporal operation) simply returns the result of the 3rd operation- a 5-day weighted average every 5 days. One could equivalently specify `operation="average"` in the file definition and get the same result, although the time coordinate for the diagnostic would be that of an average (with values of 2.5, 7.5, ... days) rather than an instantaneous quantity (with values of 5, 10, ... days).

### 3. Use of the “@” function: example 2, monthly SSH standard deviation

The square of the SSH is added as a new variable in the field definition:

```

<field_definition operation="average" freq_op="1ts">
  <field id="ssh2" long_name="square of sea surface temperature" unit="degC2" > ssh * ssh </field>
</field_definition>

```

In the file definition, monthly averages of this variable and `ssh` are then calculated and used to calculate the monthly standard deviation:

```

<file_definition>
  <file_group id="1m" output_freq="1m" output_level="10" enabled=".true." > <!-- 1m files -->
    <file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
      <field field_ref="ssh" name="sshstd" long_name="sea surface temperature standard deviation"
        operation="instant" freq_op="1m" >
        sqrt( @ssh2 - @ssh * @ssh )
      </field>
    </file>
  </file_group>
</file_definition>

```

In this example, the following operations occur in order:

- `ssh2` is calculated from `ssh` for every timestep
- 1-month averages of `ssh2` and `ssh` are calculated (temporal operation 1)
- The standard deviation, `sqrt(ssh2 - ssh * ssh)`, is calculated using these 1-month averages
- The instantaneous value of this expression is output every month (temporal operation 2)

### 4. Use of the “@” function: example 3, monthly average of SST diurnal cycle

The temporal minimum and maximum of the SST are added as new variables in the field definition:

```

<field_definition operation="average" freq_op="1m">
  <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" />
</field_definition>

```

In the file definition, these variables are then evaluated over a 1-day period and used to calculate the diurnal amplitude of the SST and its monthly average:

```

<file_definition>
  <file_group id="1m" output_freq="1m" output_level="10" enabled=".true." > <!-- 1m files -->
    <file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
      <field field_ref="sst" name="sstdcy" long_name="amplitude of sst diurnal cycle"
        operation="average" freq_op="1d" >
        @sstmax - @sstmin
      </field>
    </file>
  </file_group>
</file_definition>

```

In this example, the following operations occur in order:

- Daily minima (`sstmin`) and maxima (`sstmax`) of `sst` are calculated (temporal operation 1)
- The amplitude, `sstmax - sstmin`, is calculated using these daily extrema
- The monthly average of this expression is calculated and output every month (temporal operation 2)

#### 5. Changing variable precision

Diagnostic output precision can be modified with the `prec` attribute of the field tag family. Data packing is also supported via the `add_offset` and `scale_factor` attributes.

```

<!-- 64-bit (8-byte) float -->
<field field_ref="sst" name="tos_r8" prec="8" />
<!-- Packing to 16-bit (2-byte) integer -->
<field field_ref="sss" name="sos_i2" prec="2" add_offset="20." scale_factor="1.e-3" />

```

If the data cannot be converted to the target precision, XIOS will crash with a "NetCDF: Numeric conversion not representable" error. In the case of single-precision floating point diagnostics (`prec="4"`), this often happens when *NEMO* has sent XIOS data containing NaNs or very large/small values, which can result from *e.g.* floating point calculation errors. Forcing double-precision output (`prec="8"`) may bypass the XIOS crash, but it is usually better to inspect and troubleshoot the diagnostic data being sent from *NEMO*.

#### 6. Adding user-defined NetCDF file attributes

User-defined NetCDF attributes can be added to the output file metadata at the global and variable levels:

```

<file_definition>
  <file id="file1" name_suffix="_grid_T" description="ocean T grid variables" >
    <!-- Variable attributes -->
    <field field_ref="sst" name="tos" >
      <variable id="my_attribute1" type="string" > blabla </variable>
      <variable id="my_attribute2" type="integer" > 3 </variable>
      <variable id="my_attribute3" type="float" > 5.0 </variable>
    </field>
    <!-- Global attributes -->
    <variable id="my_global_attribute" type="string" > blabla_global </variable>
  </file>
</file_definition>

```

### 12.2.5. CF metadata standard compliance

Output from XIOS 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 3 of [subsection 12.2.4](#)) the metadata should, for the most part, comply with the CF-1.5 standard.

Some metadata required for full compliance with the CF standard (horizontal cell areas and vertices) are not output by default. It can be output by setting `ln_cfmeta=.true.` in the `&namrun` ([namelist 2.1](#)) namelist, but note that it will be added to all files with variables on the horizontal domain, which may significantly increase the file size.

### 12.2.6. Enabling NetCDF4 compression with XIOS

XIOS supports the use of gzip compression when compiled with NetCDF4 libraries but is subject to the same restrictions as the underlying HDF5 component: compression is not available when the XIOS servers are writing in parallel to a single output file. Thus, compression can only be applied in "multiple\_file" mode only, or with two levels of servers using multiple level 1 servers and a single level 2 server. Compression is activated by using the `compression_level` attribute of the `field` or `file` tag families:

```
<file_definition>
  <file name="output" output_freq="1ts" compression_level="2">
    <field id="field_A" grid_ref="grid_A" operation="average" compression_level=" 4" />
    <field id="field_B" grid_ref="grid_A" operation="average" compression_level=" 0" />
    <field id="field_C" grid_ref="grid_A" operation="average" />
  </file>
</file_definition>
```

Its value is an integer between 0 and 9. A value of 2 is normally recommended as a suitable trade-off between algorithm performance and compression levels.

It is unclear how XIOS2 decides on suitable chunking parameters before applying compression, so it may be necessary to re-chunk data when combining files produced with the "multiple\_file" output mode. The [REBUILD\\_NEMO](#) tool is capable of doing this. With XIOS3, the user is provided with more control over the chunking but the relationship between input settings and final chunk sizes is complex. See the [XIOS3 demonstrator](#) section of the user guide for an illustration.

## 12.3. Reading and writing restart files

From *NEMO* v4.2, XIOS may be used to read in a single-file restart dump produced by *NEMO*. This does not add new functionality (*NEMO* has long had the capability for all processes to read their subdomain from a single, combined restart file) but it may be advantageous on systems which struggle with too many simultaneous accesses to one file. The variables written to files associated with the logical units `numror` (OCE), `numrir` (SI3), `numrtr` (TOP) and `numrsr` (SED) can be handled by XIOS.

The use of XIOS to read restart files is activated by setting `ln_xios_read=.true.` in `&namcfg` ([namelist 17.1](#)). This setting will be ignored when multiple restart files are present, and default *NEMO* functionality will instead be used for reading.

The `iodef.xml` XIOS configuration file does not need to be changed to use this functionality, as all definitions are implemented within the *NEMO* code as a separate XIOS context. For high resolution configurations, however, there may be a need to add the following line in `iodef.xml`:

```
<variable_definition>
  <variable id="recv_field_timeout" type="double">1800</variable>
</variable_definition>
```

which sets the timeout period for reading data.

If XIOS is to be used to read from restart files generated with an earlier *NEMO* version (3.6 for instance), the dimension `z` defined in the restart file must be renamed to `nav_lev`.

XIOS can also be used to write *NEMO* restarts. The namelist parameter `nn_wxios` is used to determine the type of restart *NEMO* will write:

`nn_wxios=0`

Default functionality: each *NEMO* process writes its own restart file

`nn_wxios=1`

XIOS will write to a single restart file

`nn_wxios=2`

XIOS will write to multiple restart files, one per server

This option aims to reduce the number of restart files generated by *NEMO*, and may be useful when there is a need to change the number of processors used to run the simulation. Note that ***NEMO will not be able to read the restart files generated by XIOS with `nn_wxios=2`***. These files will have to be combined (with *e.g.* [REBUILD\\_NEMO](#)) before continuing the run.

The use of XIOS to read and write restart files is in preparation for running *NEMO* on exascale computing platforms. While this may not yield significant performance gains on current clusters, it should reduce file system bottlenecks in future attempts to run *NEMO* on hundreds of thousands of cores.

```

!-----
&namnc4      ! netcdf4 chunking and compression settings
!-----
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     = .false. ! (T) use netcdf4 chunking and compression
!                  ! (F) ignore chunking and compression information (netcdf3 compatible file)
/

```

namelist 12.1.: &namnc4

## 12.4. NetCDF4 support (legacy output file method)

As of *NEMO* v5, the legacy output method (where diagnostic and/or restart files are written by *NEMO* using the old IOIPSL interface, rather than by XIOS) only supports NetCDF4 (version 4.1 and later are recommended) built with HDF5 (version 1.8.4 and later are recommended). This allows chunking and (loss-less) compression, which can achieve a significant reduction in file size 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](#).

Datasets created with chunking and compression are not backwards compatible with the NetCDF3 "classic" format, but most analysis codes can simply be relinked with the NetCDF4 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 `ln_nc4zip=.false.` in the `&namnc4` ([namelist 12.1](#)) namelist.

Chunking and compression are applied only to 4D fields and there is no advantage in chunking across more than one time dimension, since previously written chunks would have to be read back and decompressed before being added to. Therefore, user control over chunk sizes is provided only for the three spatial dimensions. The user sets an approximate number of chunks along each spatial axis. The actual size of the chunks will depend on the global domain size for mono-processors and 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
nn_nchunks_k=31

```

for a standard ORCA2\_ICE\_PISCES configuration (with a global domain of 182x149x31) gives chunk sizes of 46x38x1 respectively in the mono-processor case. An illustration of the potential space savings that NetCDF4 chunking and compression provides is given in [table 12.8](#), which compares the results of two short runs of the deprecated ORCA2\_LIM reference configuration (now the ORCA2\_ICE\_PISCES configuration) with a 4x2 MPI decomposition. Note the variation in the compression ratio achieved, which reflects chiefly the dry to wet volume ratio of each processing region.

Note that chunking and compression can also be applied when combining output files with *e.g.* `REBUILD_NEMO`.

## 12.5. Tracer/Dynamics trends ( &namtrd ([namelist 12.2](#)) , &namtrc\_trd ([namelist 12.3](#)) )

Each trend of the time evolution equations for the dynamics ( *trddyn.F90* ) and both active ( *trdtra.F90* ) and passive ( *trdtrc.F90* ) tracers can be output following their computation, via calls to the `trd_tra` (active and passive tracers), `trd_dyn` (dynamics) and `trd_trc` (passive tracers) subroutines.

The output of trends diagnostics for the dynamics and active tracers is controlled by parameters in the `&namtrd` ([namelist 12.2](#)) namelist:

```
ln_glo_trd=.true.
```

Every `nn_trd` time-steps, 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.

Table 12.8.: Filesize comparison between NetCDF3 and NetCDF4 with chunking and compression

Filename	NetCDF3	NetCDF4	Reduction
	filesize (KB)	filesize (KB)	
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%

```

!-----
&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 12.2.: &namtrd

```

!-----
&namtrc_trd  !  diagnostics on tracer trends          ('key_trdtrc')
!             or mixed-layer trends                 ('key_trdml_d_trc')
!-----
nn_trd_trc   = 5475     ! time step frequency and tracers trends
nn_ctls_trc  = 0        ! control surface type in mixed-layer trends (0,1 or n<jpk)
rn_ucf_trc   = 1        ! unit conversion factor (=1 -> /seconds ; =86400. -> /day)
ln_trdml_d_trc_restart = .false. ! restart for ML diagnostics
ln_trdml_d_trc_instant = .true.  ! flag to diagnose trends of instantaneous or mean ML T/S
ln_trdtrc( 1) = .true.
ln_trdtrc( 2) = .true.
ln_trdtrc(23) = .true.
/
    
```

namelist 12.3.: &namtrc\_trd

```

!-----
&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 12.4.: &nam\_diadct

**ln\_dyn\_trd=.true.**

Each 3D trend of the evolution of the two momentum components is output.

**ln\_dyn\_mxl=.true.** (currently not working)

Each 3D trend of the evolution of the two momentum components averaged over the mixed layer is output.

**ln\_vor\_trd=.true.** (currently not working)

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=.true.**

Each 3D trend of the Kinetic Energy equation is output.

**ln\_PE\_trd=.true.** (currently not working with nonlinear free surface)

Each 3D trend of the Potential Energy equation is output.

**ln\_tra\_trd=.true.**

Each 3D trend of the evolution of temperature and salinity is output.

**ln\_tra\_mxl=.true.** (currently not working)

Each 2D trend of the evolution of temperature and salinity averaged over the mixed layer is output.

while the output of trends diagnostics for the passive tracers is controlled by parameters in the **&namtrc\_trd** (namelist 12.3) namelist.

As all 3D trends are output using XIOS, **key\_xios** must generally be specified. Additionally, the passive tracer trends require **key\_trdtrc** (for 3D trends) and/or **key\_trdmxl\_trc** (for 2D trends averaged over the mixed layer) to be specified.

Note that currently, **the trends diagnostics are not fully functional or tested** and a warning will be raised if they are used. In particular, the code associated with the **ln\_dyn\_mxl**, **ln\_vor\_trd**, and **ln\_tra\_mxl** namelist options is not working and an error will be raised if they are used.

## 12.6. Transports across sections

Diagnostics to compute the transport of volume, heat and salt through sections can be activated by setting **ln\_diadct=.true.** in the **&nam\_diadct** (namelist 12.4) namelist. Each section is defined by the coordinates of its 2 extremities. The pathways between them are constructed using the **SECTIONS\_DIALECT** tool and are written to a binary file *section\_iglobal.diadct*, which is later read in by *NEMO* to compute on-line transports.

The on-line transports module ( *diadct.F90* ) outputs three ascii files:

- *volume\_transport* for volume transports (unit:  $10^6 \text{ m}^3 \text{ s}^{-1}$ )
- *heat\_transport* for heat transports (unit:  $10^{15} \text{ W}$ )
- *salt\_transport* for salt transports (unit:  $10^9 \text{ Kgs}^{-1}$ )

Namelist variables in the **&nam\_diadct** (namelist 12.4) namelist 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 computation of the transports (in time steps)

**nn\_dctwri**

Averaging period of the transports (as a frequency, in time steps)

**nn\_secdebug**

Sections to debug:

- 0 - Do not debug any sections
- 1 - Debug all sections
- n - Debug section number n

**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 (based on MERSEA project metrics) that are to be computed. Another file is available for the GYRE configuration (`list_sections.ascii_GYRE`).

Each section in this file is defined by a line containing, in order:

**long1 lat1**

Coordinates of the first extremity of the section, *e.g.* -68. -54.5

**long2 lat2**

Coordinates of the second extremity of the section, *e.g.* -60. -64.7

**nclass**

The number of bounds in each class type (`nclass - 1` classes per type), *e.g.* 2

**okstrpond or nostrpond**

A string controlling whether to compute heat and salt transports (`okstrpond`) or not (`nostrpond`)

**ice or noice**

A string controlling whether to compute surface and volume ice transports (`ice`) or not (`noice`)

**section\_name**

The name of the section, *e.g.* ACC\_Drake\_Passage

Note that neither the results of the transport calculations nor the directions of positive and negative flow depend on the order in which the 2 extremities are specified in this file.

If `nclass`  $\neq$  0, the following `nclass + 1` lines contain a class type and its bounds, which may be repeated for several class types. *e.g.* for 2 class types with 2 bounds (1 class per type):

```
long1 lat1 long2 lat2 nclass (ok/no)strpond (no)ice section_name
classtype
bound_1
bound_2
classtype
bound_1
bound_2
```

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`. The top part of this script should be modified for the user's configuration, including setting the name and path of the coordinates file to use.

Examples of two sections, `ACC_Drake_Passage` with no classes, and `ATL_Cuba_Florida` with 4 temperature classes (5 class bounds), are shown:



Table 12.9.: Transport section slope coefficients

Section slope coefficient	Section type	Direction 1	Direction 2	Total transport
0.	Horizontal	Northward	Southward	positive: northward
1000.	Vertical	Eastward	Westward	positive: eastward
$\neq 0, \neq 1000.$	Diagonal	Eastward	Westward	positive: eastward

```

-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
    
```

### Reading the output files

The format of the output file is:

```

date, time-step number, section number,
section name, section slope coefficient, class number,
class name, class bound 1, class bound2,
transport direction 1, transport direction 2,
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 direction 1` is the positive part of the transport ( $\geq 0$ ) and `transport direction 2` is the negative part of the transport ( $\leq 0$ ). `section slope coefficient` gives information about the significance of transports signs and direction (see [table 12.9](#)).

## 12.7. Diagnosing the steric effect on 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:

$$\begin{aligned} \mathcal{M} &= \mathcal{V} \bar{\rho} \\ \mathcal{V} &= \mathcal{A} \bar{\eta} \end{aligned} \tag{12.1}$$

Temporal changes in total mass are obtained from the density conservation equation:

$$\frac{1}{e_3} \partial_t (e_3 \rho) + \nabla(\rho \mathbf{U}) = \frac{emp}{e_3} \Big|_{surface} \quad (12.2)$$

where  $\rho$  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 average leads to the total mass change

$$\partial_t \mathcal{M} = \mathcal{A} \overline{emp} \quad (12.3)$$

where  $\overline{emp} = \int_S emp ds$  is the net mass flux through the ocean surface. Bringing [equation 12.3](#) and the time derivative of [equation 12.1](#) together leads to the evolution equation of the mean sea level

$$\partial_t \bar{\eta} = \frac{\overline{emp}}{\bar{\rho}} - \frac{\mathcal{V}}{\mathcal{A}} \frac{\partial_t \bar{\rho}}{\bar{\rho}} \quad (12.4)$$

The first term in [equation 12.4](#) 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,  $\rho$  is replaced by  $\rho_o$  in all the equation except when  $\rho$  appears multiplied by the gravity (*i.e.* in the hydrostatic balance of the primitive equations). In particular, the mass conservation equation, [equation 12.2](#), 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 other 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,  $\eta_s$ , a spatially uniform variable, as follows:

$$\mathcal{M}_o = \mathcal{M} + \rho_o \eta_s \mathcal{A} \quad (12.5)$$

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 6.8.1](#)) in [equation 12.5](#) leads to a very simple form for the steric height:

$$\eta_s = -\frac{1}{\mathcal{A}} \mathcal{D} \quad (12.6)$$

The above formulation of the steric height of a Boussinesq ocean requires four remarks. First, one can be tempted to define  $\rho_o$  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  $\rho_o$  depends on the initial state of the ocean. Since  $\rho_o$  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 instead recommend to set a fixed value  $\rho_o = 1035 \text{ Kg m}^{-3}$ . 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 12.6](#) depends on the type of free surface which is considered. In the non linear free surface case, 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, *i.e.* when `key_linssh` is specified, 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.

Global averages of both the steric (`sshsteric` diagnostic) and thermosteric (`sshthster` diagnostic) sea level can be output by the AR5 diagnostics module (`diar5.F90`, see subsection 12.9.2). The latter is the steric sea level due to changes in ocean density arising only 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.

When this diagnostic is output, salinity data for  $S_o$  must be provided via a variable named `vosaline` in a file named `sali_ref_clim_monthly.nc`. This data must be provided as a monthly climatology; *i.e.* the file's time coordinate must have a length of 12.

## 12.8. Tidal harmonic and generic multiple-linear-regression analysis ( `diamlr.F90` )

Functionality for multiple-linear-regression (MLR) analysis of arbitrary output fields, using regressors that are a function of the continuous model time, is available as a diagnostic option of *NEMO*. Its implementation makes use of the ordinary-least-squares method (a method overview can be found [here](#)), it depends on XIOS for generating a set of intermediate output files, the set of regressors is configurable as part of the model-output XIOS configuration, and the regression analyses can be completed versatily in a post-processing step. In particular, the analysis time window remains flexible until the post-processing step, from partial model runs (depending on the selected temporal resolution for the intermediate output) to spanning multiple restart segments; also, the original regressor set can be restricted at the post-processing step. For the specific case of tidal harmonic analysis, the configuration of regressors that correspond to tidal constituents available for tidal forcing (see section 7.8) is facilitated through a substitution mechanism (*i.e.* model-provided tidal frequencies, phases, and amplitudes can be referred to symbolically in MLR analysis configurations).

### 12.8.1. Configuration of the multiple-linear-regression analysis

The MLR analysis is activated by defining an empty file-group entry

```
<file_group id="diamlr_files" output_freq="<output frequency>" enabled=".TRUE." />
```

in the XIOS configuration, where `<output frequency>` specifies the temporal resolution of the intermediate output: if defined, this file group will be populated during model initialisation. Other prerequisite XIOS-configuration elements (regressors and the time variable) are pre-defined in the default XIOS configuration file `./cfgs/SHARED/field_def_nemo-oce.xml`, and can be modified if required.

Regressors are defined and enabled within the XIOS field group `<field_group id="diamlr_fields">` in the form of individual fields that are computed from the spatially uniform field `diamlr_time` as

```
<field id="diamlr_r<mmm>" field_ref="diamlr_time" expr="<expression>" enabled=".TRUE." comment="<comment>" /> ,
```

where `<mmm>` is a 3-digit identification number, `<expression>` a functional expression, and `<comment>` an arbitrary string (which may be utilised to pass information to post-processing utilities); field `diamlr_time` contains the continuous model time in seconds. In the functional expression, XIOS requires the specified reference field `diamlr_time` to be included; therefore, in order to obtain a constant expression for fitting an intercept, `diamlr_time^0` can be chosen. The model time `diamlr_time` corresponds to module variable `adatrj` of module `dom_oce.F90` and is defined in module `daymod.F90`; its continuity across model restarts depends

on a selection made in `&namdom` (`namelist 3.2`). Similarly, a XIOS field `<field name>` can be selected for MLR analysis through the definition of a new field

```
<field id="diamlr_f<nnn>" field_ref="<field name>" enabled=".TRUE." />
```

in field group `<field_group id="diamlr_fields">`, where `<nnn>` is a 3-digit identification number.

For the purpose of tidal harmonic analysis, two orthogonal regressors per analysed tidal-constituent signal need to be defined in order to fit both the amplitude and phase of the corresponding harmonic, typically a sine and cosine function with identical argument. Further, regressor configurations can be equipped with placeholders to refer to the frequency, phase, and amplitude of each of the constituents available and evaluated for tidal forcing of the model. In particular:

`__TDE_<constituent>_omega__`

refers to the angular velocity (in units of  $\text{rad s}^{-1}$ );

`__TDE_<constituent>_phase__`

refers to the phase, including the nodel correction at the beginning of the model run (in units of rad); and

`__TDE_<constituent>_amplitude__`

refers to the equilibrium-tide amplitude (in units of m)

of tidal constituent `<constituent>`. During model initialisation, these placeholders are automatically substituted with the corresponding model-computed values for the respective tidal constituent.

A default set of regressors relevant for tidal harmonic analysis has been pre-defined (see `./cfigs/SHARED/field_def_nemo-oce.xml`) and can be redefined. An example of such a redefinition can be found in the AMM12 reference configuration, in file `./cfigs/AMM12/EXPREF/context_nemo.xml`.

### 12.8.2. The intermediate output and its post-processing

Internally, during model initialisation, the initial XIOS configuration for MLR analysis is expanded automatically through the generation of field and output-file definitions for the relevant intermediate model output. The resulting intermediate output consists of fields of scalar products between each regressor and the values of the fields selected for MLR analysis, as well as of scalar products between each regressor-regressor pair, all sampled at the configured interval. For the final analysis only the scalar products over the analysis time span are required, thus the intermediate output can be freely subset or combined (added) along its time dimension to select the analysis window (which enables analyses across multiple restart segments) during post-processing.

The total number of intermediate output variables depends on the number of analysed fields ( $n_f$ ) and the number of regressors ( $n_r$ ) (for tidal analysis,  $n_r = 2n_c + 1$ , *i.e.* twice the number of tidal constituents,  $n_c$ , plus one regressor to fit the intercept) and amounts to  $n_f n_r + 2n_r^2 - n_r$  (of which  $2n_r^2 - n_r$  variables are scalar time series). These output variables are written to output files labelled with `diamlr_scalar`, which contain the regressor-regressor scalar products, and with `diamlr_grid_<grid_type>`, which contain the regressor-diagnostic scalar-product fields for the fields defined on a grid of type `<grid_type>`.

For the computation of regression coefficients from previously generated intermediate output files, the rudimentary script `./tools/DIAMLR/diamlr.py` can be used. This script is provided as a simple example of the final analysis step: it processes suitable intermediate-output files by adding all available time slices and by computing regression coefficients for all available analysed fields and for all or a subset of the regressors identified from the content of the intermediate-output files. To complete a tidal harmonic analysis, the pairs of regression coefficients associated with each of the tidal constituents selected for analysis (the `comment` attribute could be used for identifying such pairs) could in turn be converted into maps of amplitudes and phases.

## 12.9. 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`.

### 12.9.1. Depth of various quantities ( *diahth.F90* )

The following diagnostics are available via the *diahth.F90* module when `key_diahth` is specified:

- the mixed layer depth (based on the density criterion of [de Boyer Montégut et al., 2004](#))
- the turbocline depth (based on a turbulent mixing coefficient criterion)
- the depth of the 20°C isotherm
- the depth of the thermocline (maximum of the vertical temperature gradient)

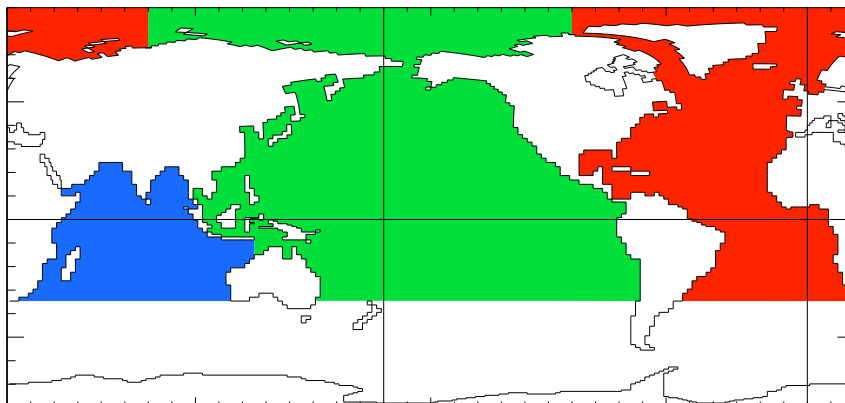


Figure 12.1.: Decomposition of the World Ocean (shown here for the ORCA2 grid) into sub-basins used to compute the heat and salt transports as well as the meridional stream-function: Atlantic basin (red), Pacific basin (green), Indian basin (blue), Indo-Pacific basin (blue+green). Note that semi-enclosed seas (Red, Med and Baltic seas) as well as Hudson Bay are removed from the sub-basins, and that the Arctic Ocean has been split into Atlantic and Pacific basins along the North fold line.

### 12.9.2. CMIP-specific and poleward transport diagnostics ( *diaar5.F90* , *diaptr.F90* )

Diagnostics in the *diaar5.F90* module correspond to outputs that are required for the AR5/CMIP5 simulations (see for example, the thermosteric sea level diagnostic in [section 12.7](#)).

The *diaptr.F90* module computes the online poleward heat and salt transports, their advective and diffusive component, the meridional stream function, and the zonal mean temperature, salinity and cell *i-k* surface area. These are computed by default for the global ocean, but if the file *subbasins.nc* is provided then these diagnostics are also computed for the Atlantic, Indian, Pacific and Indo-Pacific Ocean basins (defined north of 30°S). The *atlmsk*, *indmsk* and *pacmsk* variables in this file are masks corresponding to the Atlantic, Indian and Pacific basins, while the Indo-Pacific basin mask is computed as the union of the Indian and Pacific basin masks ([figure 12.1](#)).

The diagnostics available from both modules are listed in the `./cfgs/SHARED/field_def_nemo-oce.xml` XIOS configuration file: *diaar5.F90* diagnostics are listed under `<!-- variables available with diaar5 -->` comment headers, while *diaptr.F90* diagnostics are listed within the `<field_group id="diaptr" >` element.

### 12.9.3. De-tided diagnostic output from tidal models ( *dia25h.F90* , *diadetide.F90* )

#### 25-hour averages ( *dia25h.F90* )

Modelled fields of potential temperature, salinity, SSH, velocity, vertical diffusivity and viscosity, TKE, and the mixing length can be approximately de-tided by crudely (fully) removing the signal of the M2 (S2) tidal constituent. This operation is carried out by the *dia25h.F90* module by averaging 25 instantaneous values at one-hour intervals that span consecutive periods of 24 model hours (*i.e.* the model state at the boundaries of such sampling windows is accounted for in both adjacent averages). As a consequence of this method of averaging 25 hourly values, the least common multiple of the time-step length `rn_Dt` and 3600 s is required to be 3600 s. This diagnostic is activated when any of the available daily 25-hour-average output fields is selected in the XIOS model-output configuration: `temper25h`, `salin25h`, `ssh25h`, `vozocrtx25h`, `vomecrtx25h`, `vovecrtz25h`, `avt25h`, `avm25h`, `tke25h`, or `mxln25h`.

#### Generic de-tiding of model output ( *diadetide.F90* , experimental)

A more generic alternative to the de-tiding option provided by module *dia25h.F90* is available with module *diadetide.F90*; this option, however, has not been fully developed and well tested, and therefore should be considered to be an *experimental* feature and used with care. Like for the *dia25h.F90* implementation, the current version of this alternative de-tiding option computes daily averages with an averaging window that corresponds to twice the M2 tidal period; in contrast to module *dia25h.F90*, the averaging procedure is more accurate for sufficiently small time steps, the method can be used with an arbitrary time-step length, and it can be applied to analyse arbitrary output fields available for XIOS-based model output.

An example of the application of the de-tiding option provided by module *diadetide.F90* has been included in the AMM12 reference configuration. The corresponding activation can be found in the XIOS file

group `diadetide_files` defined in file `cfgs/AMM12/EXPREF/context_nemo.xml`. Implementation details (in particular the computation of the averaging weights) can be found in module `diadetide.F90`.

#### 12.9.4. 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.

Courant number diagnostics can be activated by setting `ln_diacfl=.true.` in the `&namctl` ([namelist 16.2](#)) namelist. The global maximum values of  $C_u$ ,  $C_v$ ,  $C_w$ , and their coordinates, for each timestep and for the whole model run, are written to an ascii file named `cfl_diagnostics.ascii`. The maximum values for the whole model run are also copied to the `ocean.output` file. Additionally, the depth maxima of  $C_u$ ,  $C_v$ , and  $C_w$  are available as 2D XIOS diagnostics (`cfl_cu`, `cfl_cv`, and `cfl_cw` fields respectively).

## Observation and Model Comparison (OBS)

### Table of contents

13.1. Running the observation operator code example . . . . .	194
13.2. Technical details and full namelist options . . . . .	195
13.3. Example feedback type observation file headers . . . . .	197
13.3.1. Temperature and salinity profile feedback file . . . . .	197
13.3.2. Sea level anomaly feedback file . . . . .	199
13.3.3. Sea surface temperature feedback file . . . . .	201
13.4. Adding code for a new variable . . . . .	202
13.5. Theoretical details . . . . .	203
13.5.1. Horizontal interpolation and averaging methods . . . . .	203
13.5.2. Grid search . . . . .	204
13.5.3. Parallel aspects of horizontal interpolation . . . . .	206
13.5.4. Vertical interpolation operator . . . . .	206
13.6. Standalone observation operator (SAO) . . . . .	209
13.6.1. Concept . . . . .	209
13.6.2. Using the standalone observation operator . . . . .	209
13.6.3. Configuring the standalone observation operator . . . . .	209
13.7. Observation utilities . . . . .	210
13.7.1. Obstools . . . . .	210
13.7.2. Building the obstools . . . . .	212
13.7.3. Dataplot . . . . .	212

### Changes record

Release	Author(s)	Modifications
5.0	<i>D. Ford</i>	<i>Update to accommodate interface changes</i>
4.0	...	...
3.6	...	...
3.4	...	...
<=3.4	...	...



The observation and model comparison code, the observation operator (OBS), reads in observation files (temperature profiles, salinity profiles, velocity profiles, sea surface temperature, sea surface salinity, sea level anomaly, sea surface velocity, sea ice concentration, sea ice thickness) 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. Its use can be extended to other variables, for instance biogeochemistry, with small changes to the code.

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 13.1) 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 fields by setting the namelist value `ln_night` 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 13.1) and multiple instances of the namelist `&namobs_dta` (namelist 13.2), one for each observation group. See the following sections for more details on setting up the namelists.

In section 13.1 a test example of the observation operator code is introduced, including where to obtain data and how to setup the namelists. In section 13.2 some more technical details of the available options are introduced, and we also show more complete namelists. In section 13.3 example “feedback” file NetCDF headers are shown. In section 13.4 a description of how to add an observation operator for a new variable is given. In section 13.5 some of the theoretical aspects of the observation operator are described including interpolation methods and running on multiple processors. In section 13.6 the standalone observation operator code is described. In section 13.7 we describe some utilities to help work with the files produced by the OBS code.

## 13.1. 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](http://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:

Top-level options are defined through the `&namobs` (namelist 13.1) namelist variables.

- `ln_diaobs` turns on the observation operator code.

```

!-----
&namobs      !  observation and model comparison                (default: OFF)
!-----
ln_diaobs    = .false.           ! Logical switch for the observation operator
nn_obsgrups  = 0                 ! Number of observation group namelists (namobs_dta) to read in
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
cn_gridsearchfile = 'grid_search' ! Grid search file name header
rn_gridsearchres = 0.5          ! Grid search resolution
dn_dobsini   = 00010101.000000  ! Initial date in window YYYYMMDD.HHMMSS
dn_dobsend   = 99991231.000000  ! Final date in window YYYYMMDD.HHMMSS
/

```

namelist 13.1.: &namobs

- `nn_obsgrups` specifies the number of observation groups and therefore number of `&namobs_dta` (namelist 13.2) namelists required; in this example it should be 1, as the temperature and salinity data are grouped together in the same input file(s).
- 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_gridsearchfile` 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. `rn_gridsearchres` specifies the resolution (in degrees) of the grid search.
- 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 13.5.3](#)).
- `rn_dobsini` and `rn_dobsend` can be used to restrict the period the observation operator is run for.

A `&namobs_dta` (namelist 13.2) namelist is then needed for each observation group. Typically, an individual variable (*e.g.* sea surface temperature) would be its own group, but if variables are collocated in the input files (*e.g.* temperature and salinity profiles, or meridional and zonal velocities) they would be considered as a single group.

- `cn_groupname` is used to name the output “feedback” file(s).
- `ln_enabled` must be `.true.` for the observation operator to be run on this group, setting it to `.false.` allows it to be turned off without needing to remove the instance of `&namobs_dta` (namelist 13.2) .
- Setting `ln_prof` specifies the observations consist of vertical profiles, as in this example. `ln_surf` indicates surface data. One and only one of `ln_prof` and `ln_surf` must be `.true.`.
- `cn_obsfiles` specifies the input filename or array of filenames.
- `cn_obstypes` specifies the observation variable(s). In this example it should be set to 'POTM','PSAL' for profile temperature and profile salinity. A full list of `cn_obstypes` options is given in [section 13.3](#).

Other `&namobs_dta` (namelist 13.2) options are available to control specific behaviour, as detailed in [section 13.2](#).

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 13.7](#). Utilities to convert other input data formats into the feedback format are also described in [section 13.7](#).

## 13.2. Technical details and full namelist options

Here we show more complete information about the available options in the `&namobs_dta` (namelist 13.2) namelists. Each item in `&namobs` (namelist 13.1) has already been described in [section 13.1](#). The first six entries in `&namobs_dta` (namelist 13.2) , which are the ones most likely to be modified by users, have also been described in [section 13.1](#).

The remaining options in `&namobs_dta` (namelist 13.2) offer more specialist control, with some only relevant to specific observation types.

The following are relevant to all observation types:

- Setting `ln_nea` rejects observations within a grid cell that neighbours land.

```

!-----
&namobs_dta  ! observation and model comparison - external data      (see: namobs)
!-----
cn_groupname      = ''           ! Name of obs group (output file will be cn_groupname/'fb_???.nc')
ln_enabled        = .true.       ! Logical switch for group being processed not ignored
ln_prof          = .false.       ! Logical switch for profile data
ln_surf          = .false.       ! Logical switch for surface data
cn_obsfiles      = ''           ! Observation file names
cn_obstypes      = ''           ! Observation types to read from files
ln_nea           = .false.       ! Logical switch for rejecting observations near land
ln_bound_reject  = .false.       ! Logical switch for rejecting obs near the boundary
ln_ignmis        = .true.       ! Logical switch for ignoring missing files
nn_2dint         = 0            ! Type of horizontal interpolation method
! Relevant if ln_prof = .true.:
nn_1dint         = 0            ! Type of vertical interpolation method
nn_profdavtypes = -1           ! Profile data types representing a daily average
ln_all_at_all    = .false.       ! Logical switch for computing all model variables at all obs points
! Relevant if ln_surf = .true.:
ln_fp_indegs    = .true.       ! Logical: T=> averaging footprint is in degrees, F=> in metres
rn_avglamscl    = 0.           ! E/W diameter of observation footprint (metres/degrees)
rn_avgphiscl    = 0.           ! N/S diameter of observation footprint (metres/degrees)
ln_night        = .false.       ! Logical switch for calculating night-time average for obs
ln_time_mean_bkg = .false.       ! Logical switch for applying time mean of background (e.g. to remove tidal signal)
rn_time_mean_period = 24.8333  ! Meaning period in hours if ln_time_mean_bkg (default is AMM tidal period)
ln_obsbias      = .false.       ! Logical switch for bias correction
cn_obsbiasfiles = ''           ! Bias input file names
cn_type_to_biascorrect = ''     ! Observation type to bias correct
cn_obsbiasfile_varname = ''     ! Bias variable name in input file
! Relevant if 'SLA' in cn_obstypes:
ln_altbias      = .false.       ! Logical switch for altimeter bias correction
cn_altbiasfile  = ''           ! Altimeter bias input file name
nn_msshc        = 0            ! MSSH correction scheme
rn_mdtcorr      = 1.61         ! MDT correction
rn_mdtcutoff    = 65.0         ! MDT cutoff for computed correction
! Relevant if 'POTM', 'PSAL', 'SST', or 'SSS' in cn_obstypes:
ln_output_clim = .false.       ! Logical switch to output climatological temperature/salinity (if ln_tradmp)
! Relevant if 'FBD' in cn_obstypes:
rn_radar_snow_penetr = 1.0     ! Snow depth penetration factor for radar ice freeboard conversion
/

```

## namelist 13.2.: &amp;namobs\_dta

- Setting `ln_bound_reject` rejects observations within a grid cell that neighbours a boundary point.
- Setting `ln_ignmis` merely issues a warning if a file listed in `cn_obsfiles` is missing, otherwise execution of the code ceases with an error.
- `nn_2dint` sets the type of horizontal interpolation method, see [section 13.5](#) for details.

The following are only relevant to profile observations:

- `nn_1dint` sets the type of vertical interpolation method, see [section 13.5](#) for details.
- `nn_profdavtypes` is an array of profile observation types corresponding to entries in the `STATION_TYPE` variable in the feedback file (see [section 13.3](#)) which should be treated as daily averages rather than at a specific time. Set to -1 if no daily averaged types.
- Setting `ln_all_at_all` means that if there is more than one variable in `cn_obstypes`, then a model counterpart is calculated for all variables at any location for which there is an observation of any of the variables.

The following are only relevant to surface observations:

- Setting `ln_fp_indegs` means that the footprint of the observations (see options below) is in units of degrees rather than metres.
- `rn_avglamscl` and `rn_avgphiscl` specify the east/west and north/south footprint of the observations respectively. This is only relevant if `nn_2dint` is set to 5 or 6. See [section 13.5](#) for details.
- Setting `ln_night` calculates a night-time average model counterpart rather than one at the nearest model time step.
- Setting `ln_time_mean_bkg` calculates a model counterpart averaged over `rn_time_mean_period` (in hours) rather than one at the nearest model time step.

- Setting `ln_obsbias` allows a NetCDF file containing an estimate of the observation bias on the model grid to be read in, and this bias to be removed from the observation values. `cn_obsbiasfiles` is a list of observation bias file names. `cn_type_to_biascorrect` is the variable in `cn_obstypes` to bias correct. `cn_obsbiasfile_varname` is the NetCDF variable to read from `cn_obsbiasfiles`.

The following are only relevant to sea level anomaly observations ('SLA' is in `cn_obstypes`), which require use of a mean dynamic topography (MDT) (see [section 13.3](#)) which may need bias correcting:

- Setting `ln_altbias` means that the MDT used is bias corrected using a bias file on the model grid in the NetCDF file `cn_altbiasfile`.
- `nn_msshc` sets the mean sea surface height (MSSH) correction to apply to the MDT. 0 means no correction is applied. 1 means a correction based on the mean difference between the MDT and model SSH is applied. 2 means a correction of `rn_mdtcorr` is applied. Either correction is only applied at latitudes between `rn_mdtcutoff` degrees north and `rn_mdtcutoff` degrees south.

The following is only relevant to temperature and salinity observations ('POTM', 'PSAL', 'SST', or 'SSS' in `cn_obstypes`):

- Setting `ln_output_clim` means that as well as a model counterpart, an equivalent climatological counterpart is output. The climatology used is the same as used for tracer damping (see [section 6.6](#)). Therefore, this option also requires `ln_tradmp` to be set in `&namtra_dmp` ([namelist 6.6](#)).

The following is only relevant to sea ice freeboard observations ('FBD' is in `cn_obstypes`):

- `rn_radar_snow_penetr` sets the snow depth penetration factor for the radar ice freeboard conversion.

### 13.3. Example feedback type observation file headers

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 temperature and salinity profile data, sea level anomaly and sea surface temperature are in the following subsections.

The VARIABLES in the feedback files must match those specified in `cn_obstypes` in `&namobs_dta` ([namelist 13.2](#)). The following variable names can be used:

- POTM: temperature profiles
- PSAL: salinity profiles
- UVEL: zonal velocities (profile or surface)
- VVEL: meridional velocities (profile or surface)
- SST: sea surface temperature
- SLA: sea level anomaly
- SSS: sea surface salinity
- ICECONC: sea ice concentration
- SIT: sea ice thickness
- FBD: sea ice freeboard

Other variables, such as biogeochemical variables from PISCES, are not currently available. However, to define a new variable a user need only add an extra option to the relevant SELECT CASE statement in the routine `dia_obs`, that matches a new observation type with the desired model variable (see [section 13.4](#)).

#### 13.3.1. Temperature and salinity 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" ;
}

```

### 13.3.2. 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" ;

```

### 13.3.3. 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" ;
}

```

## 13.4. Adding code for a new variable

As detailed in [section 13.3](#), the OBS interface is only defined for specific physics variables. However, the internal observation operator code is largely generic, and for observation types which can be directly matched to model variables, is relatively straightforward to implement for additional variables. This could include biogeochemical variables from PISCES or another coupled model.

In the routine `dia_obs` in `diaobs.F90` (`src/OCE/OBS/diaobs.F90`) there are two **SELECT CASE** constructs, one for surface variables and one for profile variables. To implement an observation operator for a new variable, an extra **CASE** need simply be added to the relevant **SELECT CASE** construct. To give a generic example for a profile variable:

```

CASE('OBSNAME')
  zprofvar(:, :, :) = MODELVARIABLE(:, :, : , Kmm)

```

where `MODELVARIABLE` is the model variable to be matched to the observation type, and `OBSNAME` is the name of the observation variable used in the feedback file and in `cn_obstypes` in the `&namobs_dta` ([namelist 13.2](#)) namelist. `OBSNAME` must have a maximum of eight characters.

Then, a corresponding namelist and feedback file(s) just need to be supplied, following the method described in [section 13.1](#).

## 13.5. Theoretical details

### 13.5.1. 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_avglamscl` in degrees or metres (set using `ln_fp_indegs`)
- `nn_2dint=6` : Rectangular footprint averaging with E/W and N/S size specified in the namelist as `rn_avglamscl` and `rn_avgphiscl` in degrees or metres (set using `ln_fp_indegs`)

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  $(\lambda_P, \phi_P)$  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](#).

## 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 13.1](#) and [figure 13.2](#).

### 13.5.2. 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

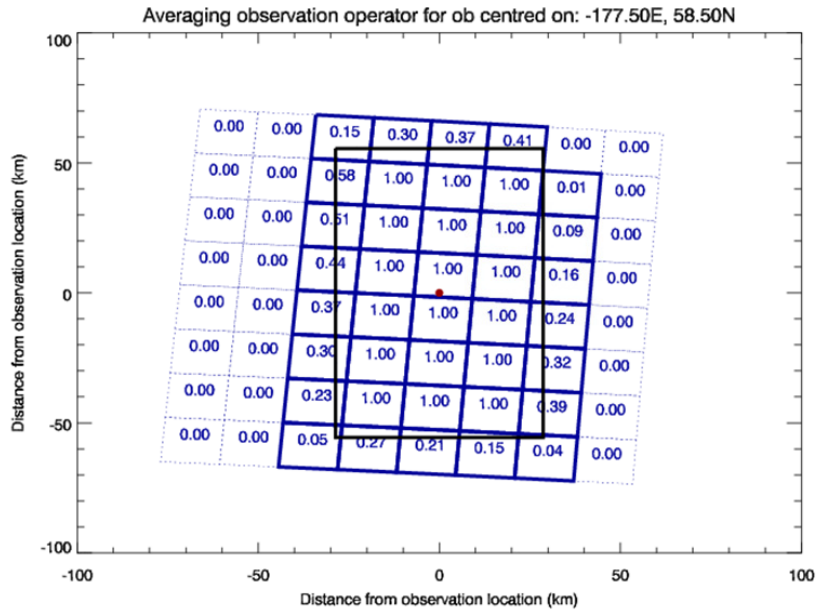


Figure 13.1.: Weights associated with each model grid box (blue lines and numbers) for an observation at  $-170.5^{\circ}\text{E}$ ,  $56.0^{\circ}\text{N}$  with a rectangular footprint of  $1^{\circ} \times 1^{\circ}$ .

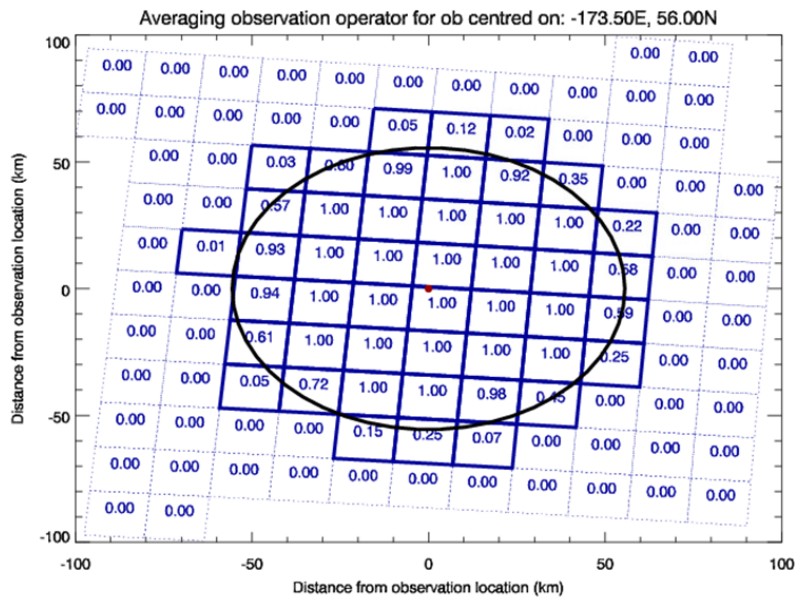


Figure 13.2.: Weights associated with each model grid box (blue lines and numbers) for an observation at  $-170.5^{\circ}\text{E}$ ,  $56.0^{\circ}\text{N}$  with a radial footprint with diameter  $1^{\circ}$ .

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](#).

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.

### 13.5.3. 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 13.3](#) 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  $4 \times 2$  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  $2 \times 2$  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 13.4](#) 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  $4 \times 2$  decomposition with ORCA2 for the same input data as in [figure 13.3](#). 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.

### 13.5.4. 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 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.

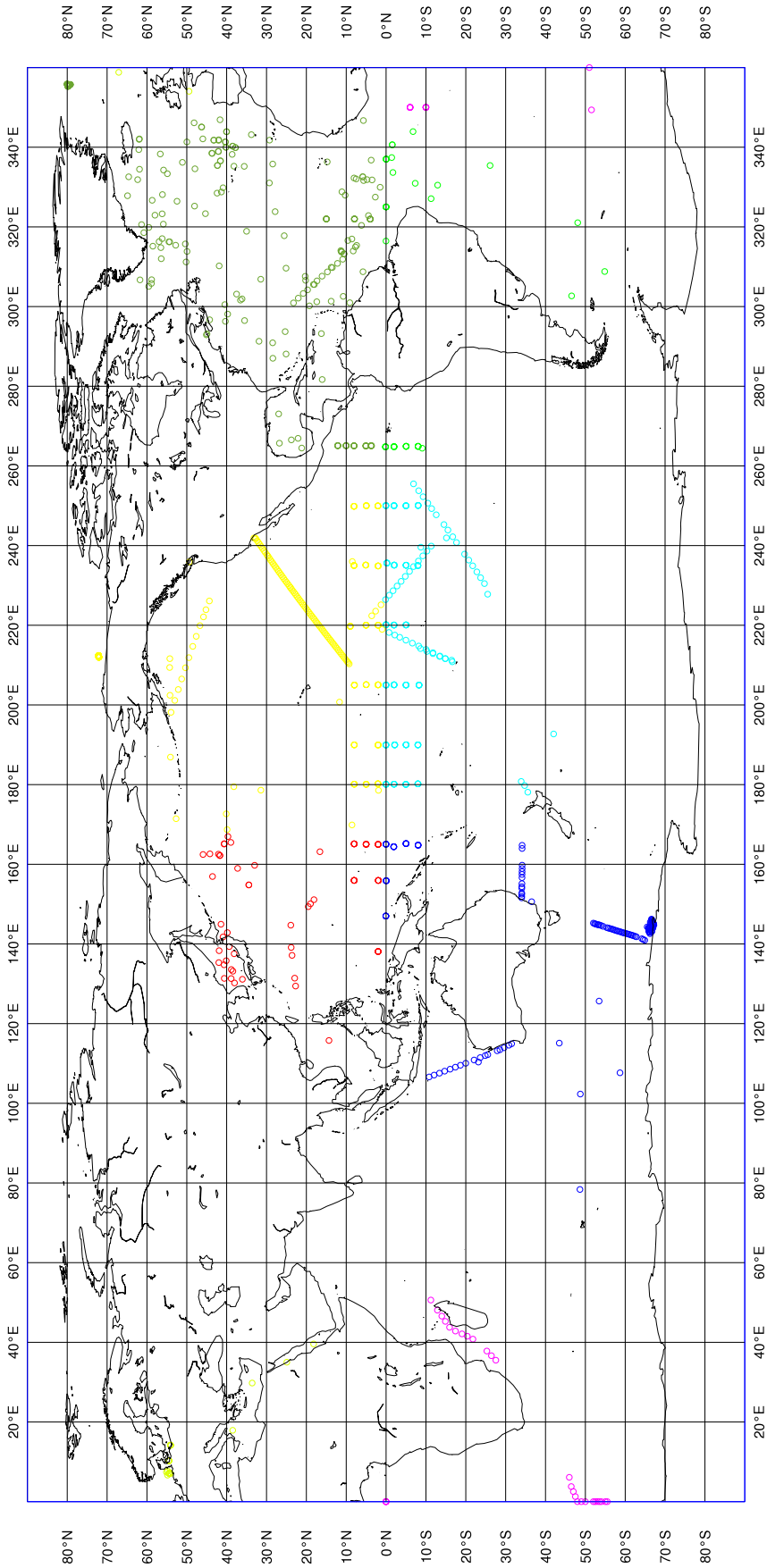


Figure 13.3.: Example of the distribution of observations with the geographical distribution of observational data



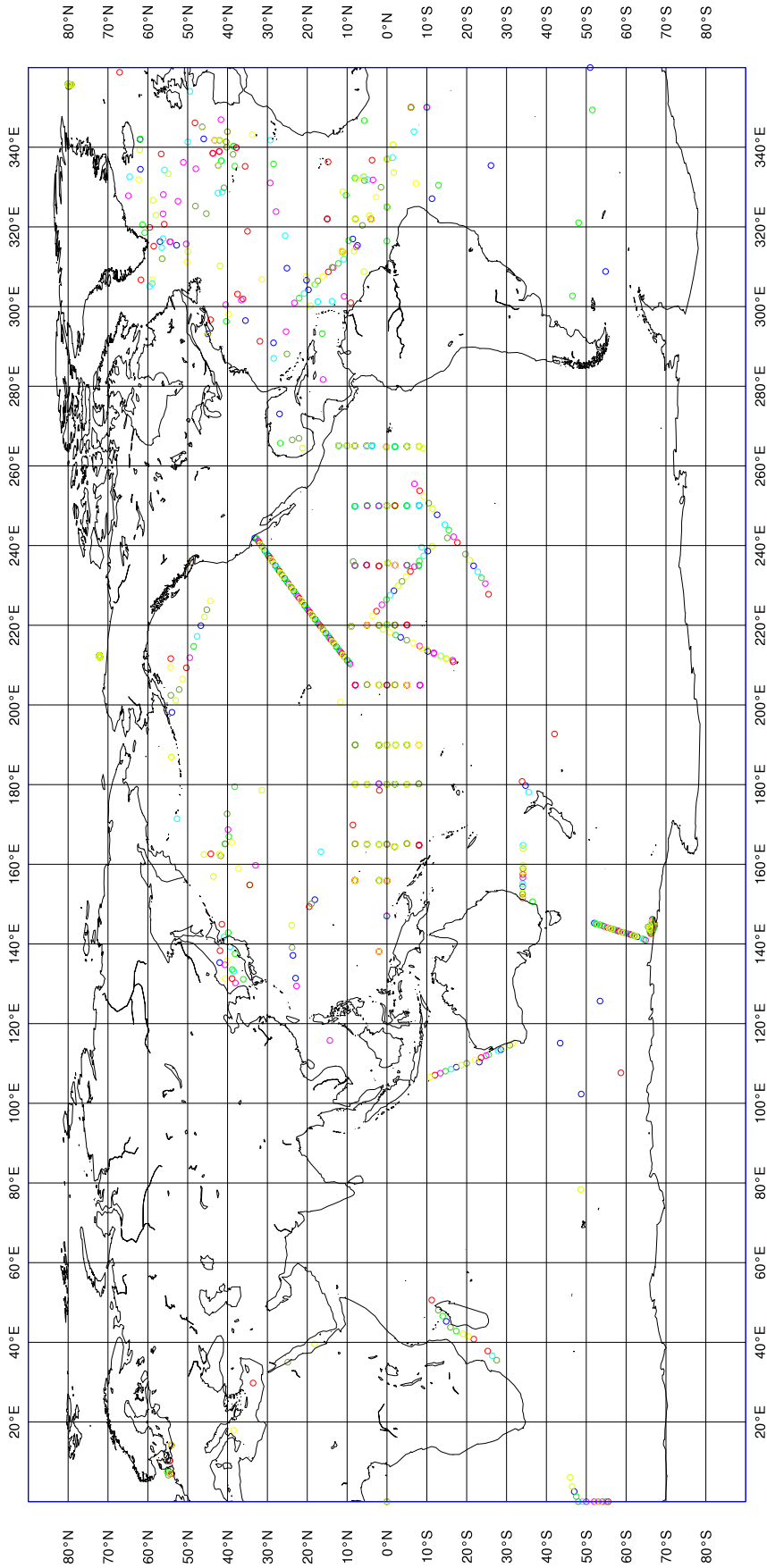


Figure 13.4.: Example of the distribution of observations with the round-robin distribution of observational data.

```

!-----
!      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
/

```

namelist 13.3.: &namsao

## 13.6. Standalone observation operator (SAO)

### 13.6.1. 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.

### 13.6.2. Using the standalone observation operator

#### Building

In addition to *OCE* the SAO requires the inclusion of the *SAO* directory. *SAO* contains a replacement *nemogcm.F90* which overwrites the resultant **nemo.exe**. Note this a similar approach to that taken by the standalone surface scheme *SAS* and the offline TOP model *OFF*.

#### Running

The simplest way to use the executable is to edit and append the **&namsao** ([namelist 13.3](#)) namelist to a full *NEMO* namelist and then to run the executable as if it were *nemo.exe*.

### 13.6.3. Configuring the standalone observation operator

The observation files and settings understood by **&namobs** ([namelist 13.1](#)) have been outlined in the online observation operator section. In addition is a further namelist **&namsao** ([namelist 13.3](#)) 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 13.3](#)) 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.

#### 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.

## 13.7. 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.

### 13.7.1. 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 ...
```

### 13.7.2. Building the obstools

To build the obstools use in the tools directory use `./maketools -n OBSTOOLS -m [ARCH]`.

### 13.7.3. 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:

```
IDL> dataplot, "filename"
```

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.

```
IDL> spawn, 'ls profb*.nc', files
```

```
IDL> dataplot, files
```

[figure 13.5](#) 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 13.6](#)).

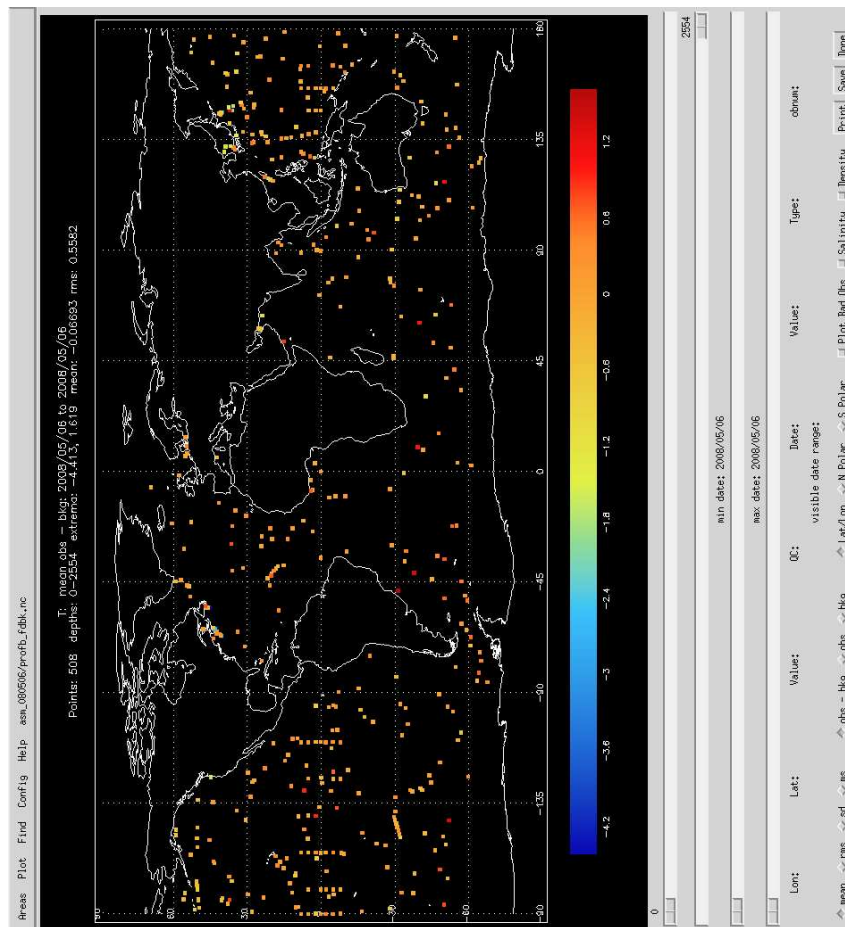


Figure 13.5.: Main window of dataplot

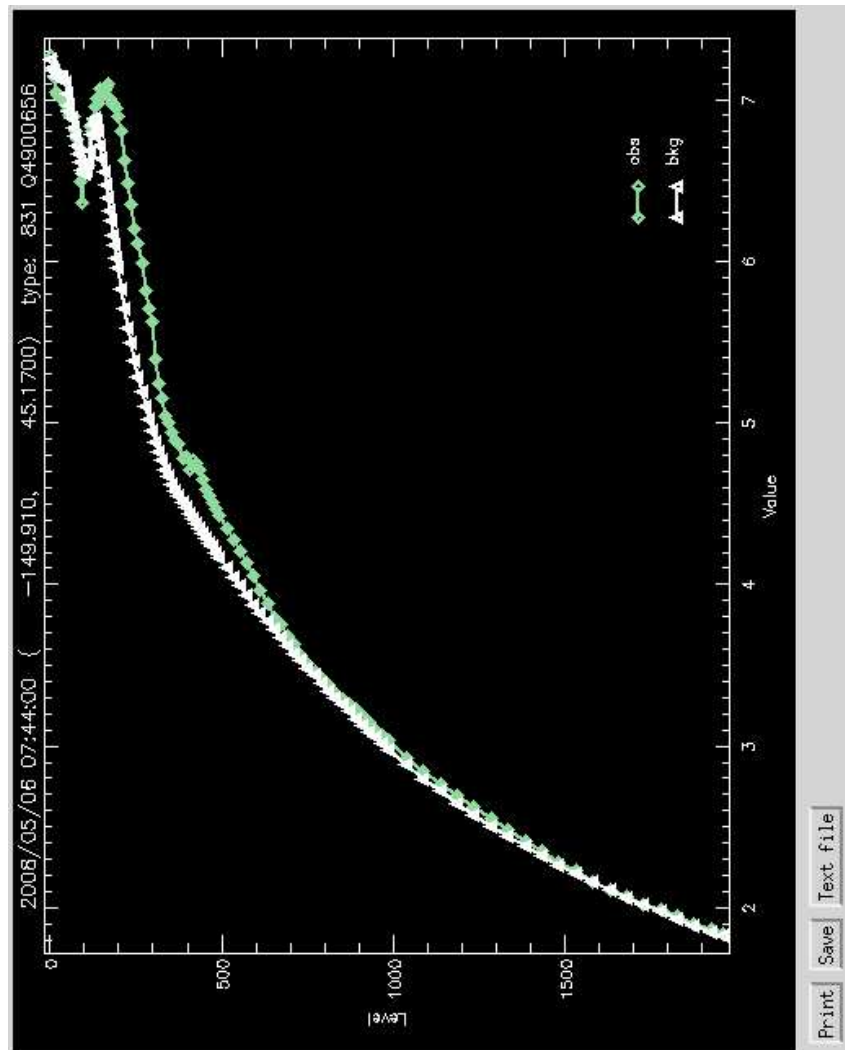


Figure 13.6.: Profile plot from dataplot produced by right clicking on a point in the main window



## Apply Assimilation Increments (ASM)

### Table of contents

14.1. Direct initialization . . . . .	216
14.2. Incremental analysis updates . . . . .	216
14.3. Divergence damping initialisation . . . . .	217
14.4. Implementation details . . . . .	217

### Changes record

Release	Author(s)	Modifications
5.0	<i>M. Bell</i>	<i>Update of the IAU section</i>
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 14.1). There is a brief description of all the namelist options provided. To build the ASM code `key_asminc` must be set.

## 14.1. 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.

## 14.2. 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 OI. IAU is used when `ln_asmiau` is set to true.

With IAU, the model state trajectory  $\mathbf{x}$  in the assimilation window ( $t_0 \leq t_i \leq t_N$ ) is corrected by adding the analysis increments for temperature, salinity, horizontal velocity and SSH as additional tendency terms to the prognostic equations:

$$\mathbf{x}^a(t_i) = M(t_i, t_0)[\mathbf{x}^b(t_0)] + F_i \delta \tilde{\mathbf{x}}^a$$

where  $F_i$  is a weighting function for applying the increments  $\delta \tilde{\mathbf{x}}^a$  defined such that  $\sum_{i=1}^N F_i = 1$ .  $\mathbf{x}^b$  denotes the model initial state and  $\mathbf{x}^a$  is the model state after the increments are applied. To control the adjustment time of the model to the increment, the increment can be applied over an arbitrary sub-window,  $t_m \leq t_i \leq t_n$ , of the main assimilation window, where  $t_0 \leq t_m \leq t_i$  and  $t_i \leq t_n \leq t_N$ . Typically the increments are spread evenly over the full window. In addition, two different weighting functions have been implemented. The first function (namelist option `niaufn = 0`) employs constant weights,

$$F_i^{(1)} = \begin{cases} 0 & \text{if } t_i < t_m \\ 1/M & \text{if } t_m < t_i \leq t_n \\ 0 & \text{if } t_i > t_n \end{cases} \quad (14.1)$$

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:

$$F_i^{(2)} = \begin{cases} 0 & \text{if } t_i < t_m \\ \alpha i & \text{if } t_m \leq t_i \leq t_{M/2} \\ \alpha (M - i + 1) & \text{if } t_{M/2} < t_i \leq t_n \\ 0 & \text{if } t_i > t_n \end{cases} \quad (14.2)$$

where  $\alpha^{-1} = \sum_{i=1}^{M/2} 2i$  and  $M$  is assumed to be even. The weights described by equation 14.2 provide a smoother transition of the analysis trajectory from one assimilation cycle to the next than that described by equation 14.1.

The NEMO IAU code for assimilation of conservative temperature, absolute salinity and horizontal velocity increments is conceptually straightforward. The horizontal velocity increments are applied by SUBROUTINE `dyn_asm_inc` and the conservative temperature and absolute salinity increments by SUBROUTINE `tra_asm_inc`. Those subroutines are called by `stp_rk3_stg3` or by `stp_mlf` depending on whether the RK3 timestep or the MLF timestep scheme is being used.

The assimilation of surface height increments is less conceptually straightforward because of the way that NEMO updates the tracer fields. The NEMO model is designed to conserve total heat and salt content. For each baroclinic timestep, the model calculates the total flux of heat and salt through all the faces of each grid cell in a single baroclinic time-step. The volume of the cell at the end of the time-step is calculated from the volume at the start and the volume fluxes through all of its faces. The heat and salt content of the cell at the start of the timestep is calculated from its conservative temperature,  $T$ , and absolute salinity,  $S$ , values and its volume at that time. The heat and salt content at the end of the timestep are those at the start minus the fluxes of heat and salt out of the cell in that timestep.  $T$  and  $S$  at the end of the timestep are then calculated using the volume at the end of the timestep.

The surface height assimilation increments change the volume of the cells. In order for this change in volume to be distributed proportionately across all the cells in a vertical column, increments are made to the horizontal divergence that is used to calculate the rate of change of the cell thicknesses,  $\partial e_3 / \partial t$ , see [equation A.13](#) in [appendix A](#). These increments are calculated by `ssh_asm_div` which is called from within `div_hor`. In addition `tra_sbc` calculates and applies increments to  $T$  and  $S$  that are proportional to the surface height assimilation increments. Any other tracers should be updated similarly (the code does not do that currently). This is the correct thing to do for balanced ssh increments (the ones which do not propagate away within a baroclinic timestep) but does not appear to be conceptually sound for unbalanced ssh increments. It would be cleaner conceptually to apply the model increments and the assimilation increments as interleaved, independent steps but that scheme has not been implemented in *NEMO* v5.0.

The tests of the IAU code for *NEMO* v5.0 utilised SETTE's ORCA2\_OBS configuration. This has globally uniform  $T$ ,  $S$  and horizontal velocity increment fields. The velocity increments had little impact on the velocities in the RK3 and the MLF.

The assimilation increments are currently applied at all three sub-stages of the RK3 scheme. Tests applying the increments only on the third sub-stage gave similar results but that code is not included in *NEMO* v5.0 because of the limitations of the tests made (see previous paragraph).

The IAU code to assimilate sea-ice concentration data has not been tested at v5.0.

### 14.3. 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:

$$\begin{cases} u_I^n = u_I^{n-1} + \frac{1}{e_{1u}} \delta_{i+1/2} (A_D \chi_I^{n-1}) \\ v_I^n = v_I^{n-1} + \frac{1}{e_{2v}} \delta_{j+1/2} (A_D \chi_I^{n-1}) \end{cases}, \quad (14.3)$$

where the divergence is defined as

$$\chi_I^{n-1} = \frac{1}{e_{1t} e_{2t} e_{3t}} (\delta_i [e_{2u} e_{3u} u_I^{n-1}] + \delta_j [e_{1v} e_{3v} v_I^{n-1}]).$$

By the application of [equation 14.3](#) 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 14.1](#)) 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.

### 14.4. Implementation details

Here we show an example `&nam_asminc` ([namelist 14.1](#)) 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 ;
```

```

-----
&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_sicinc = .false. ! Logical switch for applying sea ice concentration increments
ln_sitinc = .false. ! Logical switch for applying sea ice thickness increments
ln_asmdin = .false. ! Logical switch for Direct Initialization (DI)
ln_asmiau = .false. ! Logical switch for Incremental Analysis Updating (IAU)
nn_itbkg = 0 ! Timestep of background in [0, nitend-nit000-1]
nn_itdin = 0 ! Timestep of background for DI in [0, nitend-nit000-1]
nn_itiaustr = 1 ! Timestep of start of IAU interval in [0, nitend-nit000-1]
nn_itiaufin = 15 ! Timestep of end of IAU interval in [0, nitend-nit000-1]
nn_iaufn = 0 ! Type of IAU weighting function
ln_temnofreeze = .false. ! Don't allow the temperature to drop below freezing
nn_divdmp = 0 ! Number of iterations of divergence damping operator
ln_bv_check = .false. ! Don't apply T/S increments where Brunt-Vaisala (N2) checks fail
rn_bv_thres = 0.0 ! Brunt-Vaisala threshold for applying T/S increments
rn_zmin_bv = 400.0 ! Min depth to verify Brunt-Vaisala (N2) values
rn_zmax_bv = 1500.0 ! Max depth to verify Brunt-Vaisala (N2) values
ln_salfix = .false. ! Logical switch for ensuring that the sa > salfixmin
rn_salfixmin = -9999.0 ! Minimum salinity after applying the increments
rn_zhi_damin = 0.45 ! Ice thickness for new sea ice from DA increment
rn_ai_damin = 0.15 ! Minimum total ice concentration to apply ice thickness increments
rn_acat_damin = 0.01 ! Minimum ice concentration at category level to apply ice thickness increments
/

```

namelist 14.1.: &nam\_asminc

```

double z_inc_datef ;
double bckint(t, z, y, x) ;
double bckins(t, z, y, x) ;
double bckinu(t, z, y, x) ;
double bckinv(t, z, y, x) ;
double bckineta(t, y, x) ;

// global attributes:
:DOMAIN_number_total = 1 ;
:DOMAIN_number = 0 ;
:DOMAIN_dimensions_ids = 1, 2 ;
:DOMAIN_size_global = 182, 149 ;
:DOMAIN_size_local = 182, 149 ;
:DOMAIN_position_first = 1, 1 ;
:DOMAIN_position_last = 182, 149 ;
:DOMAIN_halo_size_start = 0, 0 ;
:DOMAIN_halo_size_end = 0, 0 ;
:DOMAIN_type = "BOX" ;
}

```

## Stochastic Parametrization of EOS (STO)

### Table of contents

15.1. Stochastic processes . . . . .	220
15.2. Implementation details . . . . .	221

### 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:

$$\rho = \frac{1}{2} \sum_{i=1}^m \{ \rho[T + \Delta T_i, S + \Delta S_i, p_o(z)] + \rho[T - \Delta T_i, S - \Delta S_i, p_o(z)] \} \quad (15.1)$$

where  $p_o(z)$  is the reference pressure depending on the depth and,  $\Delta T_i$  and  $\Delta S_i$  ( $i=1,m$ ) is a set of T/S perturbations defined as the scalar product of the respective local T/S gradients with random walks  $\xi$ :

$$\Delta T_i = \xi_i \cdot \nabla T \quad \text{and} \quad \Delta S_i = \xi_i \cdot \nabla S \quad (15.2)$$

$\xi_i$  are produced by a first-order autoregressive process (AR-1) with a parametrized decorrelation time scale, and horizontal and vertical standard deviations  $\sigma_s$ .  $\xi$  are uncorrelated over the horizontal and fully correlated along the vertical.

## 15.1. 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 adopted in the STO module is to generate 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:

$$\xi_{k+1}^{(i)} = a^{(i)} \xi_k^{(i)} + b^{(i)} w^{(i)} + c^{(i)} \quad (15.3)$$

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:

$$\begin{cases} a^{(i)} = \varphi \\ b^{(i)} = \frac{n-1}{2(4n-3)} \sqrt{1 - \varphi^2} \\ c^{(i)} = \mu^{(i)} (1 - \varphi) \end{cases} \quad \text{with} \quad \varphi = \exp(-1/\tau^{(i)}) \quad (15.4)$$

In this way, higher order processes can be easily generated recursively using the same piece of code implementing equation 15.3, and using successive processes from order 0 to  $n-1$  as  $w^{(i)}$ . The parameters in equation 15.4 are computed so that this recursive application of equation 15.3 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 ( $2^\circ$ ) NEMO configuration. 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.

## 15.2. Implementation details

The code implementing stochastic parametrisation is located in the `src/OCE/STO` directory. It contains three modules :

`stpar.F90` : define the Stochastic parameters and their time evolution

`stornng.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 parametrisation associated with the non-linearity of the equation of seawater, implementing [equation 15.2](#) so as specifics in the equation of state implementing [equation 15.1](#).

The `stpar.F90` module includes three public routines called in the model:

( `sto_par` ) is a direct implementation of [equation 15.3](#), 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 parametrisations including equation of state ( `ln_sto_eos` ), lateral diffusion ( `ln_sto_ldf` ), horizontal pressure gradient ( `ln_sto_hpg` ), ice strength ( `ln_sto_pstar` ), trend ( `ln_sto_trd` ), tracers dynamics ( `ln_sto_trc` ).

As for the current release, **only the stochastic parametrisation of equation of state is fully available and tested.**

### Options and parameters

The `ln_sto_eos` namelist variable activates stochastic parametrisation of equation of state. By default it set to `.false.` and not active. The set of parameters is available in `&namsto` ([namelist 15.1](#)) namelist (only the subset for equation of state stochastic parametrisation is listed below):

The variables of stochastic parametrisation itself (based on the global 2° 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_eosflt` : passes of Laplacian filter

`rn_eos_lim` : limitation factor (default = 3.0)

The first four parameters define the stochastic part of equation of state.



```

-----
&namsto      ! Stochastic parametrization of EOS                (default: OFF)
-----
ln_sto_ldf  = .false.  ! stochastic lateral diffusion
rn_ldf_std  = 0.1      ! lateral diffusion standard deviation (in percent)
rn_ldf_tcor = 1440.   ! lateral diffusion correlation timescale (in timesteps)
ln_sto_hpg  = .false.  ! stochastic pressure gradient
rn_hpg_std  = 0.1      ! density gradient standard deviation (in percent)
rn_hpg_tcor = 1440.   ! density gradient correlation timescale (in timesteps)
ln_sto_pstar = .false. ! stochastic ice strength
rn_pstar_std = 0.1     ! ice strength standard deviation (in percent)
rn_pstar_tcor = 1440. ! ice strength correlation timescale (in timesteps)
nn_pstar_ord = 1       ! order of autoregressive processes
nn_pstarflt = 0        ! passes of Laplacian filter
ln_sto_trd  = .false.  ! stochastic model trend
rn_trd_std  = 0.1      ! trend standard deviation (in percent)
rn_trd_tcor = 1440.   ! trend correlation timescale (in timesteps)
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_eosflt   = 0        ! passes of Laplacian filter
rn_eos_lim   = 2.0     ! limitation factor (default = 3.0)
ln_sto_trc  = .false.  ! stochastic tracer dynamics
nn_sto_trc  = 1        ! number of independent random walks
rn_trc_stdxy = 1.4     ! random walk horz. standard deviation (in grid points)
rn_trc_stdz  = 0.7     ! random walk vert. standard deviation (in grid points)
rn_trc_tcor  = 1440.   ! random walk time correlation (in timesteps)
nn_trc_ord  = 1        ! order of autoregressive processes
nn_trcflt   = 0        ! passes of Laplacian filter
rn_trc_lim   = 3.0     ! 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 15.1.: &namsto

## Table of contents

16.1. Representation of unresolved straits . . . . .	224
16.1.1. Hand made geometry changes . . . . .	224
16.2. Closed seas ( <i>closea.F90</i> ) . . . . .	224
16.3. Accuracy and reproducibility ( <i>lib_fortran.F90</i> ) . . . . .	226
16.3.1. Issues with intrinsic SIGN function ( <b>key_nosignedzero</b> ) . . . . .	226
16.3.2. MPP reproducibility . . . . .	227
16.4. Model optimisation, control print and benchmark . . . . .	227
16.4.1. Status and debugging information output . . . . .	227
16.4.2. Control print suboptions . . . . .	228

## Changes record

Release	Author(s)	Modifications
<i>X.X</i>	<i>Pierre Mathiot</i>	Update of the closed sea section
<i>4.0</i>	...	...
<i>3.6</i>	...	...
<i>3.4</i>	...	...
<i>&lt;=3.4</i>	...	...

## 16.1. 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.

### 16.1.1. 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 16.1). 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 16.1 and changes to specific locations can be coded in `usrdef_fmask.F90`. The `usr_def_fmask` routine is always called after `fmask` has been defined according to the choice of lateral boundary condition as discussed in section 9.1. The default version of `usrdef_fmask.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_fmask` makes no changes to `fmask` for any other configuration.

## 16.2. 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.

The inland seas and lakes are defined using mask fields in the domain configuration file. Special treatment of the closed sea (redistribution of net freshwater or mask those), are defined in [namelist 16.1](#) and can be trigger by `ln_closea = .true.` in `namelist namcfg`.

The options available are the following:

`ln_maskcs = .true.` All the closed seas are masked using `mask_opensea` variable.

`ln_maskcs = .false.` 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 a target area.

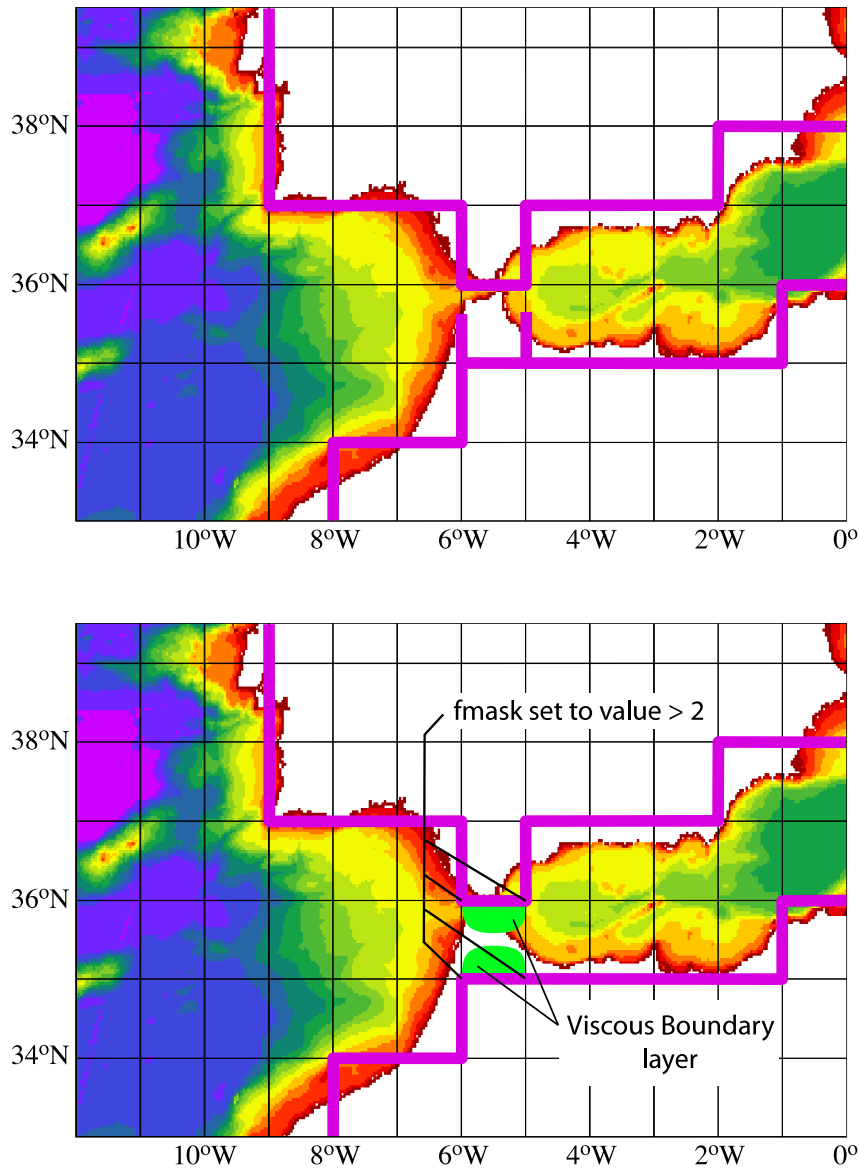


Figure 16.1.: Example of the Gibraltar strait defined in a  $1^\circ \times 1^\circ$  mesh. *Top*: using partially open cells. The meridional scale factor at  $v$ -point is reduced on both sides of the strait to account for the real width of the strait (about 20 km). Note that the scale factors of the strait  $T$ -point remains unchanged. *Bottom*: using viscous boundary layers. The four  $fmask$  parameters along the strait coastlines are set to a value larger than 4, *i.e.* "strong" no-slip case (see figure 9.2) creating a large viscous boundary layer that allows a reduced transport through the strait.

```

!-----
&namclo      ! parameters of the closed sea (cs) behavior          (default: OFF)
!-----
  ln_maskcs = .false.      ! (=T) cs are masked ; So, in this case ln_mask_csundef and ln_clo_rnf have no effect.
  !                       ! (=F => set ln_mask_csundef and ln_clo_rnf)
  !                       ! cs masks are read and net evap/precip over closed sea spread out depending on domain_cfg.nc
↪ masks.
  !                       ! See ln_mask_csundef and ln_clo_rnf for specific option related to this case
  !
  ln_mask_csundef = .true. ! (=T) undefined closed seas are masked ;
  !                       ! (=F) undefined closed seas are kept and no specific treatment is done for these closed seas
  !
  ln_clo_rnf = .true.     ! (=T) river mouth specified in domain_cfg.nc masks (rnf and emp case) are added to the runoff
↪ mask.
  !                       ! allow the treatment of closed sea outflow grid-points to be the same as river mouth
↪ grid-points
/

```

namelist 16.1.: &namclo

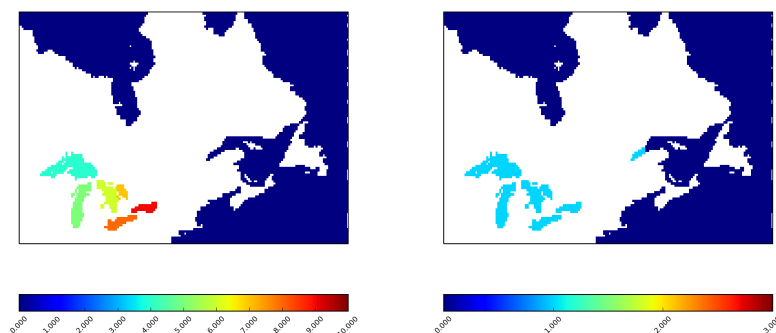


Figure 16.2.: Example of mask fields for the *closea.F90* module. *Left*: a *mask\_csrnf* field; *Right*: a *mask\_csrprnf* field. In this example, if *ln\_closea* is set to *.true.*, the mean freshwater flux over each of the American Great Lakes will be set to zero, and the total residual for all the lakes, if negative, will be put into the St Lawrence Seaway in the area shown.

When *ln\_maskcs* = *.false.*, 3 options are available for the redistribution (set up of these options is done in the tool *DOMAINcfg*):

- **glo** : The residual flux is redistributed globally.
- **emp** : The residual flux is redistributed as emp in a river outflow.
- **rnf** : The residual flux is redistributed as rnf in a river outflow if negative. If there is a net evaporation, the residual flux is redistributed globally.

For each case, 2 masks are needed (figure 16.2):

- one describing the 'sources' (ie the closed seas concerned by each options) called *mask\_csglo*, *mask\_csrnf*, *mask\_csemp*.
- one describing each group of inland seas (the Great Lakes for example) and the target area (river outflow or world ocean) for each group of inland seas (St Lawrence for the Great Lakes for example) called *mask\_csrpglo*, *mask\_csrprnf*, *mask\_csrpemp*.

Closed sea not defined (because too small, issue in the bathymetry definition ...) are defined in *mask\_csundef*. These points can be masked using the namelist option *ln\_mask\_csundef* = *.true.* or used to correct the bathymetry input file.

The masks needed for the closed sea can be created using the *DOMAINcfg* tool in the *utils/tools/DOMAINcfg* directory. See section F.4 for details on the usage of definition of the closed sea masks.

## 16.3. Accuracy and reproducibility ( *lib\_fortran.F90* )

### 16.3.1. 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 it is effective when the macro *key\_nosignedzero*

```

!-----
&namctl          ! Control prints                               (default: OFF)
!-----
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_prctl  = .FALSE.    !
sn_cfctl%l_prtrc  = .FALSE.    !
sn_cfctl%l_oasout = .FALSE.    !
sn_cfctl%l_obsstat = .FALSE.    !
sn_cfctl%l_procmn = 0           ! Minimum area number for reporting [default:0]
sn_cfctl%l_procmx = 1000000    ! Maximum area number for reporting [default:1000000]
sn_cfctl%l_procmi = 1           ! Increment for optional subsetting of areas [default:1]
sn_cfctl%l_ptiminc = 1         ! Timestep increment for writing time step progress info
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 16.2.: &namctl

is defined within the cpp file of the configuration. We use a CPP key as the overwriting of a intrinsic function can present performance issues with some computers/compilers.

### 16.3.2. 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.

## 16.4. Model optimisation, control print and benchmark

Options are defined through the `&namctl` ([namelist 16.2](#)) namelist variables.

### 16.4.1. Status and debugging information output

*NEMO* can produce a range of text information output either: in the main output file (*ocean.output*) written by the normal reporting processor (`narea == 1`) or various specialist output files (e.g. *layout.dat*, *run.stat*, *tracer.stat* etc.). Some, for example *run.stat* and *tracer.stat*, contain globally collected values for which a single file is sufficient. Others, however, contain information that could, potentially, be different for each processing

region. For computational efficiency, the default volume of text information produced is reduced to just a few files from the `narea=1` processor.

When more information is required for monitoring or debugging purposes, the various forms of output can be selected via the `sn_cfctl` structure. As well as simple on-off switches this structure also allows selection of a range of processors for individual reporting (where appropriate) and a time-increment option to restrict globally collected values to specified time-step increments.

Options within the structure are selected by the top-level switches shown here with their default settings:

```
sn_cfctl%l_runstat = .TRUE.      ! 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_prtctl = .FALSE.    !
sn_cfctl%l_prtrc  = .FALSE.    !
sn_cfctl%l_oasout = .FALSE.    !
sn_cfctl%procmin  = 0          ! Minimum area number for reporting [default:0]
sn_cfctl%procmax  = 1000000    ! Maximum area number for reporting [default:1000000]
sn_cfctl%procincr = 1          ! Increment for optional subsetting of areas [default:1]
sn_cfctl%ptmincr  = 1          ! Timestep increment for writing time step progress info
```

Details of the suboptions follow:

## 16.4.2. Control print suboptions

The options that can be individually selected fall into three categories:

### 1. Time step progress information

This category includes `run.stat` and `tracer.stat` files which record certain physical and passive tracer metrics (respectively). Typical contents of `run.stat` include global maximums of ssh, velocity; and global minimums and maximums of temperature and salinity. A netCDF version of `run.stat` (`run.stat.nc`) is also produced with the same time-series data and this can easily be expanded to include extra monitoring information. `tracer.stat` contains the volume-weighted average tracer value for each passive tracer. Collecting these metrics involves global communications and will impact on model efficiency so both these options are disabled by default by setting the respective options, `sn_cfctl%runstat` and `sn_cfctl%trcstat` to false. A compromise can be made by activating either or both of these options and setting the `sn_cfctl%timincr` entry to an integer value greater than one. This increment determines the time-step frequency at which the global metrics are collected and reported. This increment also applies to the time.step file which is otherwise updated every timestep.

### 2. One-time configuration information/progress logs

Some run-time configuration information and limited progress information is always produced by the first ocean process. This includes the `ocean.output` file which reports on all the namelist options read by the model and remains open to catch any warning or error messages generated during execution. A `layout.dat` file is also produced which details the MPI-decomposition used by the model. The suboptions: `sn_cfctl%oeout` and `sn_cfctl%layout` can be used to activate the creation of these files by all ocean processes. For example, when `sn_cfctl%oeout` is true all processors produce their own version of `ocean.output`. All files, beyond the the normal reporting processor (`narea == 1`), are named with a `_XXXX` extension to their name, where `XXXX` is a zero-padded, 4-digit area number (more than 4 digits will be used if the processor count exceeds 9999). This is useful as a debugging aid since all processes can report their local conditions. Note though that these files are buffered on most UNIX systems so bug-hunting efforts using this facility should also utilise the FORTRAN:

```
CALL FLUSH(numout)
```

statement after any additional write statements to ensure that file contents reflect the last model state. Associated with the `sn_cfctl%oeout` option is the additional `sn_cfctl%oasout` suboption. This does not activate its own output file but rather activates the writing of addition information regarding the OASIS configuration when coupling via oasis and the `sbcopl` routine. This information is written to any active `ocean.output` files.

### 3. Control sums of trends for debugging

NEMO includes an option for debugging reproducibility differences between a MPP and mono-processor runs. This is somewhat dated and clearly only useful for this purpose when dealing with configurations that can be run on a single processor. The full details can be found in this report: [The control print](#)



**option in NEMO** The switches to activate production of the control sums of trends for either the physics or passive tracers are the `sn_cfctl%prtctl` and `sn_cfctl%prttrc` suboptions, respectively. Although, perhaps, of limited use for its original intention, the ability to produce these control sums of trends in specific areas provides another tool for diagnosing model behaviour.

If only the output from a select few regions is required then additional options are available to activate options for only a simple subset of processing regions. These are: `sn_cfctl%procmin`, `sn_cfctl%procmax` and `sn_cfctl%procincr` which can be used to specify the minimum and maximum active areas and the increment. The default values are set such that all regions will be active. Note this subsetting can also be used to limit which additional `ocean.output` and `layout.dat` files are produced if those suboptions are active.

## Table of contents

17.1. Introduction . . . . .	231
17.2. List of NEMO Idealised Configurations and Test Cases . . . . .	231
17.3. List of NEMO Realistic Regional Configurations . . . . .	232
17.4. Global configurations: the ORCA family . . . . .	232
17.4.1. ORCA tripolar grid . . . . .	232
17.4.2. ORCA pre-defined resolution . . . . .	234
17.5. A special mention of the double gyre basin: GYRE family . . . . .	234
17.6. Other well documented configurations . . . . .	235

## Changes record

Release	Author(s)	Modifications
5.0	<i>Katherine Hutchinson with input from Sébastien Masson and Gervan Madec</i>	<i>Review completed</i>
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 => fill namclo)
!                      ! (=F) no control of net precip/evap over closed sea
!
ln_write_cfg = .false.  ! (=T) create the domain configuration file
cn_domcfg_out = "domain_cfg_out" ! newly created domain configuration filename
/

```

namelist 17.1.: &namcfg

## 17.1. 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 expectations. 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 17.1) namelist variables.

## 17.2. List of NEMO Idealised Configurations and Test Cases

- GYRE PISCES
- VORTEX
- LOCK EXCHANGE
- OVERFLOW
- WAD - wetting and drying
- DOME - Dynamics of Overflow Mixing and Entrainment
- ISOMIP
- ICE AGRIF
- ICE ADV2D
- ICE ADV1D
- ICE RHEO
- BENCH
- CPL OASIS
- DIA GPU
- TSUNAMI
- DONUT
- C1D ASICS
- STATION ASF
- SWG
- ADIAB WAVE

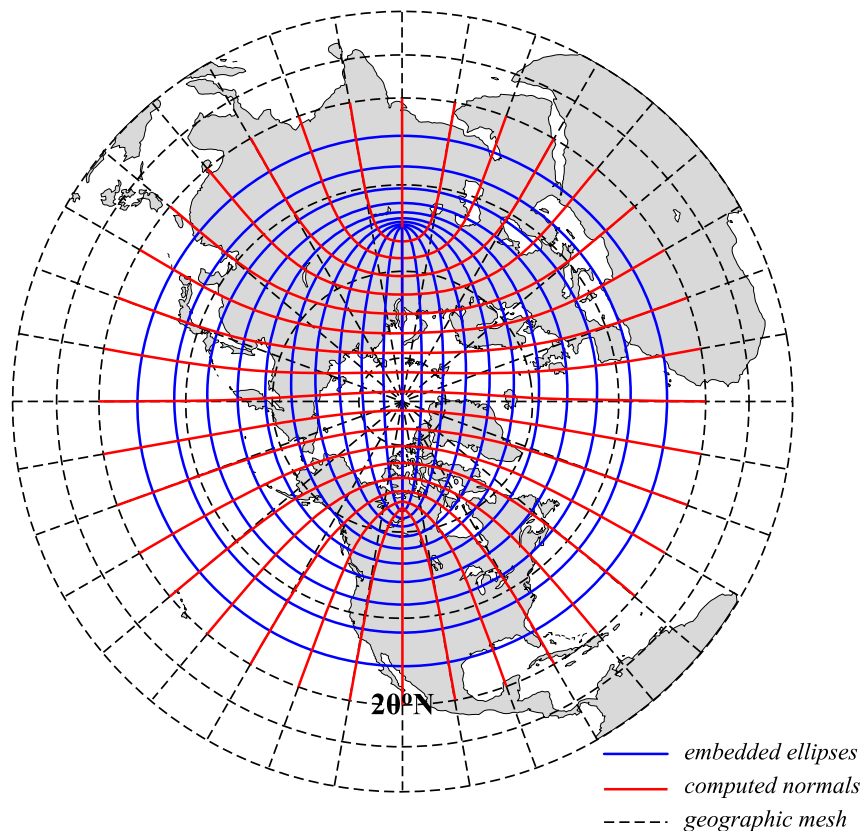


Figure 17.1.: ORCA mesh conception. The departure from an isotropic Mercator grid start poleward of  $20^{\circ}\text{N}$ . The two "north pole" are the foci of a series of embedded ellipses (blue curves) which are determined analytically and form the i-lines of the ORCA mesh (pseudo latitudes). Then, 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).

### 17.3. List of NEMO Realistic Regional Configurations

- [AMM12](#)
- [AGRIF DEMO](#)
- [WED025](#) Worked example available [here](#)

### 17.4. Global configurations: the ORCA family

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.3767939>

In this `namelist_cfg` the name of domain input file is set in `&namcfg` ([namelist 17.1](#)) block of namelist.

#### 17.4.1. 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 17.1](#)). 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  $20^{\circ}\text{N}$ , so that the Equator is a mesh line, which provides a better numerical solution for equatorial dynamics. The choice

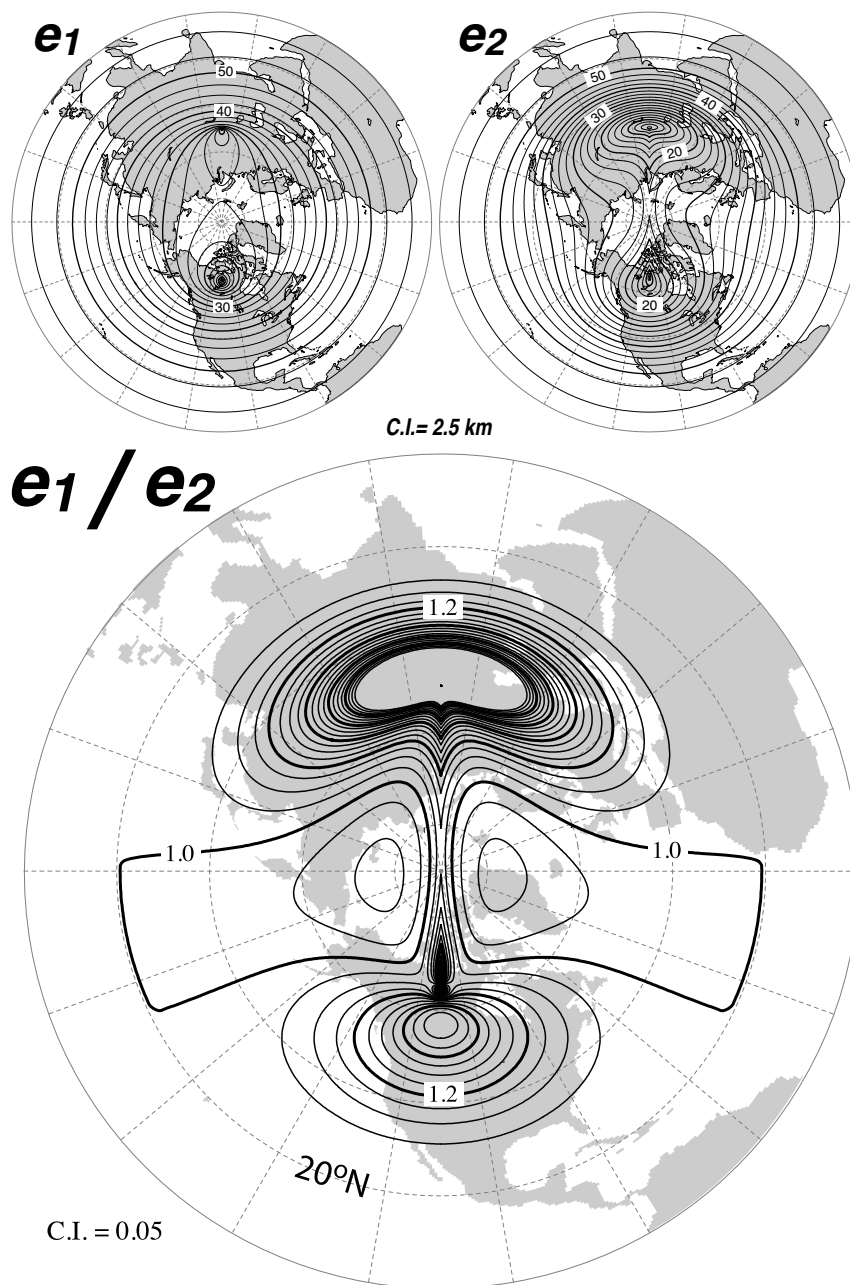


Figure 17.2.: *Top*: Horizontal scale factors ( $e_1$ ,  $e_2$ ) and *Bottom*: ratio of anisotropy ( $e_1/e_2$ ) for ORCA 0.5° mesh. South of 20°N a Mercator grid is used ( $e_1 = e_2$ ) so that the anisotropy ratio is 1. Poleward of 20°N, the two "north pole" introduce a weak anisotropy over the ocean areas ( $< 1.2$ ) except in vicinity of Victoria Island (Canadian Arctic Archipelago).

Horizontal Grid	ORCA_index	jpiglo	jpglo
2 °	2	182	149
1 °	1	362	292
0.5 °	05	722	511
0.25°	025	1442	1021

Table 17.1.: Domain size of ORCA family configurations. The flag for configurations of ORCA family need to be set in *domain\_cfg* file.

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 17.1](#) and [figure 17.2](#) 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.

### 17.4.2. 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 17.1](#)).

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.

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 17.1](#)).

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 17.1](#) 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 7.4.2](#)), 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 multiple online nested zooms, *i.e.* using AGRIF with `key_agrif` defined. This is available in the AGRIF\_DEMO configuration (located in `./cfigs/AGRIF_DEMO/` directory) that accounts for two nested refinements over the Arctic region and a third zoom over the central Pacific area using a two-ways coupling procedure.

## 17.5. A special mention of the double gyre basin: GYRE family

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  $\beta$ -plane centred at  $\sim 30^\circ\text{N}$  and rotated by  $45^\circ$ , 3180 km long, 2120 km wide and 4 km deep ([figure 16.1](#)). 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  $22^\circ\text{N}$  and  $36^\circ\text{N}$ . 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

```

!-----
&namusr_def  !  GYRE user defined namelist
!-----
nn_GYRE      = 1      !  GYRE resolution [1/degrees]
ln_bench     = .false. !  !=T benchmark with gyre: the gridsize is kept constant
jpkglo       = 31     !  number of model levels
/
    
```

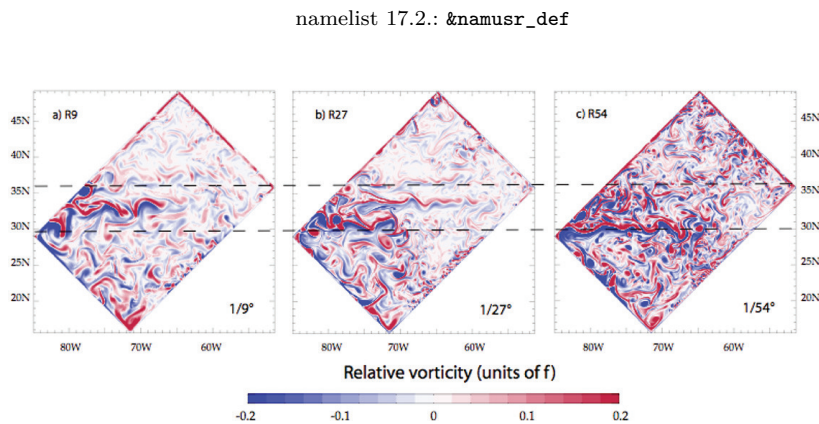


Figure 17.3.: Snapshot of relative vorticity at the surface of the model domain in GYRE R9, R27 and R54. From Lévy et al. (2010).

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 by setting `ln_read_cfg=.false.` in `&namcfg` (namelist 17.1) part of the reference configuration namelist `./cfigs/GYRE_PISCES/EXPREF/namelist_cfg`. The analytical definition of grid in GYRE is done in `usrdef_hrg.F90` and `usrdef_zgr.F90` routines. Its horizontal resolution (and thus the size of the domain) is determined by setting `nn_GYRE` in `&namusr_def` (namelist 17.2) :

$$\begin{aligned}
 jpiglo &= 30 \times nn\_GYRE + 2 + 2 \times nn\_hls \\
 jjpglo &= 20 \times nn\_GYRE + 2 + 2 \times nn\_hls
 \end{aligned}$$

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` (namelist 17.2) in namelist `./cfigs/GYRE_PISCES/EXPREF/namelist_cfg`.

## 17.6. Other well documented configurations

Please note that the following configurations are not distributed, maintained or documented by *NEMO* but we provide simply a list here of possibly useful resources.

- [NEMO-PISCES 1D](#)
- [SEAsia](#)
- [SEVERN-SWOT](#)
- [SRIL34](#)





# Curvilinear $s$ -Coordinate Equations

## Table of contents

- A.1. Chain rule for  $s$ -coordinates . . . . . 237
- A.2. Continuity equation in  $s$ -coordinates . . . . . 237
- A.3. Momentum equation in  $s$ -coordinate . . . . . 238
- A.4. Tracer equation . . . . . 241

## Changes record

Release	Author(s)	Modifications
4.0	...	...
3.6	...	...
3.4	...	...
$\leq 3.4$	...	...

## 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  $\delta i$  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  $\sigma_1$ ,  $\sigma_2$  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  $\omega$ , 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 :

$$\begin{aligned} \frac{Du}{Dt} \Big|_s &= \frac{\partial u}{\partial t} \Big|_s - \zeta v + \frac{1}{2e_1} \frac{\partial(u^2+v^2)}{\partial i} + \frac{1}{e_3} \omega \frac{\partial u}{\partial s} \\ &= \frac{\partial u}{\partial t} \Big|_s + \frac{1}{e_1 e_2} \left( \frac{\partial(e_2 u v)}{\partial i} + \frac{\partial(e_1 u v)}{\partial j} \right) + \frac{1}{e_3} \frac{\partial(\omega u)}{\partial s} \\ &\quad - u \left[ \frac{1}{e_1 e_2} \left( \frac{\partial(e_2 u)}{\partial i} + \frac{\partial(e_1 v)}{\partial j} \right) + \frac{1}{e_3} \frac{\partial \omega}{\partial s} \right] \\ &\quad - \frac{v}{e_1 e_2} \left( v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right). \end{aligned}$$

Introducing the vertical scale factor inside the horizontal derivative of the first two terms (*i.e.* the horizontal

divergence), it becomes :

$$\begin{aligned}
&= \left. \frac{\partial u}{\partial t} \right|_s + \frac{1}{e_1 e_2 e_3} \left( \frac{\partial(e_2 e_3 u^2)}{\partial i} + \frac{\partial(e_1 e_3 u v)}{\partial j} - e_2 u v \frac{\partial e_3}{\partial i} - e_1 u v \frac{\partial e_3}{\partial j} \right) + \frac{1}{e_3} \frac{\partial(\omega u)}{\partial s} \\
&\quad - u \left[ \frac{1}{e_1 e_2 e_3} \left( \frac{\partial(e_2 e_3 u)}{\partial i} + \frac{\partial(e_1 e_3 v)}{\partial j} - e_2 u \frac{\partial e_3}{\partial i} - e_1 v \frac{\partial e_3}{\partial j} \right) + \frac{1}{e_3} \frac{\partial \omega}{\partial s} \right] \\
&\quad - \frac{v}{e_1 e_2} \left( v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \\
&= \left. \frac{\partial u}{\partial t} \right|_s + \frac{1}{e_1 e_2 e_3} \left( \frac{\partial(e_2 e_3 u u)}{\partial i} + \frac{\partial(e_1 e_3 u v)}{\partial j} \right) + \frac{1}{e_3} \frac{\partial(\omega u)}{\partial s} \\
&\quad - u \left[ \frac{1}{e_1 e_2 e_3} \left( \frac{\partial(e_2 e_3 u)}{\partial i} + \frac{\partial(e_1 e_3 v)}{\partial j} \right) + \frac{1}{e_3} \frac{\partial \omega}{\partial s} \right] - \frac{v}{e_1 e_2} \left( v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right).
\end{aligned}$$

Introducing a more compact form for the divergence of the momentum fluxes, and using (equation A.13), the  $s$ -coordinate continuity equation, it becomes :

$$= \left. \frac{\partial u}{\partial t} \right|_s + \nabla \cdot (\mathbf{U} u)|_s + u \frac{1}{e_3} \frac{\partial e_3}{\partial t} - \frac{v}{e_1 e_2} \left( v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right)$$

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:

$$\left. \frac{Du}{Dt} \right|_s = \frac{1}{e_3} \left. \frac{\partial(e_3 u)}{\partial t} \right|_s + \nabla \cdot (\mathbf{U} u)|_s - \frac{v}{e_1 e_2} \left( v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right). \quad (\text{A.16})$$

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  $\sigma_1$  and  $\sigma_2$  by their expression equation A.1, it becomes:

$$\begin{aligned}
-\frac{1}{\rho_o e_1} \left. \frac{\partial p}{\partial i} \right|_z &= -\frac{1}{\rho_o e_1} \left( \left. \frac{\partial p}{\partial i} \right|_s + g \rho \left. \frac{\partial z}{\partial i} \right|_s \right) \\
-\frac{1}{\rho_o e_2} \left. \frac{\partial p}{\partial j} \right|_z &= -\frac{1}{\rho_o e_2} \left( \left. \frac{\partial p}{\partial j} \right|_s + g \rho \left. \frac{\partial z}{\partial j} \right|_s \right).
\end{aligned} \quad (\text{A.17})$$

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 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  $\eta$  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:

$$\begin{aligned} \frac{\partial u}{\partial t} = +(\zeta + f) v - \frac{1}{2 e_1} \frac{\partial}{\partial i} (u^2 + v^2) - \frac{1}{e_3} \omega \frac{\partial u}{\partial k} \\ - \frac{1}{e_1} \left( \frac{\partial p'_h}{\partial i} + g d \frac{\partial z}{\partial i} \right) - \frac{g}{e_1} \frac{\partial \eta}{\partial i} + D_u^U + F_u^U, \end{aligned} \quad (\text{A.20a})$$

$$\begin{aligned} \frac{\partial v}{\partial t} = -(\zeta + f) u - \frac{1}{2 e_2} \frac{\partial}{\partial j} (u^2 + v^2) - \frac{1}{e_3} \omega \frac{\partial v}{\partial k} \\ - \frac{1}{e_2} \left( \frac{\partial p'_h}{\partial j} + g d \frac{\partial z}{\partial j} \right) - \frac{g}{e_2} \frac{\partial \eta}{\partial j} + D_v^U + F_v^U. \end{aligned} \quad (\text{A.20b})$$

whereas the flux form momentum equation differs from it by the formulation of both the time derivative and the pressure gradient term:

$$\begin{aligned} \frac{1}{e_3} \frac{\partial (e_3 u)}{\partial t} = -\nabla \cdot (\mathbf{U} u) + \left\{ f + \frac{1}{e_1 e_2} \left( v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right\} v \\ - \frac{1}{e_1} \left( \frac{\partial p'_h}{\partial i} + g d \frac{\partial z}{\partial i} \right) - \frac{g}{e_1} \frac{\partial \eta}{\partial i} + D_u^U + F_u^U, \end{aligned} \quad (\text{A.21a})$$

$$\begin{aligned} \frac{1}{e_3} \frac{\partial (e_3 v)}{\partial t} = -\nabla \cdot (\mathbf{U} v) - \left\{ f + \frac{1}{e_1 e_2} \left( v \frac{\partial e_2}{\partial i} - u \frac{\partial e_1}{\partial j} \right) \right\} u \\ - \frac{1}{e_2} \left( \frac{\partial p'_h}{\partial j} + g d \frac{\partial z}{\partial j} \right) - \frac{g}{e_2} \frac{\partial \eta}{\partial j} + D_v^U + F_v^U. \end{aligned} \quad (\text{A.21b})$$

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,  $\omega$  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 2 <sup>nd</sup> order tracer diffusive operators . . . . .	244
B.2. Iso/Diapycnal 2 <sup>nd</sup> order tracer diffusive operators . . . . .	245
B.3. Lateral/Vertical momentum diffusive operators . . . . .	247

### 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:

$$D^T = \frac{1}{e_1 e_2} \left[ \frac{\partial}{\partial i} \left( \frac{e_2}{e_1} A^{iT} \frac{\partial T}{\partial i} \Big|_z \right) \Big|_z + \frac{\partial}{\partial j} \left( \frac{e_1}{e_2} A^{iT} \frac{\partial T}{\partial j} \Big|_z \right) \Big|_z \right] + \frac{\partial}{\partial z} \left( A^{vT} \frac{\partial T}{\partial z} \right) \quad (\text{B.1})$$

### In generalized vertical coordinates

In  $s$ -coordinates, we defined the slopes of  $s$ -surfaces,  $\sigma_1$  and  $\sigma_2$  by [equation A.1](#) and the vertical/horizontal ratio of diffusion coefficient by  $\epsilon = A^{vT}/A^{iT}$ . The diffusion operator is given by:

$$D^T = \nabla|_s \cdot [A^{iT} \mathfrak{R} \cdot \nabla|_s T] \quad \text{where } \mathfrak{R} = \begin{pmatrix} 1 & 0 & -\sigma_1 \\ 0 & 1 & -\sigma_2 \\ -\sigma_1 & -\sigma_2 & \epsilon + \sigma_1^2 + \sigma_2^2 \end{pmatrix} \quad (\text{B.2})$$

or in expanded form:

$$D^T = \frac{1}{e_1 e_2 e_3} \left\{ \begin{aligned} & \frac{\partial}{\partial i} \left[ e_2 e_3 A^{iT} \left( \frac{1}{e_1} \frac{\partial T}{\partial i} \Big|_s - \frac{\sigma_1}{e_3} \frac{\partial T}{\partial s} \right) \right] \Big|_s \\ & + \frac{\partial}{\partial j} \left[ e_1 e_3 A^{iT} \left( \frac{1}{e_2} \frac{\partial T}{\partial j} \Big|_s - \frac{\sigma_2}{e_3} \frac{\partial T}{\partial s} \right) \right] \Big|_s \\ & + e_1 e_2 \frac{\partial}{\partial s} \left[ A^{iT} \left( -\frac{\sigma_1}{e_1} \frac{\partial T}{\partial i} \Big|_s - \frac{\sigma_2}{e_2} \frac{\partial T}{\partial j} \Big|_s + (\epsilon + \sigma_1^2 + \sigma_2^2) \frac{1}{e_3} \frac{\partial T}{\partial s} \right) \right] \Big|_s \end{aligned} \right\}.$$

[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:

$$\frac{1}{e_1 e_2 e_3} \left( \frac{\partial (e_2 e_3 \bullet)}{\partial i} \Big|_s \right) \cdot \left[ A^{lT} \begin{pmatrix} 1 & -\sigma_1 \\ -\sigma_1 & \varepsilon + \sigma_1^2 \end{pmatrix} \cdot \left( \frac{1}{e_1} \frac{\partial \bullet}{\partial i} \Big|_s \right) \right] (T)$$

## B.2. Iso/Diapycnal 2<sup>nd</sup> 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,  $\epsilon$  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  $i, j, \rho$ -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  $i, j, \rho$ -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  $i, j, \rho$ -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:

$$\mathbf{D}^{\mathbf{U}} = \nabla_h (A^{lm} \chi) - \nabla_h \times (A^{lm} \zeta \mathbf{k}) + \frac{1}{e_3} \frac{\partial}{\partial k} \left( \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right), \quad (\text{B.8})$$

that is, in expanded form:

$$\begin{aligned} D_u^{\mathbf{U}} &= \frac{1}{e_1} \frac{\partial (A^{lm} \chi)}{\partial i} - \frac{1}{e_2} \frac{\partial (A^{lm} \zeta)}{\partial j} + \frac{1}{e_3} \frac{\partial}{\partial k} \left( \frac{A^{vm}}{e_3} \frac{\partial u}{\partial k} \right), \\ D_v^{\mathbf{U}} &= \frac{1}{e_2} \frac{\partial (A^{lm} \chi)}{\partial j} + \frac{1}{e_1} \frac{\partial (A^{lm} \zeta)}{\partial i} + \frac{1}{e_3} \frac{\partial}{\partial k} \left( \frac{A^{vm}}{e_3} \frac{\partial v}{\partial k} \right). \end{aligned}$$

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 . . . . . 249
- C.2. Continuous conservation . . . . . 250
- C.3. Discrete total energy conservation: vector invariant form . . . . . 252
  - C.3.1. Total energy conservation . . . . . 252
  - C.3.2. Vorticity term (coriolis + vorticity part of the advection) . . . . . 252
  - C.3.3. Pressure gradient term . . . . . 255
- C.4. Discrete total energy conservation: flux form . . . . . 256
  - C.4.1. Total energy conservation . . . . . 256
  - C.4.2. Coriolis and advection terms: flux form . . . . . 256
- C.5. Discrete enstrophy conservation . . . . . 257
- C.6. Conservation properties on tracers . . . . . 259
  - C.6.1. Advection term . . . . . 259
- C.7. Conservation properties on lateral momentum physics . . . . . 259
  - C.7.1. Conservation of potential vorticity . . . . . 260
  - C.7.2. Dissipation of horizontal kinetic energy . . . . . 260
  - C.7.3. Dissipation of enstrophy . . . . . 261
  - C.7.4. Conservation of horizontal divergence . . . . . 261
  - C.7.5. Dissipation of horizontal divergence variance . . . . . 261
- C.8. Conservation properties on vertical momentum physics . . . . . 261
- C.9. Conservation properties on tracer physics . . . . . 263
  - C.9.1. Conservation of tracers . . . . . 263
  - C.9.2. Dissipation of tracer variance . . . . . 264

## 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 constraint in the continuous world. The total energy (*i.e.* kinetic plus potential energies) is conserved:

$$\partial_t \left( \int_D \left( \frac{1}{2} \mathbf{U}_h^2 + \rho g z \right) dv \right) = 0 \quad (\text{C.3})$$

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):

$$\begin{aligned} \int_D \frac{1}{e_1 e_2} (v \partial_i e_2 - u \partial_j e_1) (\mathbf{k} \times \mathbf{U}_h) \cdot \mathbf{U}_h dv &= 0 \\ \int_D \mathbf{U}_h \cdot \begin{pmatrix} \nabla \cdot (\mathbf{U} u) \\ \nabla \cdot (\mathbf{U} v) \end{pmatrix} dv + \frac{1}{2} \int_D \mathbf{U}_h^2 \frac{1}{e_3} \partial_t e_3 dv &= 0 \end{aligned}$$

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:

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

Vector invariant form:

$$\begin{aligned} \int_D \mathbf{U}_h \cdot \text{VOR} dv &= 0 \\ \int_D \mathbf{U}_h \cdot \text{KEG} dv + \int_D \mathbf{U}_h \cdot \text{ZAD} dv - \int_D \frac{\mathbf{U}_h^2}{2} \frac{1}{e_3} \partial_t e_3 dv &= 0 \\ - \int_D \mathbf{U}_h \cdot (\text{HPG} + \text{SPG}) dv = - \int_D \nabla \cdot (\rho \mathbf{U}) g z dv + \int_D g \rho \partial_t z dv \end{aligned}$$

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:

$$\begin{aligned} & \sum_{i,j,k} \left\{ u \partial_t u b_u + v \partial_t v b_v \right\} + \frac{1}{2} \sum_{i,j,k} \left\{ \frac{u^2}{e_{3u}} \partial_t e_{3u} b_u + \frac{v^2}{e_{3v}} \partial_t e_{3v} b_v \right\} \\ & = - \sum_{i,j,k} \left\{ \frac{1}{e_{3t}} \partial_t (e_{3t} \rho) g z_t b_t \right\} - \sum_{i,j,k} \left\{ \rho g \partial_t (z_t) b_t \right\} \end{aligned} \quad (\text{C.5})$$

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:

$$-e_3 q \mathbf{k} \times \mathbf{U}_h \equiv \begin{pmatrix} +\frac{1}{e_{1u}} \frac{\overline{q (e_{1v} e_{3v} v)^{i+1/2}}^j}{\overline{(e_{2u} e_{3u} u)^{j+1/2}}^i} \\ -\frac{1}{e_{2v}} \frac{\overline{q (e_{1v} e_{3v} v)^{i+1/2}}^j}{\overline{(e_{2u} e_{3u} u)^{j+1/2}}^i} \end{pmatrix}$$

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:

$$\begin{aligned} & \int_D - (e_3 q \mathbf{k} \times \mathbf{U}_h) \cdot \mathbf{U}_h dv \\ & \equiv \sum_{i,j,k} \left\{ \frac{1}{e_{1u}} \overline{q V^{i+1/2}}^j u b_u - \frac{1}{e_{2v}} \overline{q U^{j+1/2}}^i v b_v \right\} \\ & \equiv \sum_{i,j,k} \left\{ q \overline{V^{i+1/2}}^j U - q \overline{U^{j+1/2}}^i V \right\} \\ & \equiv \sum_{i,j,k} q \left\{ \overline{V^{i+1/2}} \overline{U^{j+1/2}} - \overline{U^{j+1/2}} \overline{V^{i+1/2}} \right\} \equiv 0 \end{aligned}$$

In other words, the domain averaged kinetic energy does not change due to the vorticity term.

##### Vorticity term with EEN scheme ( `ln_dynvor_een=.true.` )

With the EEN scheme, the vorticity terms are represented as:

$$\begin{cases} +q e_3 v \equiv +\frac{1}{e_{1u}} \sum_{i_p, k_p}^j \overline{q_{j_p}^{i_p}} (e_{1v} e_{3v} v)_{j+j_p}^{i+i_p-1/2} \\ -q e_3 u \equiv -\frac{1}{e_{2v}} \sum_{i_p, k_p}^i \overline{q_{j_p}^{i_p}} (e_{2u} e_{3u} u)_{j+j_p-1/2}^{i+i_p} \end{cases} \quad (\text{C.6})$$

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:

$${}^j_i \mathbb{Q}_{j_p}^{i_p} = \frac{1}{12} \left( q_{j+j_p}^{i-i_p} + q_{j+i_p}^{i+j_p} + q_{j-j_p}^{i+i_p} \right) \quad (\text{C.7})$$

This formulation does conserve the total kinetic energy. Indeed,

$$\begin{aligned}
 & \int_D -\mathbf{U}_h \cdot (\zeta \mathbf{k} \times \mathbf{U}_h) \, dv \\
 \equiv & \sum_{i,j,k} \left\{ \left[ \sum_{i_p, k_p}^{i+1/2-i_p} \mathbb{Q}_{j_p}^{i_p} V_{j+j_p}^{i+1/2-i_p} \right] U_j^{i+1/2} - \left[ \sum_{i_p, k_p}^i \mathbb{Q}_{j_p}^{i_p} U_{j+1/2-j_p}^{i+i_p} \right] V_{j+1/2}^i \right\} \\
 \equiv & \sum_{i,j,k} \sum_{i_p, k_p} \left\{ \begin{aligned} & \mathbb{Q}_{j_p}^{i_p} V_{j+j_p}^{i+1/2-i_p} U_j^{i+1/2} - \mathbb{Q}_{j_p}^{i_p} U_{j+1/2-j_p}^{i+i_p} V_{j+1/2}^i \end{aligned} \right\}
 \end{aligned}$$

Expanding the summation on  $i_p$  and  $k_p$ , it becomes:

$$\begin{aligned}
 \equiv & \sum_{i,j,k} \left\{ \begin{aligned} & \mathbb{Q}_{+1/2}^{-1/2} V_{j+1/2}^{i+1} U_j^{i+1/2} - \mathbb{Q}_{+1/2}^{-1/2} U_j^{i-1/2} V_{j+1/2}^i \\ & + \mathbb{Q}_{-1/2}^{-1/2} V_{j-1/2}^{i+1} U_j^{i+1/2} - \mathbb{Q}_{-1/2}^{-1/2} U_{j+1}^{i-1/2} V_{j+1/2}^i \\ & + \mathbb{Q}_{+1/2}^{+1/2} V_{j+1/2}^i U_j^{i+1/2} - \mathbb{Q}_{+1/2}^{+1/2} U_j^{i+1/2} V_{j+1/2}^i \\ & + \mathbb{Q}_{-1/2}^{+1/2} V_{j-1/2}^i U_j^{i+1/2} - \mathbb{Q}_{-1/2}^{+1/2} U_{j+1}^{i+1/2} V_{j+1/2}^i \end{aligned} \right\}
 \end{aligned}$$

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:

$$\begin{aligned}
 \equiv & \sum_{i,j,k} \left\{ \begin{aligned} & \mathbb{Q}_{+1/2}^{-1/2} \left[ V_{j+1/2}^i U_j^{i-1/2} - U_j^{i-1/2} V_{j+1/2}^i \right] \\ & + \mathbb{Q}_{-1/2}^{-1/2} \left[ V_{j-1/2}^i U_j^{i-1/2} - U_j^{i-1/2} V_{j-1/2}^i \right] \\ & + \mathbb{Q}_{+1/2}^{+1/2} \left[ V_{j+1/2}^i U_j^{i+1/2} - U_j^{i+1/2} V_{j+1/2}^i \right] \\ & + \mathbb{Q}_{-1/2}^{+1/2} \left[ V_{j-1/2}^i U_j^{i+1/2} - U_{j-1}^{i+1/2} V_{j-1/2}^i \right] \end{aligned} \right\} \equiv 0
 \end{aligned}$$

### 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  $\delta$  operator) and the continuity equation, then [equation 3.4](#) again, then the commutativity of operators  $\bar{\cdot}$  and  $\delta$ , 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:

$$\begin{aligned}
 \equiv & \sum_{i,j,k} \frac{1}{2} \left\{ \overline{W}^{i+1/2} \delta_{k+1/2} [u^2] + \overline{W}^{j+1/2} \delta_{k+1/2} [v^2] \right\} - \sum_{i,j,k} \left( \frac{u^2}{2} \partial_t b_u + \frac{v^2}{2} \partial_t b_v \right) \\
 \equiv & \sum_{i,j,k} \left\{ \overline{W}^{i+1/2} \overline{u}^{k+1/2} \delta_{k+1/2} [u] + \overline{W}^{j+1/2} \overline{v}^{k+1/2} \delta_{k+1/2} [v] \right\} - \sum_{i,j,k} \left( \frac{u^2}{2} \partial_t b_u + \frac{v^2}{2} \partial_t b_v \right) \\
 \equiv & \sum_{i,j,k} \left\{ \frac{1}{\overline{b_u}} \overline{\overline{W}^{i+1/2} \delta_{k+1/2} [u]}^k u \, b_u + \frac{1}{\overline{b_v}} \overline{\overline{W}^{j+1/2} \delta_{k+1/2} [v]}^k v \, b_v \right\} - \sum_{i,j,k} \left( \frac{u^2}{2} \partial_t b_u + \frac{v^2}{2} \partial_t b_v \right)
 \end{aligned}$$

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  $1/2 (\overline{u}^i + \overline{v}^j)$ . This leads to the following expression for the vertical advection:

$$\frac{1}{e_3} \omega \partial_k \mathbf{U}_h \equiv \left( \begin{array}{c} \frac{1}{e_{1u} e_{2u} e_{3u}} \overline{\overline{e_{1t} e_{2t} \omega \delta_{k+1/2} [\overline{u}^{i+1/2}]}}^{i+1/2,k} \\ \frac{1}{e_{1v} e_{2v} e_{3v}} \overline{\overline{e_{1t} e_{2t} \omega \delta_{k+1/2} [\overline{v}^{j+1/2}]}}^{j+1/2,k} \end{array} \right)$$

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:

$$\begin{aligned}
 e_{1u} e_{2u} \partial_t (e_{3u}) &= \overline{\overline{e_{1t} e_{2t} \partial_t (e_{3t})}}^{i+1/2} \\
 e_{1v} e_{2v} \partial_t (e_{3v}) &= \overline{\overline{e_{1t} e_{2t} \partial_t (e_{3t})}}^{j+1/2}
 \end{aligned}$$

which is (over-)satisfied by defining the vertical scale factor as follows:

$$\begin{aligned}
 e_{3u} &= \frac{1}{e_{1u} e_{2u}} \overline{\overline{e_{1t} e_{2t} e_{3t}}}^{i+1/2} \\
 e_{3v} &= \frac{1}{e_{1v} e_{2v}} \overline{\overline{e_{1t} e_{2t} e_{3t}}}^{j+1/2}
 \end{aligned}$$

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  $\delta$  operator, [equation 5.1](#), 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:

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

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):

$$\begin{aligned}
 & - \int_D u \cdot \nabla \cdot (\mathbf{U} u) \, dv \\
 \equiv & - \sum_{i,j,k} \left\{ \frac{1}{b_u} \left( \delta_{i+1/2} [\bar{U}^i \bar{u}^i] + \delta_j [\bar{V}^{i+1/2} \bar{u}^{j+1/2}] + \delta_k [\bar{W}^{i+1/2} \bar{u}^{k+1/2}] \right) \right\} b_u u \\
 \equiv & - \sum_{i,j,k} \left\{ \delta_{i+1/2} [\bar{U}^i \bar{u}^i] + \delta_j [\bar{V}^{i+1/2} \bar{u}^{j+1/2}] + \delta_k [\bar{W}^{i+1/2} \bar{u}^{k+1/2}] \right\} u \\
 \equiv & + \sum_{i,j,k} \left\{ \bar{U}^i \bar{u}^i \delta_i [u] + \bar{V}^{i+1/2} \bar{u}^{j+1/2} \delta_{j+1/2} [u] + \bar{W}^{i+1/2} \bar{u}^{k+1/2} \delta_{k+1/2} [u] \right\} \\
 \equiv & + \frac{1}{2} \sum_{i,j,k} \left\{ \bar{U}^i \delta_i [u^2] + \bar{V}^{i+1/2} \delta_{j+1/2} [u^2] + \bar{W}^{i+1/2} \delta_{k+1/2} [u^2] \right\} \\
 \equiv & - \sum_{i,j,k} \frac{1}{2} \left\{ U \delta_{i+1/2} [\bar{u}^2] + V \delta_{j+1/2} [\bar{u}^2] + W \delta_{k+1/2} [\bar{u}^2] \right\} \\
 \equiv & - \sum_{i,j,k} \frac{1}{2} \bar{u}^i \left\{ \delta_{i+1/2} [U] + \delta_{j+1/2} [V] + \delta_{k+1/2} [W] \right\} \\
 \equiv & + \sum_{i,j,k} \frac{1}{2} \bar{u}^i \left\{ \left( \frac{1}{e_{3t}} \frac{\partial e_{3t}}{\partial t} \right) b_t \right\}
 \end{aligned}$$

Applying similar manipulation applied to the second term of the scalar product leads to:

$$- \int_D \mathbf{U}_h \cdot \left( \begin{array}{c} \nabla \cdot (\mathbf{U} u) \\ \nabla \cdot (\mathbf{U} v) \end{array} \right) dv \equiv + \sum_{i,j,k} \frac{1}{2} (\bar{u}^i + \bar{v}^j) \left\{ \left( \frac{1}{e_{3t}} \frac{\partial e_{3t}}{\partial t} \right) b_t \right\}$$

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:

$$\left\{ \begin{array}{l} + \frac{1}{e_{1u}} \bar{q}^i \overline{\overline{(e_{1v} e_{3v} v)}}^{i,j+1/2} \\ - \frac{1}{e_{2v}} \bar{q}^j \overline{\overline{(e_{2u} e_{3u} u)}}^{i+1/2,j} \end{array} \right. \quad (\text{C.9})$$

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 5.2](#), 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  $\delta$  operators commute:  $\delta_{i+1/2} [\bar{a}^i] = \bar{\delta}_i [a]^{i+1/2}$ , and introducing the horizontal divergence  $\chi$ , 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:

$$\begin{cases} +q e_3 v \equiv +\frac{1}{e_{1u}} \sum_{i_p, k_p}^{i+1/2-i_p} \mathbb{Q}_{j_p}^{i_p} (e_{1v} e_{3v} v)_{j+j_p}^{i+i_p-1/2} \\ -q e_3 u \equiv -\frac{1}{e_{2v}} \sum_{i_p, k_p}^{j+1/2-j_p} \mathbb{Q}_{j_p}^{i_p} (e_{2u} e_{3u} u)_{j+j_p-1/2}^{i+i_p} \end{cases} \quad (\text{C.11})$$

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,  ${}_j^i \mathbb{Q}_{j_p}^{i_p}$ , defined at  $T$ -point, are given by:

$${}_j^i \mathbb{Q}_{j_p}^{i_p} = \frac{1}{12} \left( q_{j+j_p}^{i-i_p} + q_{j+i_p}^{i+j_p} + q_{j-j_p}^{i+i_p} \right) \quad (\text{C.7})$$

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  ${}_j^i \mathbb{Q}_{+1/2}^{+1/2}$ , 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:

$$\begin{aligned}
 & \int_D q \mathbf{k} \cdot \frac{1}{e_3} \nabla \times (e_3 q \mathbf{k} \times \mathbf{U}_h) dv \\
 & \equiv \sum_{i,j,k} q \left\{ \delta_{i+1/2} \left[ -{}_j^i \mathbb{Q}_{+1/2}^{+1/2} U_j^{i+1/2} \right] - \delta_{j+1/2} \left[ {}_j^i \mathbb{Q}_{+1/2}^{+1/2} V_{j+1/2}^i \right] \right\} \\
 & \equiv \sum_{i,j,k} \left\{ \delta_i [q] {}_j^i \mathbb{Q}_{+1/2}^{+1/2} U_j^{i+1/2} + \delta_j [q] {}_j^i \mathbb{Q}_{+1/2}^{+1/2} V_{j+1/2}^i \right\} \\
 & \dots \\
 & \quad \text{Demonstration to be done...} \\
 & \dots \\
 & \equiv \frac{1}{2} \sum_{i,j,k} \left\{ \delta_i \left[ \left( {}_j^i \mathbb{Q}_{+1/2}^{+1/2} \right)^2 \right] \bar{U}^{i,j+1/2} + \delta_j \left[ \left( {}_j^i \mathbb{Q}_{+1/2}^{+1/2} \right)^2 \right] \bar{V}^{i+1/2,j} \right\} \\
 & \equiv -\frac{1}{2} \sum_{i,j,k} \left( {}_j^i \mathbb{Q}_{+1/2}^{+1/2} \right)^2 \left\{ \delta_{i+1/2} \left[ \bar{U}^{i,j+1/2} \right] + \delta_{j+1/2} \left[ \bar{V}^{i+1/2,j} \right] \right\} \\
 & \equiv \sum_{i,j,k} -\frac{1}{2} \left( {}_j^i \mathbb{Q}_{+1/2}^{+1/2} \right)^2 \overline{\overline{b_t \chi}}^{i+1/2, j+1/2} \\
 & \equiv 0
 \end{aligned}$$

## 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.* 2<sup>nd</sup> 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  $\tau_u$ ,  $\tau_v$ , and  $\tau_w$  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 demonstarted as follows:

$$\begin{aligned} & \int_D \frac{1}{e_3} Q \frac{\partial (e_3 T)}{\partial t} \, dv = - \int_D \tau \nabla \cdot (T \mathbf{U}) \, dv \\ & \equiv - \sum_{i,j,k} T \left\{ \delta_i [U \bar{T}^{i+1/2}] + \delta_j [V \bar{T}^{j+1/2}] + \delta_k [W \bar{T}^{k+1/2}] \right\} \\ & \equiv + \sum_{i,j,k} \left\{ U \bar{T}^{i+1/2} \delta_{i+1/2} [T] + V \bar{T}^{j+1/2} \delta_{j+1/2} [T] + W \bar{T}^{k+1/2} \delta_{k+1/2} [T] \right\} \\ & \equiv + \frac{1}{2} \sum_{i,j,k} \left\{ U \delta_{i+1/2} [T^2] + V \delta_{j+1/2} [T^2] + W \delta_{k+1/2} [T^2] \right\} \\ & \equiv - \frac{1}{2} \sum_{i,j,k} T^2 \left\{ \delta_i [U] + \delta_j [V] + \delta_k [W] \right\} \\ & \equiv + \frac{1}{2} \sum_{i,j,k} T^2 \left\{ \frac{1}{e_{3t}} \frac{\partial e_{3t} T}{\partial t} \right\} \end{aligned}$$

which is the discrete form of  $\frac{1}{2} \int_D T^2 \frac{1}{e_3} \frac{\partial e_3}{\partial t} \, dv$ .

## C.7. Conservation properties on lateral momentum physics

The discrete formulation of the horizontal diffusion of momentum ensures the conservation of potential vorticity and the horizontal divergence, and the dissipation of the square of these quantities (*i.e.* enstrophy and the variance of the horizontal divergence) as well as the dissipation of the horizontal kinetic energy. In particular,

when the eddy coefficients are horizontally uniform, it ensures a complete separation of vorticity and horizontal divergence fields, so that diffusion (dissipation) of vorticity (enstrophy) does not generate horizontal divergence (variance of the horizontal divergence) and *vice versa*.

These properties of the horizontal diffusion operator are a direct consequence of properties [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  $\chi$  and dissipates  $\chi^2$  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:

$$\int_D \frac{1}{e_3} \frac{\partial}{\partial k} \left( \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) dv = \vec{0}$$

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:

$$\begin{aligned} & \int_D \chi \nabla \cdot \left( \frac{1}{e_3} \frac{\partial}{\partial k} \left( \frac{A^{vm}}{e_3} \frac{\partial \mathbf{U}_h}{\partial k} \right) \right) dv \\ & \equiv \sum_{i,j,k} \frac{\chi}{e_{1t} e_{2t}} \left\{ \delta_{i+1/2} \left( \frac{e_{2u}}{e_{3u}} \delta_k \left[ \frac{A_u^{vm}}{e_{3uw}} \delta_{k+1/2} [u] \right] \right) \right. \\ & \quad \left. + \delta_{j+1/2} \left( \frac{e_{1v}}{e_{3v}} \delta_k \left[ \frac{A_v^{vm}}{e_{3vw}} \delta_{k+1/2} [v] \right] \right) \right\} e_{1t} e_{2t} e_{3t} \\ & \equiv A^{vm} \sum_{i,j,k} \chi \left\{ \delta_{i+1/2} \left( \delta_k \left[ \frac{1}{e_{3uw}} \delta_{k+1/2} [e_{2u} u] \right] \right) \right. \\ & \quad \left. + \delta_{j+1/2} \left( \delta_k \left[ \frac{1}{e_{3vw}} \delta_{k+1/2} [e_{1v} v] \right] \right) \right\} \\ & \equiv -A^{vm} \sum_{i,j,k} \frac{\delta_{k+1/2} [\chi]}{e_{3w}} \left\{ \delta_{k+1/2} [\delta_{i+1/2} [e_{2u} u] + \delta_{j+1/2} [e_{1v} v]] \right\} \\ & \equiv -A^{vm} \sum_{i,j,k} \frac{1}{e_{3w}} \delta_{k+1/2} [\chi] \delta_{k+1/2} [e_{1t} e_{2t} \chi] \\ & \equiv -A^{vm} \sum_{i,j,k} \frac{e_{1t} e_{2t}}{e_{3w}} (\delta_{k+1/2} [\chi])^2 \equiv 0 \end{aligned}$$

## 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:

$$\begin{aligned} & \int_D \nabla \cdot (A \nabla T) dv \\ & \equiv \sum_{i,j,k} \left\{ \delta_i \left[ A_u^{iT} \frac{e_{2u} e_{3u}}{e_{1u}} \delta_{i+1/2} [T] \right] + \delta_j \left[ A_v^{iT} \frac{e_{1v} e_{3v}}{e_{2v}} \delta_{j+1/2} [T] \right] \right. \\ & \quad \left. + \delta_k \left[ A_w^{iT} \frac{e_{1t} e_{2t}}{e_{3t}} \delta_{k+1/2} [T] \right] \right\} \equiv 0 \end{aligned}$$







# Iso-Neutral Diffusion and Eddy Advection using Triads

## Table of contents

- D.1. Choice of `namtra_ldf` namelist parameters . . . . . 266
- D.2. Triad formulation of iso-neutral diffusion . . . . . 266
  - D.2.1. Iso-neutral diffusion operator . . . . . 266
  - D.2.2. Standard discretization . . . . . 267
  - D.2.3. Expression of the skew-flux in terms of triad slopes . . . . . 267
  - D.2.4. Full triad fluxes . . . . . 269
  - D.2.5. Ensuring the scheme does not increase tracer variance . . . . . 269
  - D.2.6. Triad volumes in Griffes’s scheme and in *NEMO* . . . . . 270
  - D.2.7. Summary of the scheme . . . . . 271
  - D.2.8. Treatment of the triads at the boundaries . . . . . 272
  - D.2.9. Limiting of the slopes within the interior . . . . . 272
  - D.2.10. Tapering within the surface mixed layer . . . . . 272
- D.3. Eddy induced advection formulated as a skew flux . . . . . 275
  - D.3.1. Continuous skew flux formulation . . . . . 275
  - D.3.2. Discrete skew flux formulation . . . . . 276
  - D.3.3. Treatment of the triads at the boundaries . . . . . 277
  - D.3.4. Limiting of the slopes within the interior . . . . . 277
  - D.3.5. Tapering within the surface mixed layer . . . . . 277
  - D.3.6. Streamfunction diagnostics . . . . . 277

## Changes record

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

## D.1. Choice of `&namtra_ldf` (namelist 6.2) 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 10.2). 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 10.3. 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 (subsubsection D.2.10). This is the same treatment as used in the default implementation subsection 10.2.2; figure 10.2. 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 subsubsection 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:

$$D^{lT} = -\nabla \cdot f^{lT} \equiv -\frac{1}{e_1 e_2 e_3} \left[ \frac{\partial}{\partial i} (f_1^{lT} e_2 e_3) + \frac{\partial}{\partial j} (f_2^{lT} e_2 e_3) + \frac{\partial}{\partial k} (f_3^{lT} e_1 e_2) \right], \quad (\text{D.1a})$$

where the diffusive flux per unit area of physical space

$$f^{lT} = -A^{lT} \mathfrak{R} \cdot \nabla T, \quad (\text{D.1b})$$

$$\text{with } \mathfrak{R} = \begin{pmatrix} 1 & 0 & -r_1 \\ 0 & 1 & -r_2 \\ -r_1 & -r_2 & r_1^2 + r_2^2 \end{pmatrix} \quad \text{and} \quad \nabla T = \begin{pmatrix} \frac{1}{e_1} \frac{\partial T}{\partial i} \\ \frac{1}{e_2} \frac{\partial T}{\partial j} \\ \frac{1}{e_3} \frac{\partial T}{\partial k} \end{pmatrix}. \quad (\text{D.1c})$$

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  $F_{\text{iso}}$  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 6.8](#), 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  $e_{3_{i+1/2}^k}$  in the area  $e_{3_{i+1/2}^k} e_{2_{i+1/2}^k} i^k$  at the  $u$ -point cancels out with the  $1/e_{3_{i+1/2}^k}$  associated with the vertical tracer gradient, is then [equation 6.8](#)

$$(F_u^{13})_{i+\frac{1}{2}}^k = A_{i+\frac{1}{2}}^k e_{2_{i+1/2}^k} \overline{\overline{r_1}}^{i,k} \overline{\overline{\delta_k T}}^{i,k},$$

where

$$\overline{\overline{r_1}}^{i,k} = -\frac{e_{3_{i+1/2}^k} \delta_{i+1/2}[\rho]}{e_{1_{i+1/2}^k} \overline{\overline{\delta_k \rho}}^{i,k}},$$

and here and in the following we drop the  $lT$  superscript from  $A^{lT}$  for simplicity. Unfortunately the resulting combination  $\overline{\overline{\delta_k \bullet}}^{i,k}$  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 10](#)). 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



### 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

$$(F_u^{11})_{i+\frac{1}{2}}^k = - (A_i^{k+1} a_1 + A_i^{k+1} a_2 + A_i^k a_3 + A_i^k a_4) \frac{\delta_{i+1/2} [T^k]}{e_{1u}^k}_{i+1/2}, \quad (\text{D.8})$$

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

$${}^k \mathbb{F}_{u_{i_p}}^{k_p}(T) = -A_i^k {}^k \mathbb{A}_{u_{i_p}}^{k_p} \left( \frac{\delta_{i+i_p} [T^k]}{e_{1u}^k}_{i+i_p} - {}^k \mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p} [T^i]}{e_{3w}^k}_{k+k_p} \right) \quad (\text{D.9})$$

can be identified with each triad. Then, because the same metric factors  $e_{3w}^k$  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

$$(F_w^{33})_i^{k+\frac{1}{2}} = - (A_i^{k+1} a'_1 s_1'^2 + A_i^{k+1} a'_2 s_2'^2 + A_i^k a'_3 s_3'^2 + A_i^k a'_4 s_4'^2) \delta_{k+\frac{1}{2}} [T^{i+1}], \quad (\text{D.11})$$

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

$${}^k \mathbb{F}_{w_{i_p}}^{k_p}(T) = A_i^k {}^k \mathbb{A}_{w_{i_p}}^{k_p} \left( {}^k \mathbb{R}_{i_p}^{k_p} \frac{\delta_{i+i_p} [T^k]}{e_{1u}^k}_{i+i_p} - \left( {}^k \mathbb{R}_{i_p}^{k_p} \right)^2 \frac{\delta_{k+k_p} [T^i]}{e_{3w}^k}_{k+k_p} \right) \quad (\text{D.12})$$

$$= - \left( {}^k \mathbb{A}_{w_{i_p}}^{k_p} / {}^k \mathbb{A}_{u_{i_p}}^{k_p} \right) {}^k \mathbb{R}_{i_p}^{k_p} {}^k \mathbb{F}_{u_{i_p}}^{k_p}(T) \quad (\text{D.13})$$

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  $(F_w^{31})_i^{k+\frac{1}{2}} + (F_w^{33})_i^{k+\frac{1}{2}}$  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:

$$F_{\text{iso}}(T) \equiv \sum_{i_p, k_p} \left( \begin{array}{c} {}^k \\ i+1/2-i_p \end{array} \mathbb{F}_{u_{i_p}}^{k_p}(T) \right) \quad (\text{D.14})$$

### 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

$${}^k_i \mathbb{F}_{w_{i_p}}{}^{k_p}(T) \delta_{k+k_p}[T^i]. \quad (\text{D.16})$$

The total variance tendency driven by the triad is the sum of these two. Expanding  ${}^k_i \mathbb{F}_{u_{i_p}}{}^{k_p}(T)$  and  ${}^k_i \mathbb{F}_{w_{i_p}}{}^{k_p}(T)$  with [equation D.9](#) and [equation D.12](#), it is

$$\begin{aligned} -A_i^k \left\{ {}^k_i \mathbb{A}_{u_{i_p}}{}^{k_p} \left( \frac{\delta_{i+i_p}[T^k]}{e_{1u}{}^k_{i+i_p}} - {}^k_i \mathbb{R}_{i_p}{}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w}{}^k_i} \right) \delta_{i+i_p}[T^k] \right. \\ \left. - {}^k_i \mathbb{A}_{w_{i_p}}{}^{k_p} \left( \frac{\delta_{i+i_p}[T^k]}{e_{1u}{}^k_{i+i_p}} - {}^k_i \mathbb{R}_{i_p}{}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w}{}^k_i} \right) {}^k_i \mathbb{R}_{i_p}{}^{k_p} \delta_{k+k_p}[T^i] \right\}. \end{aligned}$$

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

$${}^k_i \mathbb{V}_{i_p}{}^{k_p} = {}^k_i \mathbb{A}_{u_{i_p}}{}^{k_p} e_{1u}{}^k_{i+i_p} = {}^k_i \mathbb{A}_{w_{i_p}}{}^{k_p} e_{3w}{}^k_i{}^{k+k_p}, \quad (\text{D.17})$$

the variance tendency reduces to the perfect square

$$-A_i^k {}^k_i \mathbb{V}_{i_p}{}^{k_p} \left( \frac{\delta_{i+i_p}[T^k]}{e_{1u}{}^k_{i+i_p}} - {}^k_i \mathbb{R}_{i_p}{}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w}{}^k_i{}^{k+k_p}} \right)^2 \leq 0. \quad (\text{D.18})$$

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

$$\nabla T = \left( \delta_{i+i_p}[T^k] / e_{1u}{}^k_{i+i_p}, \delta_{k+k_p}[T^i] / e_{3w}{}^k_i{}^{k+k_p} \right)$$

### D.2.6. Triad volumes in Griffies's scheme and in NEMO

To complete the discretization we now need only specify the triad volumes  ${}^k_i \mathbb{V}_{i_p}{}^{k_p}$ . [Griffies 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

$$-\overline{A}_{i+1/2}{}^k \frac{b_{u_{i+1/2}}{}^k}{e_{1u}{}^k_{i+i_p}} \frac{\delta_{i+1/2}[T^k]}{e_{1u}{}^k_{i+i_p}} = -\overline{A}_{i+1/2}{}^k \frac{e_{1w}{}^k_{i+1/2} e_{1v}{}^k_{i+1/2} \delta_{i+1/2}[T^k]}{e_{1u}{}^k_{i+1/2}}. \quad (\text{D.21})$$

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  $\overline{A}_{i+1/2}{}^k$  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](#):

$$F_{\text{iso}}(T) \equiv \sum_{i_p, k_p} \begin{pmatrix} {}^k_{i+1/2-i_p} \mathbb{F}_{u i_p}^{k_p}(T) \\ {}^{k+1/2-k_p} \mathbb{F}_{w i_p}^{k_p}(T) \end{pmatrix},$$

where [equation D.9](#):

$${}^k \mathbb{F}_{u i_p}^{k_p}(T) = -A_i^k \frac{{}^k \mathbb{V}_{i_p}^{k_p}}{e_{1u} {}^k_{i+i_p}} \left( \frac{\delta_{i+i_p} [T^k]}{e_{1u} {}^k_{i+i_p}} - {}^k \mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p} [T^i]}{e_{3w} {}^k_{i+i_p}} \right), \quad (\text{D.22a})$$

and

$${}^k \mathbb{F}_{w i_p}^{k_p}(T) = A_i^k \frac{{}^k \mathbb{V}_{i_p}^{k_p}}{e_{3w} {}^k_{i+i_p}} \left( {}^k \mathbb{R}_{i_p}^{k_p} \frac{\delta_{i+i_p} [T^k]}{e_{1u} {}^k_{i+i_p}} - \left( {}^k \mathbb{R}_{i_p}^{k_p} \right)^2 \frac{\delta_{k+k_p} [T^i]}{e_{3w} {}^k_{i+i_p}} \right), \quad (\text{D.22b})$$

with [equation D.20](#)

$${}^k \mathbb{V}_{i_p}^{k_p} = \frac{1}{4} b_{u i+i_p}^k.$$

The divergence of the expression [equation D.14](#) for the fluxes gives the iso-neutral diffusion tendency at each tracer point:

$$D_l^T = \frac{1}{b_T} \sum_{i_p, k_p} \left\{ \delta_i \left[ {}^k_{i+1/2-i_p} \mathbb{F}_{u i_p}^{k_p} \right] + \delta_k \left[ {}^{k+1/2-k_p} \mathbb{F}_{w i_p}^{k_p} \right] \right\}$$

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:

$$D_l^T = \frac{1}{b_T} \delta_i \left[ \frac{e_{2u} e_{3u}}{e_{1u}} \bar{A}^i \delta_{i+1/2} [T] \right] \quad \text{when} \quad {}^k \mathbb{R}_{i_p}^{k_p} = 0$$

**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,

$$\frac{1}{b_w} \sum_{i_p, k_p} \left\{ {}^k \mathbb{V}_{i_p}^{k_p} A_i^k \left( {}^k \mathbb{R}_{i_p}^{k_p} \right)^2 \right\} = \frac{1}{4b_w} \sum_{i_p, k_p} \left\{ b_{u i+i_p}^k A_i^k \left( {}^k \mathbb{R}_{i_p}^{k_p} \right)^2 \right\},$$

(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](#),

$$-A_i^k \mathbb{V}_{i_p}^{k_p} \left( \frac{\delta_{i+i_p}[T^k]}{e_{1u}^k} - \mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[T^i]}{e_{3w}^{k+k_p}} \right) \left( \frac{\delta_{i+i_p}[S^k]}{e_{1u}^k} - \mathbb{R}_{i_p}^{k_p} \frac{\delta_{k+k_p}[S^i]}{e_{3w}^{k+k_p}} \right).$$

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  $\mathbb{R}_{1/2}^{-1/2}$  (magenta) and  $\mathbb{R}_{-1/2}^{-1/2}$  (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  $\mathbb{F}_{u_{1/2}}^{-1/2}$  and  $\mathbb{F}_{u_{-1/2}}^{-1/2}$ ; 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  $\mathbb{R}_{1/2}^{1/2}$  and  $\mathbb{R}_{-1/2}^{1/2}$  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.

### D.2.9. Limiting of the slopes within the interior

As discussed in [subsection 10.2.2](#), 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  $\sigma_i$  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

$$\mathbb{R}_{i_p}^{k_p} = \mathbb{R}_{i_p}^{k_p} + \frac{\delta_{i+i_p}[z_T^k]}{e_{1u}^k} \quad (\text{D.24})$$

is limited like this and then the corresponding  $\mathbb{R}_{i_p}^{k_p}$  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.

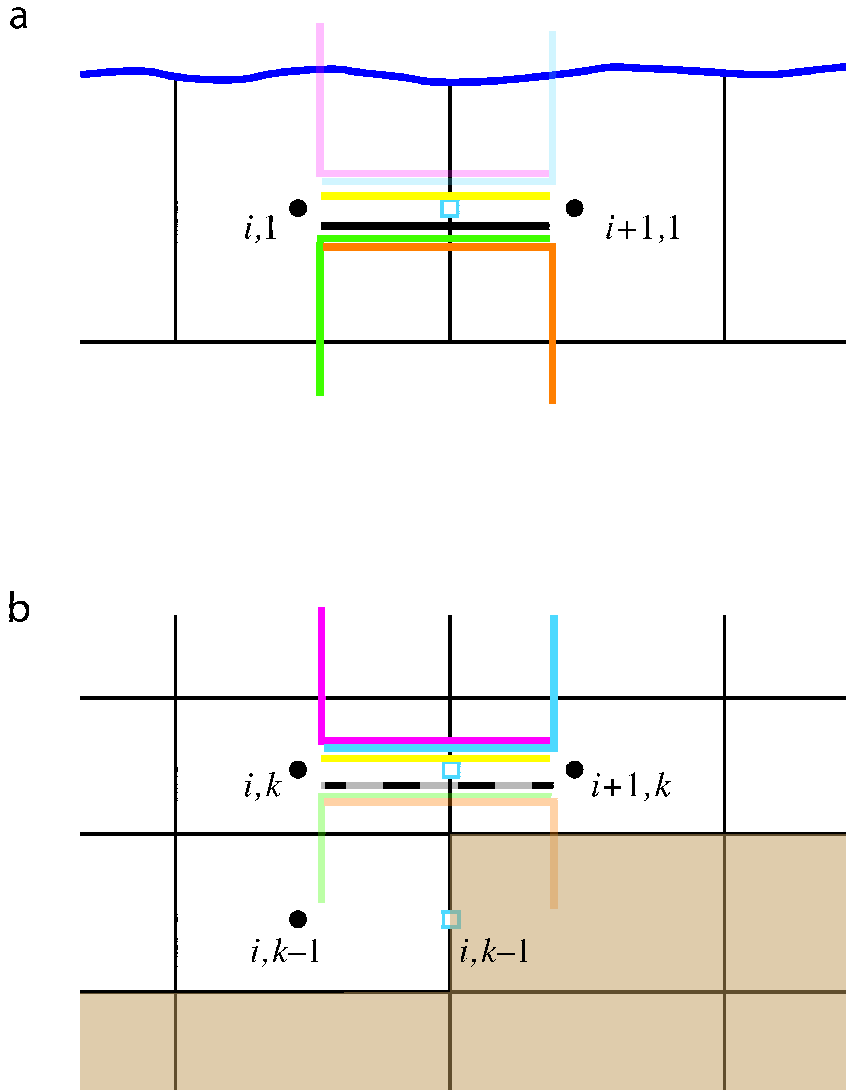


Figure D.3.: (a) Uppermost model layer  $k = 1$  with  $i, 1$  and  $i + 1, 1$  tracer points (black dots), and  $i + 1/2, 1$   $u$ -point (blue square). Triad slopes  ${}^i\mathbb{R}_{1/2}^{-1/2}$  (magenta) and  ${}^{i+1}\mathbb{R}_{-1/2}^{-1/2}$  (blue) poking through the ocean surface are masked (faded in figure). However, the lateral  ${}_{11}$  contributions towards  ${}^i\mathbb{F}_{u_{1/2}}^{-1/2}$  and  ${}^{i+1}\mathbb{F}_{u_{-1/2}}^{-1/2}$  (yellow line) are still applied, giving diapycnal diffusive fluxes. (b) Both near bottom triad slopes  ${}^i\mathbb{R}_{1/2}^{1/2}$  and  ${}^{i+1}\mathbb{R}_{-1/2}^{1/2}$  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.`

### 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 10.2, to values

$$\tilde{r}_{ML i} = -\frac{z}{h} \tilde{r}_i|_{z=-h} \quad \text{for } z > -h, \quad (\text{D.25})$$

and then the  $r_i$  relative to vertical coordinate surfaces are appropriately adjusted to

$$ML i = \tilde{r}_{ML i} - \sigma_i \quad \text{for } z > -h.$$

Thus the diffusion operator within the mixed layer is given by:

$$D^{IT} = \nabla \cdot (A^{IT} \mathfrak{R} \nabla T) \quad \text{with} \quad \mathfrak{R} = \begin{pmatrix} 1 & 0 & -ML_1 \\ 0 & 1 & -ML_2 \\ -ML_1 & -ML_2 & {}^2_{ML_1} + {}^2_{ML_2} \end{pmatrix}$$

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  ${}^k_i\mathbb{R}_{i_p}^{k_p}$ , and the  ${}^k_i\mathbb{R}_{i_p}^{k_p}$  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,  $k_{\text{ML}}$ , 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  $i, k_{\text{ML}}$  tracer point.
2. We define ‘basal’ triad slopes  ${}_i\mathbb{R}_{i_p}^{k_p}$  as the slopes of those triads whose vertical ‘arms’ go down from the  $i, k_{\text{ML}}$  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  ${}_i\mathbb{R}_{i_p}^{k_p}$  are representative of the thermocline. The four basal triads defined in the bottom part of figure D.4 are then

$${}_i\mathbb{R}_{i_p}^{k_p} = {}_i^{k_{\text{ML}}-k_p-1/2}\mathbb{R}_{i_p}^{k_p},$$

with *e.g.* the green triad

$${}_i\mathbb{R}_{i_p}^{-1/2} = {}_i^{k_{\text{ML}}-1/2}\mathbb{R}_{i_p}^{-1/2}.$$

The vertical flux associated with each of these triads passes through the  $w$ -point  $i, k_{\text{ML}} - 1/2$  lying *below* the  $i, k_{\text{ML}}$  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  $z_{\text{ML}}$ ) that sets the  $h$  used to taper the slopes in equation D.25.

3. Finally, we calculate the adjusted triads  ${}^k_i\mathbb{R}_{i_p}^{k_p}$  within the mixed layer, by multiplying the appropriate  ${}_i\mathbb{R}_{i_p}^{k_p}$  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  $i, k$

$${}^k_i\mathbb{R}_{i_p}^{-1/2} = \frac{z_{w k-1/2}}{z_{\text{base } i}} {}_i\mathbb{R}_{i_p}^{-1/2}$$

and more generally

$${}^k_i\mathbb{R}_{i_p}^{k_p} = \frac{z_{w k+k_p}}{z_{\text{base } i}} {}_i\mathbb{R}_{i_p}^{k_p}.$$

### 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  ${}^k_i\mathbb{R}_{i_p}^{k_p}$

\*To ensure good behaviour where horizontal density gradients are weak, we in fact follow Gerdes et al. (1991) and set  ${}^*_{\text{ML } i} = \text{sgn}(\tilde{r}_i) \min(|\tilde{r}_{\text{ML } i}^2 / \tilde{r}_i|, |\tilde{r}_i|) - \sigma_i$ .

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 10.1 is used in  $z$ -coordinate, and the sum equation 10.1 + equation 10.2 in  $z^*$  or  $s$ -coordinates.

The eddy induced velocity is given by:

$$\begin{aligned} u^* &= -\frac{1}{e_3} \partial_i \psi_1, \\ v^* &= -\frac{1}{e_3} \partial_j \psi_2, \\ w^* &= \frac{1}{e_1 e_2} \{ \partial_i (e_2 \psi_1) + \partial_j (e_1 \psi_2) \}, \end{aligned} \tag{D.27a}$$

where the streamfunctions  $\psi_i$  are given by

$$\begin{aligned} \psi_1 &= A_e \tilde{r}_1, \\ \psi_2 &= A_e \tilde{r}_2, \end{aligned} \tag{D.27b}$$

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 6.1) and not just a 2<sup>nd</sup> 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:

$$\mathbf{F}_{\text{eiv}}^T = \begin{pmatrix} +e_2 \psi_1 \partial_k T \\ -e_2 \psi_1 \partial_i T \end{pmatrix} \tag{D.28}$$

The total fluxes per unit physical area are then

$$\begin{aligned} f_1^* &= \frac{1}{e_3} \psi_1 \partial_k T \\ f_2^* &= \frac{1}{e_3} \psi_2 \partial_k T \\ f_3^* &= -\frac{1}{e_1 e_2} \{ e_2 \psi_1 \partial_i T + e_1 \psi_2 \partial_j T \}. \end{aligned} \tag{D.29}$$

Note that equation D.29 takes the same form whatever the vertical coordinate, though of course the slopes  $\tilde{r}_i$  which define the  $\psi_i$  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:

$$F_{\text{eiv}}(T) \equiv \sum_{i_p, k_p} \begin{pmatrix} {}^k_{i+1/2-i_p} \mathbb{S}_{u_{i_p}}^{k_p}(T) \\ {}^{k+1/2-k_p}_i \mathbb{S}_{w_{i_p}}^{k_p}(T) \end{pmatrix},$$

where the skew flux in the  $i$ -direction associated with a given triad is (equation D.9, equation D.22a):

$${}^k_i \mathbb{S}_{u_{i_p}}^{k_p}(T) = +\frac{1}{4} A_{ei} {}^k_{e_{1u} \ i+i_p} \frac{b_{u_{i+i_p}}^k}{{}^k_{e_{1u} \ i+i_p}} {}^k_i \tilde{\mathbb{R}}_{i_p}^{k_p} \frac{\delta_{k+k_p}[T^i]}{{}^k_{e_{3w} \ i}}, \quad (\text{D.30a})$$

and equation D.22b in the  $k$ -direction, changing the sign to be consistent with equation D.28:

$${}^k_i \mathbb{S}_{w_{i_p}}^{k_p}(T) = -\frac{1}{4} A_{ei} {}^k_{e_{3w} \ i} \frac{b_{u_{i+i_p}}^k}{{}^k_{e_{3w} \ i}} {}^k_i \tilde{\mathbb{R}}_{i_p}^{k_p} \frac{\delta_{i+i_p}[T^k]}{{}^k_{e_{1u} \ i+i_p}}. \quad (\text{D.30b})$$

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  ${}^k_i \tilde{\mathbb{R}}_{i_p}^{k_p}(T)$ . For, following subsection D.2.5 and equation D.15, the associated horizontal skew-flux  ${}^k_i \mathbb{S}_{u_{i_p}}^{k_p}(T)$  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

$${}^k_i \mathbb{S}_{u_{i_p}}^{k_p}(T) \delta_{i+i_p}[T^k], \quad (\text{D.31})$$

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

$${}^k_i \mathbb{S}_{w_{i_p}}^{k_p}(T) \delta_{k+k_p}[T^i]. \quad (\text{D.32})$$

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

$$ge_{3w} {}^{k+k_p}_i \mathbb{S}_{w_{i_p}}^{k_p}(\rho) = ge_{3w} {}^{k+k_p}_i \left[ -\alpha_i^k {}^k_i \mathbb{S}_{w_{i_p}}^{k_p}(T) + \beta_i^k {}^k_i \mathbb{S}_{w_{i_p}}^{k_p}(S) \right].$$

Substituting  ${}^k_i \mathbb{S}_{w_{i_p}}^{k_p}$  from equation D.30b, gives

$$\begin{aligned} &= -\frac{1}{4} g A_{ei} {}^k_{b_{u_{i+i_p}}^k} {}^k_i \tilde{\mathbb{R}}_{i_p}^{k_p} \frac{-\alpha_i^k \delta_{i+i_p}[T^k] + \beta_i^k \delta_{i+i_p}[S^k]}{{}^k_{e_{1u} \ i+i_p}} \\ &= +\frac{1}{4} g A_{ei} {}^k_{b_{u_{i+i_p}}^k} \left( {}^k_i \tilde{\mathbb{R}}_{i_p}^{k_p} + \frac{\delta_{i+i_p}[z_T^k]}{{}^k_{e_{1u} \ i+i_p}} \right) {}^k_i \tilde{\mathbb{R}}_{i_p}^{k_p} \frac{-\alpha_i^k \delta_{k+k_p}[T^i] + \beta_i^k \delta_{k+k_p}[S^i]}{{}^k_{e_{3w} \ i}}, \end{aligned} \quad (\text{D.33})$$

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

$$g\delta_{i+i_p}[z_T^k] \left[ -\alpha_i^k {}^k\mathbb{S}_{u_{i_p}}^{k_p}(T) + \beta_i^k {}^k\mathbb{S}_{u_{i_p}}^{k_p}(S) \right] \\ = +\frac{1}{4}gA_{e_i}^k b_{u_{i+i_p}}^k \frac{\delta_{i+i_p}[z_T^k]}{e_{1u}^k} \left( {}^k\mathbb{R}_{i_p}^{k_p} + \frac{\delta_{i+i_p}[z_T^k]}{e_{1u}^k} \right) \frac{-\alpha_i^k \delta_{k+k_p}[T^i] + \beta_i^k \delta_{k+k_p}[S^i]}{e_{3w}^k}, \quad (\text{D.34})$$

(using [equation D.30a](#)) and so the total PE change [equation D.33](#) + [equation D.34](#) associated with the triad fluxes is

$$ge_{3w}^k {}^k\mathbb{S}_{w_{i_p}}^{k_p}(\rho) + g\delta_{i+i_p}[z_T^k] {}^k\mathbb{S}_{u_{i_p}}^{k_p}(\rho) \\ = +\frac{1}{4}gA_{e_i}^k b_{u_{i+i_p}}^k \left( {}^k\mathbb{R}_{i_p}^{k_p} + \frac{\delta_{i+i_p}[z_T^k]}{e_{1u}^k} \right)^2 \frac{-\alpha_i^k \delta_{k+k_p}[T^i] + \beta_i^k \delta_{k+k_p}[S^i]}{e_{3w}^k}.$$

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  ${}^k\tilde{\mathbb{R}}_{i_p}^{k_p}$  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  ${}^k\tilde{\mathbb{R}}_{i_p}^{k_p}$ ) are always tapered linearly from their value immediately below the mixed layer to zero at the surface [equation D.25](#), as described in [subsection D.2.10](#). This is option (c) of [figure 10.2](#). 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 10.3](#)), 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:

$$\psi_{i+1/2}^{k+1/2} = \frac{1}{4} \sum_{i_p, k_p} A_{e_{i+1/2-i_p}}^{k+1/2-k_p} {}^{k+1/2-k_p}\mathbb{R}_{i+1/2-i_p}^{k_p}.$$



The streamfunction  $\psi_1$  is calculated similarly at  $vw$  points. The eddy-induced velocities are then calculated from the straightforward discretisation of [equation D.27a](#):

$$\begin{aligned}
 u_{i+1/2}^{*k} &= -\frac{1}{e_{3u_i}^k} \left( \psi_{1_{i+1/2}}^{k+1/2} - \psi_{1_{i+1/2}}^{k+1/2} \right), \\
 v_{j+1/2}^{*k} &= -\frac{1}{e_{3v_j}^k} \left( \psi_{2_{j+1/2}}^{k+1/2} - \psi_{2_{j+1/2}}^{k+1/2} \right), \\
 w_{i,j}^{*k+1/2} &= \frac{1}{e_{1t}e_{2t}} \left\{ e_{2u_{i+1/2}}^{k+1/2} \psi_{1_{i+1/2}}^{k+1/2} - e_{2u_{i-1/2}}^{k+1/2} \psi_{1_{i-1/2}}^{k+1/2} + \right. \\
 &\quad \left. e_{2v_{j+1/2}}^{k+1/2} \psi_{2_{j+1/2}}^{k+1/2} - e_{2v_{j-1/2}}^{k+1/2} \psi_{2_{j-1/2}}^{k+1/2} \right\},
 \end{aligned}$$

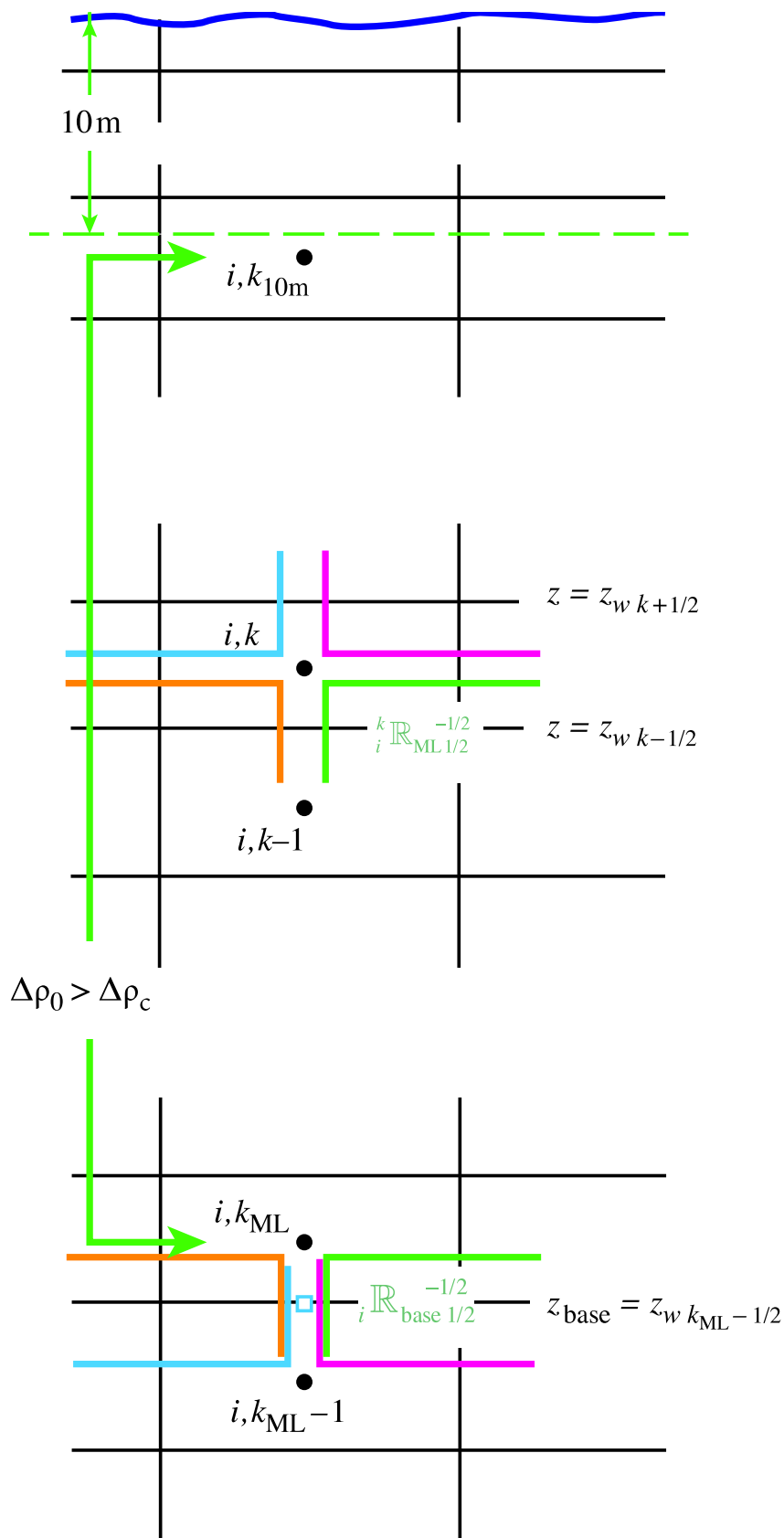


Figure D.4.: Definition of mixed-layer depth and calculation of linearly tapered triads. The figure shows a water column at a given  $i, j$  (simplified to  $i$ ), with the ocean surface at the top. Tracer points are denoted by bullets, and black lines the edges of the tracer cells;  $k$  increases upwards.

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),  ${}^k \mathbb{R}_{i, base}^{k_p}$ . 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.



# North Pole Folding

## Table of contents

E.1. North Pole Folding around a T-Point . . . . . 281  
E.2. North Pole Folding around a F-Point . . . . . 285

## Changes record

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

### E.1. North Pole Folding around a T-Point

When `l_Iperio = .true.`, `l_NFold = .true.` and `c_NFtype = 'T'` the North Pole Folding is done around 2 T-points. This is the case in ORCA 2°, 1/4°, 1/12° and 1/36°.

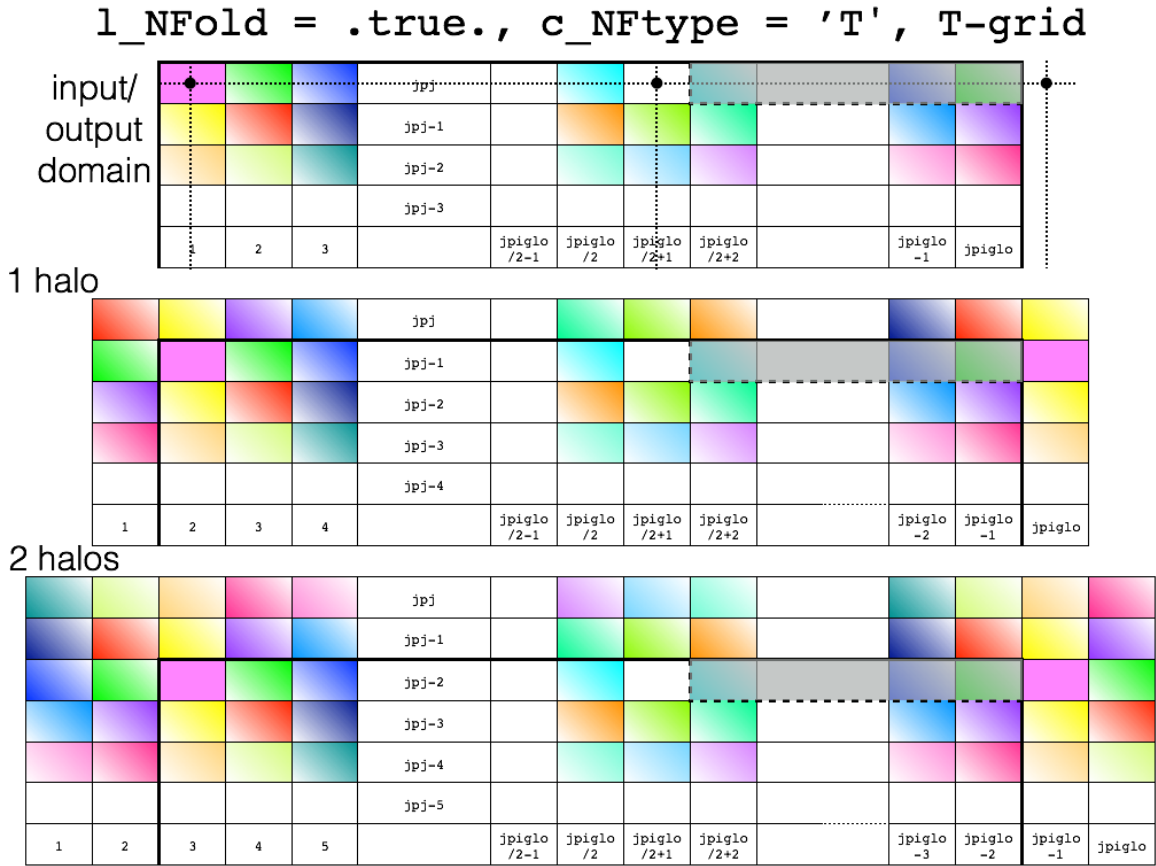


Figure E.1.: North fold boundary for the T grid, with a T-point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the `lbc_lnk` routine (found in `lbc_lnk.F90` module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by the inner domain values when calling the `lbc_lnk`. Cells with the same color are at the same geographical location. Color shading shows the change in gradient on each side of the pivot-points.

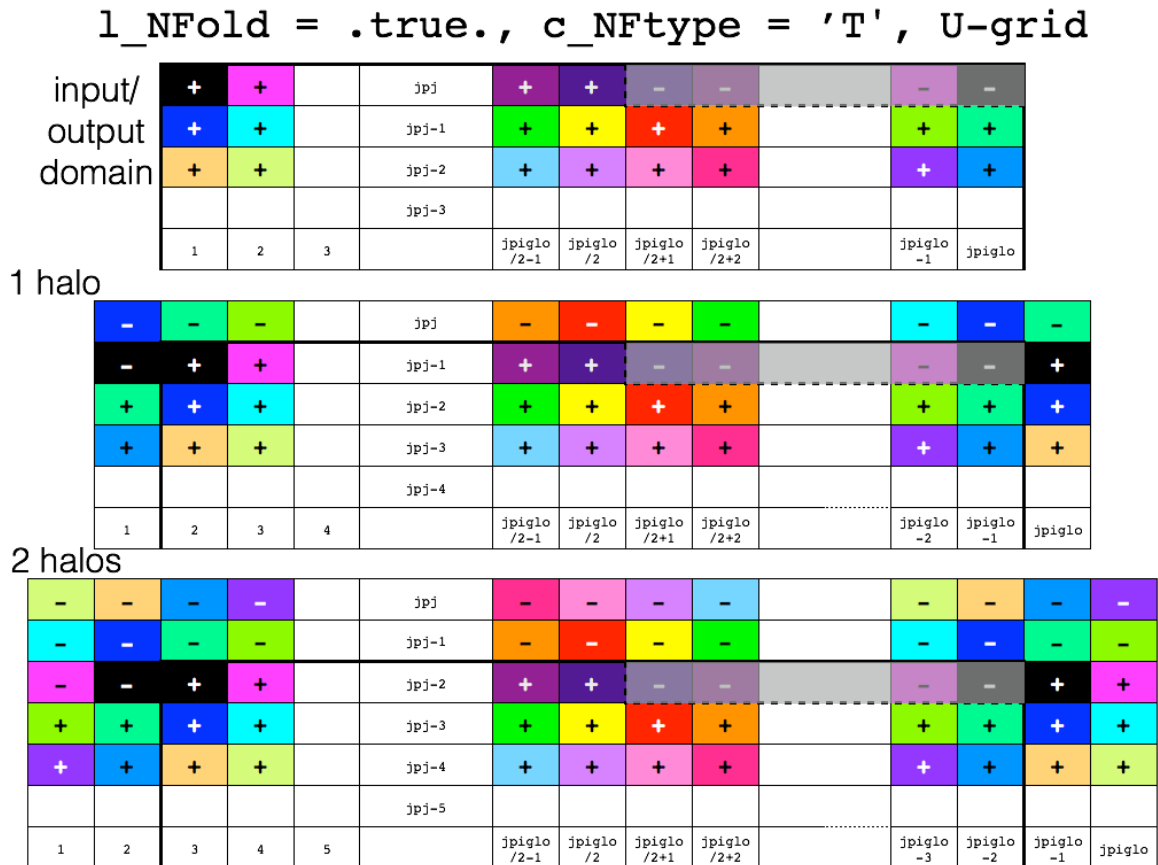


Figure E.2.: North fold boundary for the U grid, with a *T*-point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the `lbc_lnk` routine (found in `lbc_lnk.F90` module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by othe inner domain values when calling the `lbc_lnk`. Cells with the same color are at the same geographical location. Signs + or - are illustrating the change of signe on each side of the pivotal point.

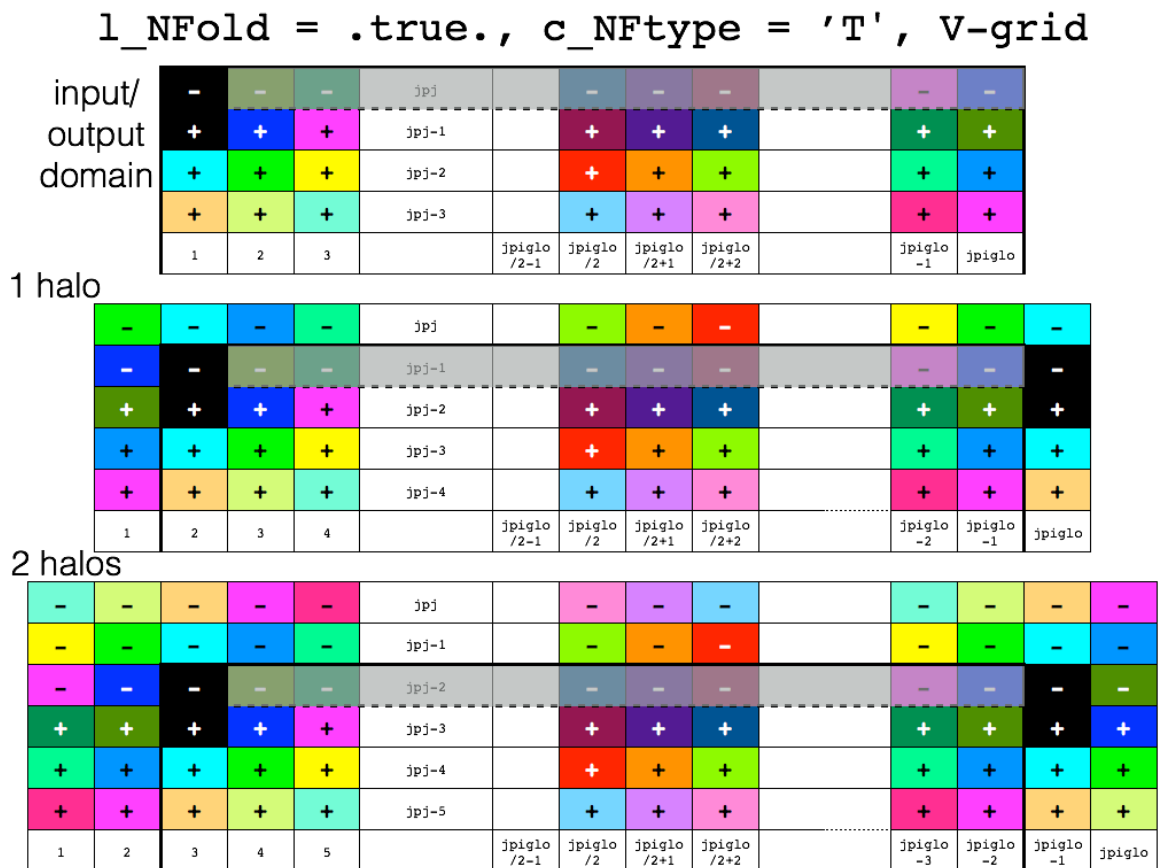


Figure E.3.: North fold boundary for the V grid, with a T-point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the `lbc_lnk` routine (found in `lbc_lnk.F90` module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by the inner domain values when calling the `lbc_lnk`. Cells with the same color are at the same geographical location. Signs + or - are illustrating the change of sign on each side of the pivotal point.





## E.2. North Pole Folding around a F-Point

When `l_Iperio = .true.`, `l_NFold = .true.` and `c_NFtype = 'F'` the North Pole Folding is done around 2 F-points. This is the case in ORCA 1° and 1/2°.

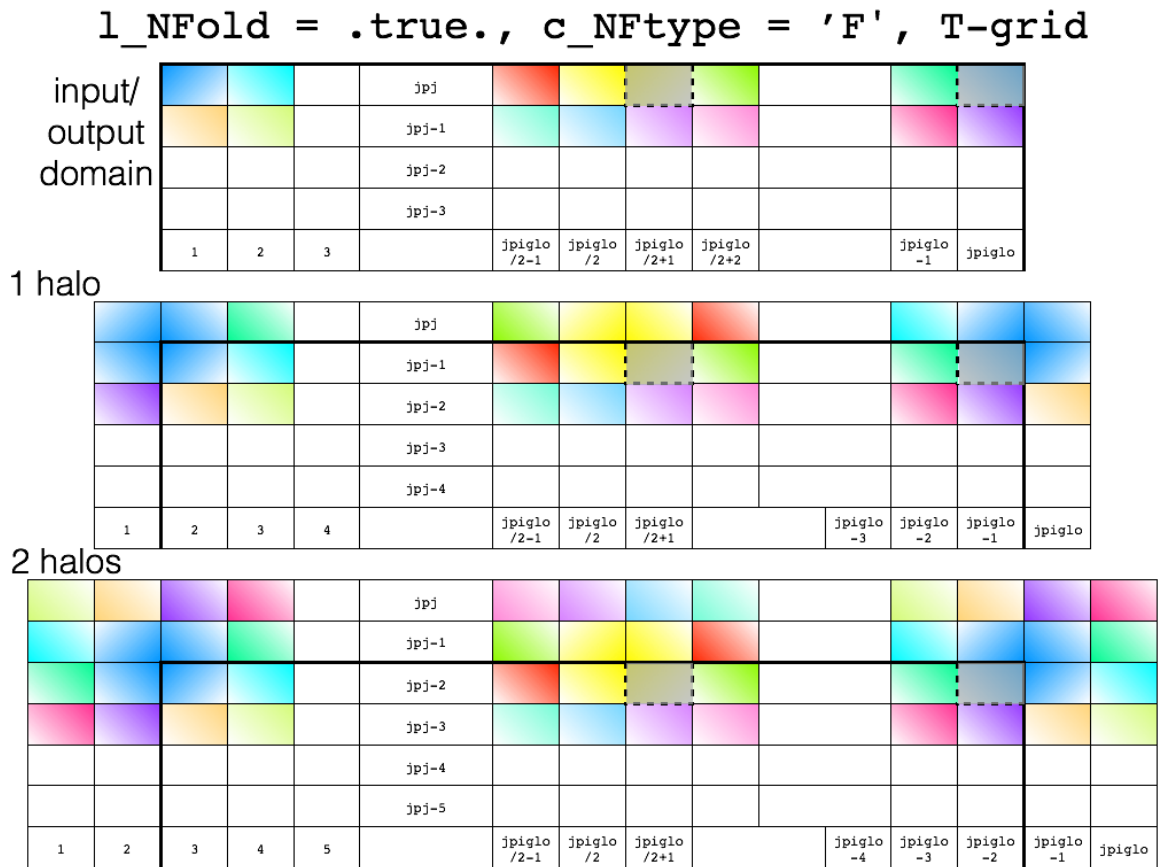


Figure E.5.: North fold boundary for the T grid, with a *F*-point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the `lbc_lnk` routine (found in `lbc_lnk.F90` module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by the inner domain values when calling the `lbc_lnk`. Cells with the same color are at the same geographical location. Color shading shows the change in gradient on each side of the pivot-points.

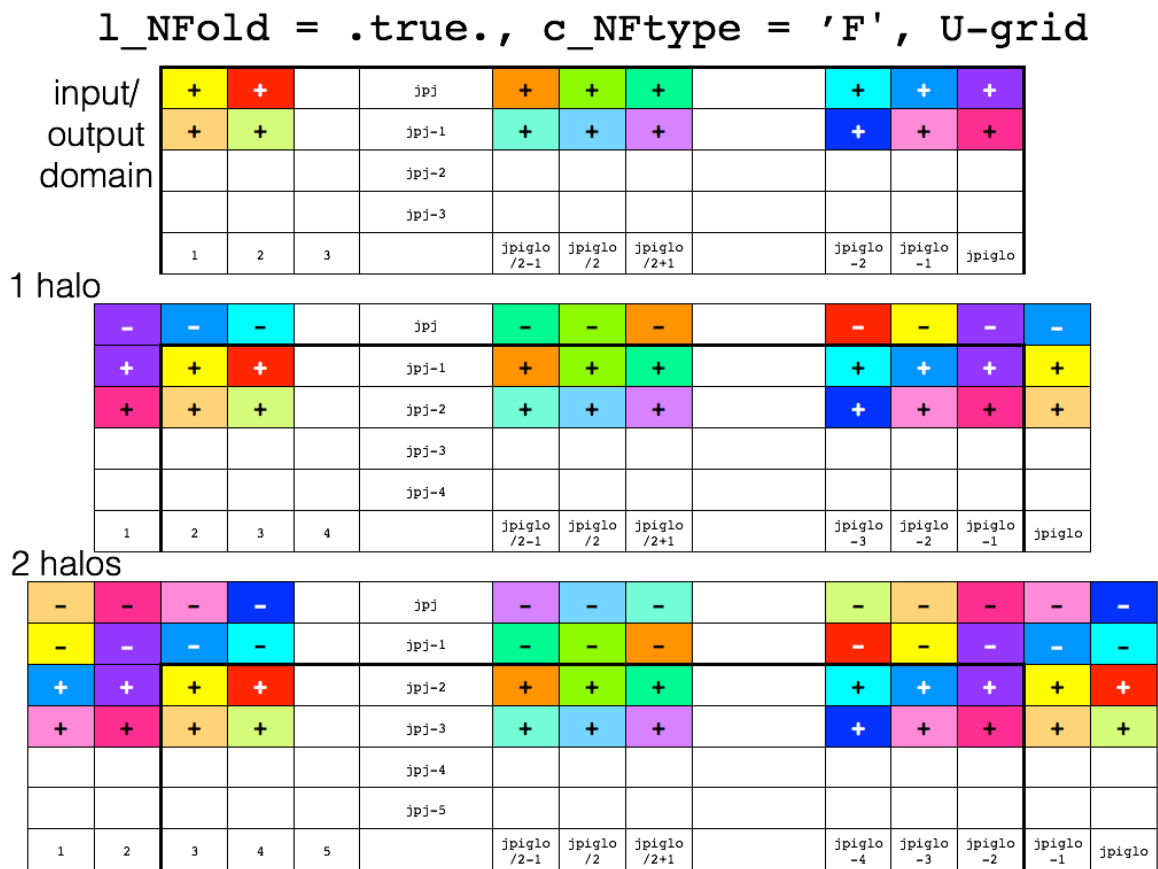


Figure E.6.: North fold boundary for the U grid, with a *F*-point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the `lbc_lnk` routine (found in `lbc_lnk.F90` module). Cells with the same color are at the same geographical location. Signs + or - are illustrating the change of signe on each side of the pivotal point.



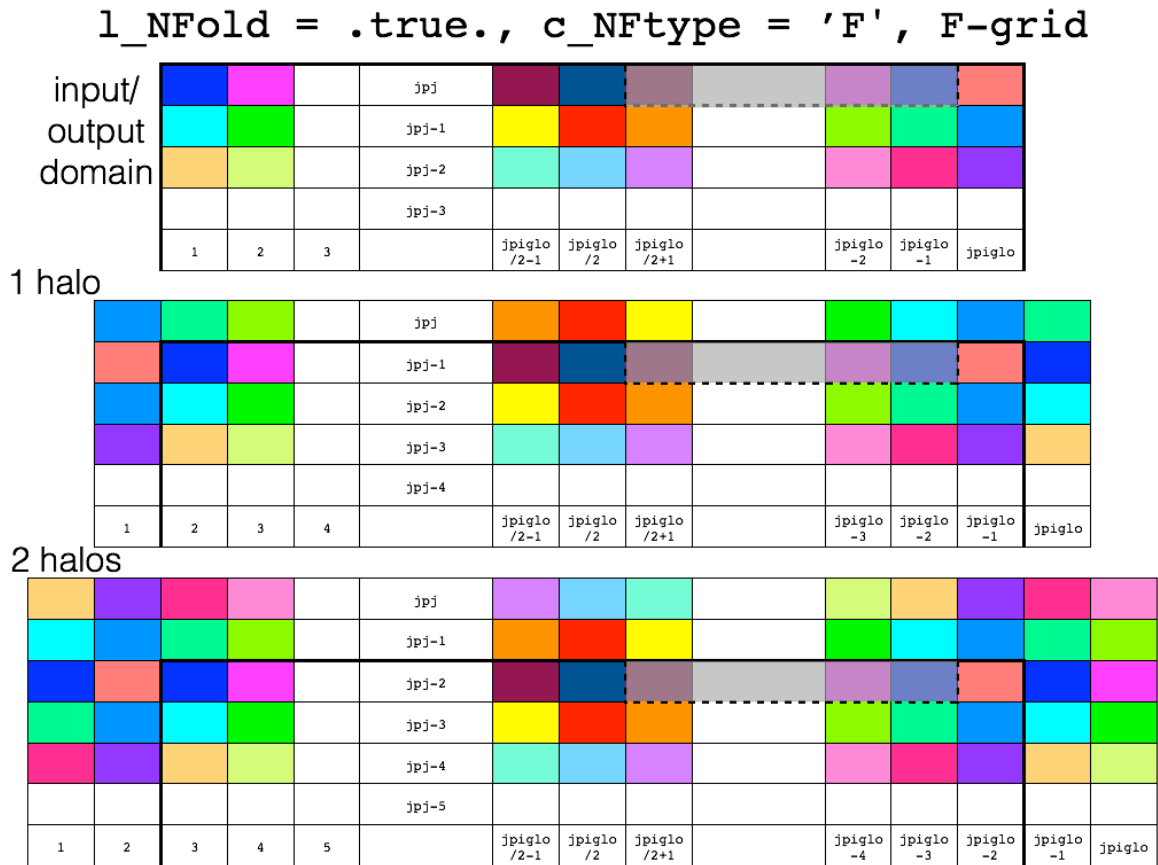


Figure E.8.: North fold boundary for the F grid, with a *F*-point pivot and cyclic east-west boundary condition. Cells in the halos, outside of the inner domain (thick black line), will be defined by using only the values in the inner domain when calling the `lbc_lnk` routine (found in `lbc_lnk.F90` module). Grey shading defines duplicated cells inside the inner domain that will also be refined (overwritten) by other inner domain values when calling the `lbc_lnk`. Cells with the same color are at the same geographical location.



# A brief guide to the DOMAINcfg tool

## Table of contents

- F.1. Choice of horizontal grid . . . . . 290
- F.2. Vertical grid . . . . . 291
  - F.2.1. Vertical reference coordinate . . . . . 291
  - F.2.2. Model bathymetry . . . . . 293
  - F.2.3. Choice of vertical grid . . . . . 294
- F.3. Ice shelf cavity definition . . . . . 297
- F.4. Closed sea mask definition (*domclo.F90*) . . . . . 297

## Changes record

Release	Author(s)	Modifications
5.0	<i>Katherine Hutchinson and Pierre Mathiot</i>	<i>update to NEMO 5.0</i>
4.2	<i>Pierre Mathiot</i>	<i>Add ice shelf and closed sea option description</i>
4.0	<i>Andrew Coward</i>	<i>Creation from materials removed from chapter 3 that are still relevant to the DOMAINcfg tool when setting up new domains</i>

```

-----
&namdom      !  space and time domain (bathymetry, mesh, timestep)
-----
ln_read_cfg = .false.  ! Read from a domain_cfg file
nn_bathy    = 1        ! = 0 compute analytically
                ! = 1 read the bathymetry file
                ! = 2 compute from external bathymetry
                ! = 3 compute from parent (if "key_agrif")

nn_interp   = 1        ! type of interpolation (nn_bathy =2)
cn_domcfg   = ' '      ! Name of the domain_cfg input file
cn_fcoord   = 'coordinates.nc' ! external coordinates file (jphgr_msh = 0)
cn_topo     = 'bathy_meter.nc' ! external topo file (nn_bathy =1/2)
cn_topolvl  = 'bathy_level.nc' ! external topo file (nn_bathy =1)
cn_fisfd    = 'isf_draft_meter.nc' ! external isf draft (nn_bathy =1 and ln_isfcav = .true.)
cn_bath     = 'Bathymetry' ! topo name in file (nn_bathy =1/2)
cn_bathlvl  = 'Bathy_level' ! lvl name in file (nn_bathy =1)
cn_visfd    = 'isf_draft' ! isf draft variable (nn_bathy =1 and ln_isfcav = .true.)
cn_lon      = 'nav_lon' ! lon name in file (nn_bathy =2)
cn_lat      = 'nav_lat' ! lat name in file (nn_bathy =2)
rn_scale    = 1.      ! multiplicative factor to account for possibly negative input bathymetry (agrif only)
rn_bathy    = 0.      ! value of the bathymetry. if (=0) bottom flat at jpkm1
nn_msh      = 0       ! create (=1) a mesh file or not (=0)
rn_hmin     = -3.     ! min depth of the ocean (>0) or min number of ocean level (<0)
rn_e3zps_min = 20.    ! partial step thickness is set larger than the minimum of
rn_e3zps_rat = 0.1    ! rn_e3zps_min and rn_e3zps_rat*3t, 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
ppglam0     = 0.0     ! longitude of first row and column T-point (jphgr_msh = 1)
ppghi0      = -35.0   ! latitude of first row 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        = 100.760928500000 ! Double tanh function parameters
ppkth2      = 48.029893720000 !
ppacr2      = 13.000000000000 !
/

```

namelist F.1.: &namdom\_domcfg

This appendix briefly describes some of the options available in the `DOMAINcfg` tool mentioned in [chapter 3](#).

Please note that there are plans to update and clean this tool and provide better step-by-step instructions. In the interim, for practical guidelines on the use of the tool, please see the [DOMAINcfg README](#).

The `DOMAINcfg` tool allows 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.

## F.1. Choice of horizontal grid

The user has three options available in defining a horizontal grid, which involve the namelist variable `jphgr_msh` ([namelist F.1](#)).

**`jphgr_msh = 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_msh = 1 to 4`** 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

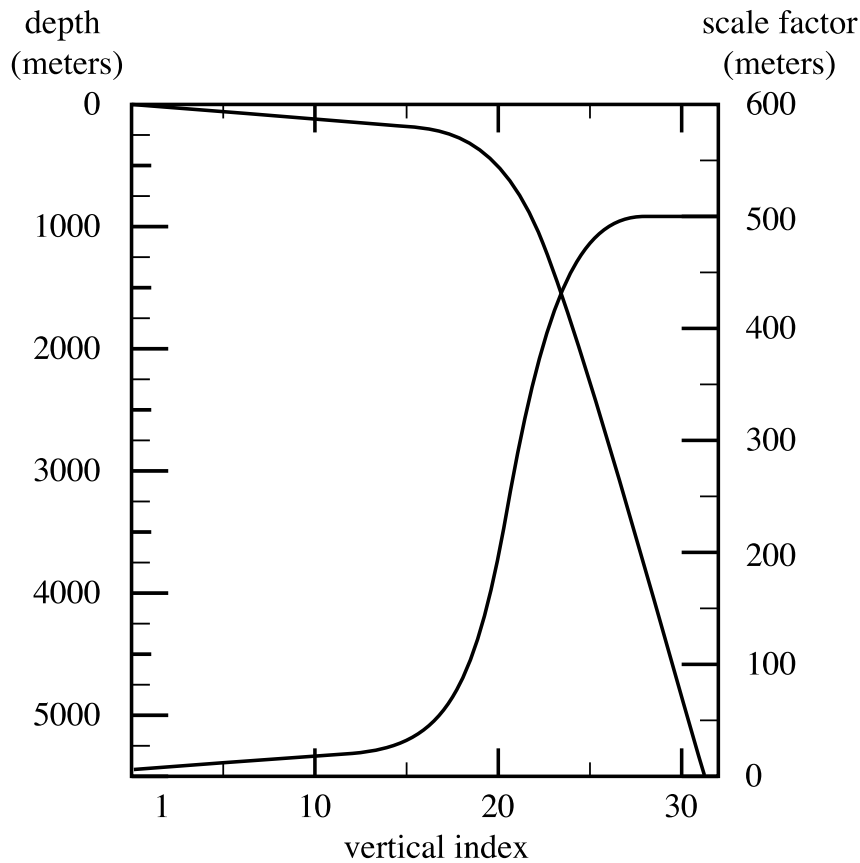


Figure F.1.: Default vertical mesh for ORCA2: 30 ocean levels (L30). Vertical level functions for (a) T-point depth and (b) the associated scale factor for the  $z$ -coordinate case.

`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_msh=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_msh=4`) avoids this anisotropy by refining the meridional scale factors in the same way as the zonal ones. In this case, meridional scale factors and latitudes are calculated analytically using the formulae appropriate for a Mercator projection, based on `ppe1_deg` which is a reference grid spacing at the equator (this applies even when the geographical equator is situated outside the model domain).

In these two cases (`jphgr_msh=1` or `4`), the grid position is defined by the longitude and latitude of the south-westernmost point (`ppglam0` and `ppgphi0`). Note that for the Mercator grid the user needs only to 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_msh=2` and `3`. The domain is either an  $f$ -plane (`jphgr_msh=2`, Coriolis factor is constant) or a beta-plane (`jphgr_msh=3`, the Coriolis factor is linear in the  $j$ -direction). The grid size is uniform in meters 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.

## F.2. Vertical grid

### F.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 [subsection F.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.

Since version 4.0, the default definition of cell depths for both  $w$ - and  $t$ -levels is done in a discrete sense where  $e_3^0 T = d_k(z_0^W)$  and  $e_3^0 W = d_k(z_0^T)$ , by setting `ln_e3_dep=.true.` in the (namelist F.1) (DOMAINcfg &namdom variant). (namely  $z_0^T(k) = 0.5(z_0^W(k) + z_0^W(k+1))$  and  $e_3^0 W = d_k(z_0^T)$ ). For backward compatibility with version 3.6, by setting `ln_e3_dep=.false.` the vertical location of  $w$ - and  $t$ -levels are defined using the former analytic expression of the depth  $z_0(k)$ , whose analytical derivative with respect to  $k$  provides the vertical scale factors. All the above operations are done through statement functions in the routine `domzgr.F90` (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` (maximum 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):

$$z_0(k) = h_{sur} - h_0 k - h_1 h_{cr} \log \left[ \cosh((k - h_{th})/h_{cr}) \right] \quad (\text{F.1})$$

$$e_3^0(k) = \left| -h_0 - h_1 \tanh \left[ (k - h_{th})/h_{cr} \right] \right| \quad (\text{F.2})$$

where  $k = 1$  to `jpk` for  $w$ -levels and  $k = 1$  to `jpk-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 F.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.

$$z_0(k) = h_{sur} - h_0 k - h_1 h_{cr} \log \left[ \cosh((k - h_{th})/h_{cr}) \right] - h_{21} h_{cr} \log \left[ \cosh((k - h_{2th})/h_{2cr}) \right] \quad (\text{F.3})$$

$$e_3^0(k) = \left| -h_0 - h_1 \tanh \left[ (k - h_{th})/h_{cr} \right] - h_{21} \tanh \left[ (k - h_{2th})/h_{2cr} \right] \right| \quad (\text{F.4})$$

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:

$$\begin{aligned} e_3^T(k) &= z_W(k+1) - z_W(k) \\ e_3^W(k) &= z_T(k) - z_T(k-1) \end{aligned} \quad (\text{F.5})$$

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:

$$\begin{aligned} e_3(1 + 1/2) &= 10. & z(1) &= 0. \\ e_3(jpk - 1/2) &= 500. & z(jpk) &= -5000. \end{aligned} \quad (\text{F.6})$$

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 F.5 have been determined such that equation F.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 F.1 and given in table F.1. Those values correspond to the parameters `ppsur`, `ppa0`, `ppa1`, `ppkth` in the DOMAINcfg variant of &namdom (namelist F.1).

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 17.1) namelist, and specifies instead the four following parameters:

- `ppacr = h_{cr}`: stretching factor (nondimensional). The larger `ppacr`, the smaller the stretching. Values from 3 to 10 are usual.
- `ppkth = h_{th}`: is approximately the model level at which maximum stretching occurs (nondimensional, usually of order 1/2 or 2/3 of `jpk`)

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 F.1.: Default vertical mesh in  $z$ -coordinate for 30 layers ORCA2 configuration as computed from equation F.5 using the coefficients given in equation F.6

- `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.

## F.2.2. Model bathymetry

The bathymetry can either be defined analytically by hand in `DOMAINcfg` variant of `domzgr.F90` in the `zgr_bat` subroutine (`nn_bathy` <= 0) with some predefined bathymetry or from a netcdf file provided by the user (`nn_bathy` > 0).

For the case `nn_bathy` <= 0 :

`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` : There are two predefined bathymetry. One for the OVERFLOW, and one for the DOME test cases (see user guide). If the configuration name (`cp_cfg` is different from DOME or OVERFLOW, the default is a flat bottom with a random noise. The user is strongly encourage to check `zgr_bat` to see if it fits exactly its need (many parameter hard coded for each cases).

For the case `nn_bathy` > 0 :

**nn\_bathy=1** : There is two cases. For  $z$  or  $\sigma$  coordinates ( **ln\_zps** or **ln\_sco** ), the user needs to provide a bathymetry and ice shelf draft depth variables ( **cn\_bath** and **cn\_visfd** ) in netcdf files ( **cn\_topo** and **cn\_fisfd** ). Ice shelf draft variable is optional as it depends if **ln\_isfcav** is activated or not. Each files need to provide data on each grid point of the 2D model grid. The depth convention for both variables is positive, in meters and the continent are defined by a 0 or negative value.

The bathymetry is usually built by interpolating a standard bathymetry product (*e.g.* ETOPO or GEBCO) onto the horizontal ocean mesh. For the case full cell ( **ln\_zco** ), the user need to provide a bottom level variables ( **cn\_bathlvl1** in a netcdf file ( **cn\_topo1vl1** ). Representation of the ice shel cavities is not implemented in this case.

In both cases, there is specific hand edits for the ORCA2 configuration ( **cp\_cfg='orca'** and **jp\_cfg=2** ) for Gibraltar and Bab el Mandeb straits.

**nn\_bathy>1** : ???

### F.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 (and shallowest level if **ln\_isfcav** ) at each  $(i, j)$  point using the bathymetry (and ice shelf draftif **ln\_isfcav** ), 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 F.2.1, *except* in the bottom layer. The thickness of the bottom layer is allowed to vary as a function of geographical location  $(\lambda, \varphi)$  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 **jpj-2** can have a thickness smaller than  $e_{3t}(jk)$ .

The model deepest layer (**jpj-1**) is allowed to have either a smaller or larger thickness than  $e_{3t}(jk)$ : the maximum thickness allowed is  $2 * e_{3t}(jk - 1)$ .

In case of ice shelf, partial step are allowed at the top interface using the same methodology.

This has to be kept in mind when specifying values in **&namdom** (DOMAINcfg variant ; **namelist F.1**), 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  $e_{3t}(jk - 1)$  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  $e_{3t}(jk) * \text{rn\_e3zps\_rat}$  (a fraction, usually 10%, of the default thickness  $e_{3t}(jk)$ ).

#### 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:

namelist F.2.: &amp;namzgr\_sco\_domcfg

```

-----
&namzgr_sco      !  s-coordinate or hybrid z-s-coordinate          (default: OFF)
-----
  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
                                !!!!!!! Envelop bathymetry
  rn_rmax        = 0.3       !  maximum cut-off r-value allowed (0<r_max<1)
                                !!!!!!! SH94 stretching coefficients (ln_s_sh94 = .true.)
  rn_theta       = 6.0       !  surface control parameter (0<theta<=20)
  rn_bb          = 0.8       !  stretching with SH94 s-sigma
                                !!!!!!! SF12 stretching coefficient (ln_s_sf12 = .true.)
  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
                                !  bottom cell depth (Zb) is a linear function of water depth Zb = H*a + b
  rn_zb_a        = 0.024    !  bathymetry scaling factor for calculating Zb
  rn_zb_b        = -0.2     !  offset for calculating Zb
                                !!!!!!! Other stretching (not SH94 or SF12) [also uses rn_theta above]
  rn_thetb       = 1.0     !  bottom control parameter (0<thetb<= 1)
/
    
```

$$z(k) = h(i, j) z_0(k)$$

$$e_3(k) = h(i, j) z'_0(k)$$

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 (figure 3.5) or an envelop bathymetry can be defined (figure 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:

$$C(s) = (1-b) \frac{\sinh(\theta s)}{\sinh(\theta)} + b \frac{\tanh[\theta(s + \frac{1}{2})] - \tanh(\frac{\theta}{2})}{2 \tanh(\frac{\theta}{2})}$$

where  $H_c$  is the critical depth (`rn_hc`) at which the coordinate transitions from pure  $\sigma$  to the stretched coordinate, and  $\theta$  (`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 F.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  $\gamma$  is defined such that:

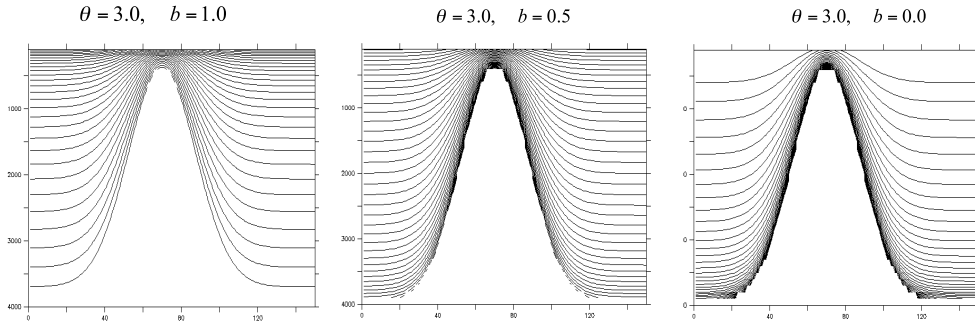


Figure F.2.: Examples of the stretching function applied to a seamount; from left to right: surface, surface and bottom, and bottom intensified resolutions

$$z = -\gamma h \quad \text{with} \quad 0 \leq \gamma \leq 1 \quad (\text{F.7})$$

The function is defined with respect to  $\sigma$ , 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  $\sigma$  that is solvable in  $A$  and  $B$  as a function of the user prescribed stretching parameter  $\alpha$  (`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.

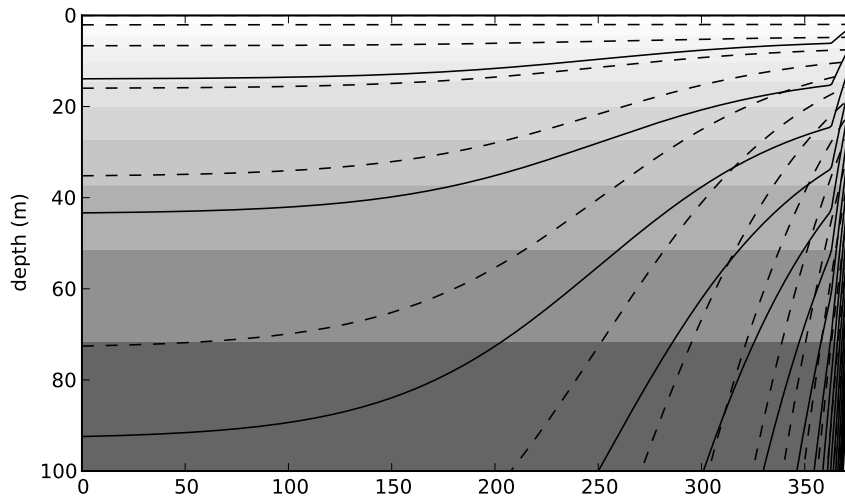


Figure F.3.: A comparison of the [Song and Haidvogel \(1994\)](#)  $S$ -coordinate (solid lines), a 50 level  $Z$ -coordinate (contoured surfaces) and the [Siddorn and Furner \(2013\)](#)  $S$ -coordinate (dashed lines) in the surface 100 m for an idealised bathymetry that goes from 50 m to 5500 m depth. 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.

## namelist F.3.: &amp;namzgr\_isf

```

-----
&namzgr_isf      !   isf cavity geometry definition                (default: OFF)
-----
!
  rn_isfdep_min   = 10.      ! minimum isf draft tickness (if lower, isf draft set to this value)
  rn_glh_min     = 1.e-3    ! minimum water column thickness to define the grounding line
  rn_isfhw_min   = 10       ! minimum water column thickness in the cavity once the grounding line defined.
  ln_isfchannel   = .false.  ! remove channel (based on 2d mask build from isfdraft-bathy)
  ln_isfconnect  = .false.  ! force connection under the ice shelf (based on 2d mask build from isfdraft-bathy)
  nn_kisfmax     = 999      ! limiter in level on the previous condition. (if change larger than this number, get back
↳ to value before we enforce the connection)
  rn_zisfmax     = 7000.    ! limiter in m      on the previous condition. (if change larger than this number, get back
↳ to value before we enforce the connection)
  ln_isfcheminey = .false.  ! close cheminey
  ln_isfsubgl    = .false.  ! remove subglacial lake created by the remapping process
  rn_isfsubgllon = 0.0      ! longitude of the seed to determine the open ocean
  rn_isfsubgllat = 0.0      ! latitude of the seed to determine the open ocean
/

```

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.

### F.3. Ice shelf cavity definition

If the under ice shelf seas are opened ( `ln_isfcav` ), the depth of the ice shelf/ocean interface has to be included in the `isfdraft_meter` file (Netcdf format). This file needs to include the `isf_draft` variable. A positive value will mean ice shelf/ocean interface below the reference 0m ssh. The exact shape of the ice shelf cavity (grounding line position and minimum thickness of the water column under an ice shelf, ...) can be specify in `&namzgr_isf` ([namelist F.3](#)) (`DOMAINcfg` only, [namelist F.3](#)). The details of each of these parameters is available in [subsection 8.1.6](#) in the ISF section ([section 8.1](#)).

### F.4. Closed sea mask definition ( `domclo.F90` )

The options available to define the closed seas with `DOMAINcfg` are listed in `&namclo`, while the control on how closed sea net fresh water input will be redistributed by *NEMO* is described in [subsection 3.2.3](#) and [section 16.2](#). The individual definition of each closed sea is managed by `sn_lake`. In this fields the user needs to define:

- The name of the closed sea (print output purposes).
- The seed location to define the area of the closed sea (if seed on land because not present in this configuration, this closed sea will be ignored).
- The seed location for the target area.
- The type of target area ('local', 'coast' or 'global'). See point 6 for definition of these cases.
- The type of redistribution scheme for the net fresh water flux over the closed sea (as a runoff in a target area, as emp in a target area, as emp globally). For the runoff case, if the net fwf is negative, it will be redistributed globally.
- The radius of the target area (not used for the 'global' case). The target defined by a 'local' target area of a radius of 100km, for example, correspond sto all the wet points within this radius. The coastal case will return only the coastal point within the specified radius.
- The target id. This id is used to group multiple lakes into the same river outflow (Great Lakes for example).

namelist F.4.: `&namclo`

```

-----
&namclo ! (closed sea : need ln_domclo = .true. in namcfg)
-----
  rn_lon_opnsea = -2.0      ! longitude seed of open ocean
  rn_lat_opnsea = -2.0      ! latitude seed of open ocean
  nn_closea = 8            ! number of closed seas (= 0; only the open_sea mask will be computed)
  !                          name ! lon_src ! lat_src ! lon_trg ! lat_trg ! river mouth area ! net evap/precip correction scheme !
↪ radius tgt ! id trg
  !                          ! (degree)! (degree)! (degree)! (degree)! local/coast/global ! (glo/rnf/emp) !
↪ (m) !
  ! North American lakes
  sn_lake(1) = 'superior' , -86.57 , 47.30 , -66.49 , 50.45 , 'local' , 'rnf' ,
↪ 550000.0 , 2
  sn_lake(2) = 'michigan' , -87.06 , 42.74 , -66.49 , 50.45 , 'local' , 'rnf' ,
↪ 550000.0 , 2
  sn_lake(3) = 'huron' , -82.51 , 44.74 , -66.49 , 50.45 , 'local' , 'rnf' ,
↪ 550000.0 , 2
  sn_lake(4) = 'erie' , -81.13 , 42.25 , -66.49 , 50.45 , 'local' , 'rnf' ,
↪ 550000.0 , 2
  sn_lake(5) = 'ontario' , -77.72 , 43.62 , -66.49 , 50.45 , 'local' , 'rnf' ,
↪ 550000.0 , 2
  ! African Lake
  sn_lake(6) = 'victoria' , 32.93 , -1.08 , 30.44 , 31.37 , 'coast' , 'emp' ,
↪ 100000.0 , 3
  ! Asian Lakes
  sn_lake(7) = 'caspien' , 50.0 , 44.0 , 0.0 , 0.0 , 'global' , 'glo' ,
↪ 0.0 , 1
  sn_lake(8) = 'aral' , 60.0 , 45.0 , 0.0 , 0.0 , 'global' , 'glo' ,
↪ 0.0 , 1
/

```

The closed sea module defines a number of masks in the *domain\_cfg* output:

- mask\_opensea:** a mask of the main ocean without all the closed seas closed. This mask is defined by a flood filling algorithm with an initial seed (localisation defined by `rn_lon_opnsea` and `rn_lat_opnsea`).
- mask\_csglo, mask\_csrnf, mask\_csemp:** a mask of all the closed seas defined in the namelist by `sn_lake` for each redistribution scheme. The total number of defined closed seas has to be defined in `nn_closea`.
- mask\_csgrglo, mask\_csgrrnf, mask\_csgrpemp:** a mask of all the closed seas and targets grouped by target id for each type of redistribution scheme.
- mask\_csundef:** a mask of all the closed sea not defined in `sn_lake`. This allows NEMO to mask them if needed or to inform the user of potential minor issues in its bathymetry.





## Table of contents

G.1. Introduction . . . . .	300
G.2. Overview and general conventions . . . . .	300
G.3. Architecture . . . . .	301
G.4. Style rules . . . . .	301
G.4.1. Argument list format . . . . .	301
G.4.2. Array syntax . . . . .	301
G.4.3. Case . . . . .	301
G.4.4. Comments . . . . .	302
G.4.5. Continuation lines . . . . .	302
G.4.6. Declaration of arguments and local variables . . . . .	302
G.4.7. F90 Standard . . . . .	302
G.4.8. Free-Form Source . . . . .	302
G.4.9. Indentation . . . . .	303
G.4.10. Loops . . . . .	303
G.4.11. Naming Conventions: files . . . . .	303
G.4.12. Naming Conventions: modules . . . . .	303
G.4.13. Naming Conventions: variables . . . . .	303
G.4.14. Operators . . . . .	303
G.4.15. Pre processor . . . . .	303
G.5. DO LOOP macros . . . . .	304
G.6. Content rules . . . . .	305
G.6.1. Configurations . . . . .	305
G.6.2. Constants . . . . .	305
G.6.3. Declaration for variables and constants . . . . .	306
G.6.4. Headers . . . . .	306
G.6.5. Interface blocks . . . . .	306
G.6.6. I/O Error Conditions . . . . .	306
G.6.7. PRINT - ASCII output files . . . . .	307
G.6.8. Precision . . . . .	307
G.6.9. Structures . . . . .	308
G.7. Packages coding rules . . . . .	308
G.7.1. Bounds checking . . . . .	308
G.7.2. Communication . . . . .	308
G.7.3. Error conditions . . . . .	308
G.7.4. Memory management . . . . .	308
G.7.5. Optimisation . . . . .	309
G.7.6. Package attribute: PRIVATE, PUBLIC, USE, ONLY . . . . .	309
G.7.7. Parallelism using MPI . . . . .	309
G.8. Features to be avoided . . . . .	309

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.

## G.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.

## G.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 SI<sup>3</sup> sea-ice model where it is in Kelvin. See `./src/OCE/DOM/phycst.F90` files for conversions.

---

\*UCAR conventions

## G.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
- BDY management of structured/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`).

## G.4. Style rules

### G.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 )
```

### G.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 * anothertwodarray(:,:)
```

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
```

### G.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.



### G.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

```

### G.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).

### G.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).<sup>†</sup>

### G.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<sub>tke</sub>, 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.

### G.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 FORTRANkeyword 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:

### G.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 >

### G.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.

<sup>†</sup>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	<b>m n</b> <i>but not nn__</i>	<b>a b e f g h</b> <b>o q to x</b> <i>but not fs rn__</i>	<b>l</b> <i>but not lp ld ll ln__</i>	<b>c</b> <i>but not cp cd cl cn__</i>	<b>d</b> <i>but not dp dd dl dn__</i>	<b>y</b> <i>but not yp yd yl</i>
dummy argument	<b>k</b> <i>but not kf</i>	<b>p</b> <i>but not pp pf</i>	<b>ld</b>	<b>cd</b>	<b>dd</b>	<b>yd</b>
local variable	<b>i</b>	<b>z</b>	<b>ll</b>	<b>cl</b>	<b>cd</b>	<b>yl</b>
loop control	<b>j</b> <i>but not jp</i>					
parameter	<b>jp</b>	<b>pp</b>	<b>lp</b>	<b>cp</b>	<b>dp</b>	<b>yp</b>
namelist	<b>nn__</b>	<b>rn__</b>	<b>ln__</b>	<b>cn__</b>	<b>dn__</b>	
CPP macro	<b>kf</b>	<b>sf</b>				

For example, the assimilation increments module name is *asminc.F90*, the CPP key is *key\_asminc* and the associated logical is *lk\_asminc*.

The following syntax:

```
!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.

## G.5. DO LOOP macros

Another aspect of the preprocessor is the use of macros to substitute code elements. In some cases these are used to reduce unnecessary array dimensions. A good example are the substitutions introduced by the `key_qco` key:

```
!if defined key_qco
# define e3t(i,j,k,t) (e3t_0(i,j,k)*(1._wp+r3t(i,j,t)*tmask(i,j,k)))
...
!elif defined key_linssh
# define e3t(i,j,k,t) e3t_0(i,j,k)
...
#endif
```

which are used to reduce 4-d arrays to a 3-d functional form or an invariant, 3-d array depending on other options. Such macros should be located in files with `_substitute.h90` endings to their names (e.g. `domzgr_substitute.h90`).

From 4.2, a more pervasive use of macros has been introduced in the form of DO LOOP macros. These macros have replaced standard nested, loops over the spatial dimensions. In particular:

```
DO jj = ....
  DO ji = ....
    .
    .
  END DO
END DO

OR

DO jk = ....
  DO jj = ...
    DO ji = ...
      .
      .
    END DO
  END DO
END DO
```

and white-space variants thereof.

The macro naming convention takes the form: `DO_2D( L, R, B, T )` where:

- `L` is the Left offset from the PE's inner domain
- `R` is the Right offset from the PE's inner domain
- `B` is the Bottom offset from the PE's inner domain
- `T` is the Top offset from the PE's inner domain

So, given an inner domain of `2,jpim1` and `2,jpjm1`, a typical example would replace:

```
DO jj = 2, jpj
  DO ji = 1, jpim1
    .
    .
  END DO
END DO
```

with:

```
DO_2D( 1, 0, 0, 1 )
  .
  .
END_2D
```

similar conventions apply to the 3D loops macros. `jk` loop limits are retained through macro arguments and are not restricted. This includes the possibility of strides for which an extra set of `DO_3DS` macros are defined.

The purpose of these macros is to enable support for extra-width halos. The width of the halo is determined by the value of the namelist parameter: `nn_hls`. Version 4.2 will work with either `nn_hls=1` or `nn_hls=2` but there is currently a performance penalty to using `nn_hls=2` since more development is needed before any benefits are realised. Code developers should consider whether or not loops need to be over:

- The inner domain only (e.g. `DO_2D( 0, 0, 0, 0 )`)
- The entire domain (e.g. `DO_2D( nn_hls, nn_hls, nn_hls, nn_hls )`)
- All but the outer halo (e.g. `DO_2D( nn_hls-1, nn_hls-1, nn_hls-1, nn_hls-1 )`)
- A mixture on different boundaries (e.g. `DO_2D( nn_hls, nn_hls-1, nn_hls, nn_hls-1 )`)

The correct use of these macros will eventually lead to performance gains through the removal of unnecessary computation and a reduction in communications.

## G.6. Content rules

### G.6.1. Configurations

The configuration defines the domain and the grid on which *NEMO* is running. From 4.2 onwards, all configuration-specific settings should be read from variables in, or attributes of, the domain configuration file (or set in `usrdef` supplied subroutines). See section "Spatial domain configuration" of the *NEMO* Manual for more details.

### G.6.2. Constants

Physical constants (e.g.  $\pi$ , 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.



### G.6.3. 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.

### G.6.4. 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`

### G.6.5. 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.

### G.6.6. 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.

‡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.

### G.6.7. PRINT - ASCII output files

Output listing and errors are directed to `numout` logical unit =6 and produces a file called `ocean.output`. Usually, this is produced by only the first ranked process in an MPP environment. This process will have the `lwp` logical variable set and this can be used to restrict output. For example: 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
```

At run-time, the user can use `sn_cfctl` options to have output from more processes in MPP.

### G.6.8. 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 corresponding 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
```

#### Use Mixed-Precision ( `key_single` )

The code working precision for `REAL` variables is declared in the `./src/OCE/par_kind.F90` module as follows:

```
# if defined key_single
INTEGER, PUBLIC, PARAMETER :: wp = sp !: working precision
# else
INTEGER, PUBLIC, PARAMETER :: wp = dp !: working precision
# endif
```

The default is `dp`, so that all the real variables declared `wp` are assigned to be double precision.

In version 4.2, the `key_single` is available to switch the precision of the `wp` parameter to single precision. This does not mean that the code will be fully single precision, which can lead to instabilities in the model. In fact, the declaration of a set of variables has been hardcoded to `dp` and the key `key_single` enables to run in mixed-precision. Variables that need to stay double precision, no matter the value of the working precision are declared as:

```
REAL(dp), DIMENSION(jpi,jpj,jpk) :: arr_name
```

This set of variables has been highlighted using the methodology described in [Tintó Prims et al. \(2019\)](#) and this mixed-precision version of `NEMO` is tested for the official configuration `ORCA2` compiled without the `ICE` module. Apart from changing the value of the `wp` parameter another macro will be activated by the usage of the `key_single`. Inside the `./src/OCE/single_precision_substitute.h90` two macros are defined:

```
#if defined key_single
# define CASTSP(x) REAL(x,sp)
# define CASTDP(x) REAL(x,dp)
#else
# define CASTSP(x) x
# define CASTDP(x) x
#endif
```

When the `key_single` is used at compilation time all the variable to which this macro is applied will be put inside a cast, to double or single precision depending on the macro used. This is done to ensure coherence at compilation time between variables and dummy arguments in some function and subroutine calls. When the `key_single` is omitted the cast is no longer needed and the macro evaluates to the variable itself.

## G.6.9. 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 ! output the tracer or not
END TYPE PTRACER

TYPE(PTRACER) , DIMENSION(jptra) :: tracer
```

Missing rule on structure name??

## G.7. Packages coding rules

### G.7.1. Bounds checking

*NEMO* is able to run when an array bounds checking option is enabled.

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.

### G.7.2. 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. <sup>||</sup>

### G.7.3. 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*.

### G.7.4. Memory management

The main action is to identify and declare which arrays are **PUBLIC** and which are **PRIVATE**.

As of version 3.3.1 of *NEMO*, the use of static arrays (size fixed at compile time) has been deprecated. All module arrays are now declared **ALLOCATABLE** and allocated in either the <module\_name>\_alloc() or <module\_name>\_init() routines. The success or otherwise of each **ALLOCATE** must be checked using the stat=<integer variable> optional argument.

In addition to arrays contained within modules, many routines in *NEMO* require local, "workspace" arrays to hold the intermediate results of calculations. These arrays are mostly declared in such a way as to be automatically allocated on the stack when the routine is called. Examples of an automatic arrays are:

```
SUBROUTINE sub(n)
  REAL(wp) :: za(n)
  REAL(wp), DIMENSION(jpi,jpj) :: zhdiv ! 2D workspace
  ...
END SUBROUTINE sub
```

Sometimes these local arrays are only required for specific options selected at run-time. Allocatable arrays should be used to avoid unnecessary use of stack storage in these cases. For example:

<sup>||</sup>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.

```

SUBROUTINE wzv(...)
...
REAL(wp), ALLOCATABLE, DIMENSION(:, :, :) :: zhdiv ! 3D workspace
...
IF( ln_vvl_ztilde .OR. ln_vvl_layer ) THEN
  ALLOCATE( zhdiv(jpi, jpj, jpk) )
...
  DEALLOCATE( zhdiv )
ELSEIF
...
END SUBROUTINE sub

```

### G.7.5. 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.

### G.7.6. 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.

### G.7.7. Parallelism using MPI

*NEMO* is written in order to be able to run on one processor, or on one or more using MPI. From 4.2, this is the default assumption but a non-MPI, single processor executable can be compiled by activating the cpp key: **key\_mpi\_off**.

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.

## G.8. 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. \*\*

---

\*\*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.

- **DATA** and **BLOCK DATA** (use initialisers)

## 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](#).
- Andreas, E., L. Mahrt, and D. Vickers, 2015: An improved bulk air-sea surface flux algorithm, including spray-mediated transfer. *Q.J.R. Meteorol. Soc.*, **141**, 642–654, [doi](#).
- Arakawa, A. and Y.-J. G. Hsu, 1990: Energy conserving and potential-enstrophy 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](#).
- Asay-Davis, X. S., S. L. Cornford, G. Durand, B. K. Galton-Fenzi, R. M. Gladstone, G. H. Gudmundsson, T. Hattermann, D. M. Holland, D. Holland, P. R. Holland, D. F. Martin, P. Mathiot, F. Pattyn, and H. Seroussi, 2016: Experimental design for three interrelated marine ice sheet and ocean model intercomparison projects: Mismip v. 3 (mismip+), isomip v. 2 (isomip+) and misomip v. 1 (misomip1). *Geoscientific Model Development*, **9** (7), 2471–2497, [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](#).
- 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](#).
- Bigg, G. R., M. R. Wadley, D. P. Stevens, and J. A. Johnson, 1997: Modelling the dynamics and thermodynamics of icebergs. *Cold Regions Science and Technology*, **26** (2), 113–135, [doi](#).
- Bignami, F., S. Marullo, R. Santoleri, and M. E. Schiano, 1995: Longwave radiation budget in the mediterranean sea. *Journal of Geophysical Research*, **100**, 2501–2514, [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](#).
- 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 air–sea 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](#).
- Bruciaferri, D., G. Shapiro, and F. Wobus, 2018: A multi-envelope vertical coordinate system for numerical ocean modelling. *Ocean Dynamics*, **68** (10), 1239–1258, [doi](#).
- Burchard, H., 2002: Energy-conserving discretisation of turbulent shear and buoyancy production. *Ocean Modelling*, **4** (3–4), 347–361, [doi](#).
- Burgard, C., N. C. Jourdain, R. Reese, A. Jenkins, and P. Mathiot, 2022: An assessment of basal melt parameterisations for antarctic ice shelves. *The Cryosphere*, **16** (12), 4931–4975, [doi](#), [URL](#).
- 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](#).
- Castellari, S., N. Pinardi, and K. Leaman, 1998: A model study of air-sea interactions in the mediterranean sea. *Journal of Marine System*, **18**, 89–114, [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](#).
- Cheng, Y., V. M. Canuto, and A. M. Howard, 2002: An improved model for the turbulent pbl. *Journal of the Atmospheric Sciences*, **59**, 1550–1565, [doi](#), [URL](#).
- Ciliberti, S. A., E. Jansen, G. Coppini, E. Peneva, D. Azevedo, S. Causio, L. Stefanizzi, S. Creti, R. Lecci, L. Lima, M. Ilicak, N. Pinardi, and A. Palazov, 2022: The black sea physics analysis and forecasting system within the framework of the copernicus marine service. *Journal of Marine Science and Engineering*, **10** (1), [doi](#).
- Couvelard, X., F. Lemarié, G. Samson, J.-L. Redelsperger, F. Ardhuin, R. Benshila, and G. Madec, 2020: Development of a two-way-coupled ocean–wave model: assessment on a global nemo(v3.6)–ww3(v6.02) coupled configuration. *Geosci. Model Dev*, **13**, 3067–3090, [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](#).
- Cuxart, J., P. Bougeault, and J. L. Redelsperger, 2000: A turbulence scheme allowing for mesoscale and large-eddy simulations. *Quarterly Journal of the Royal Meteorological Society*, **126**, 1–30, [doi](#), [URL](#).
- 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](#).



- de Lavergne, C., S. Rathore, G. Madec, J.-B. Sallée, C. Ethé, A. Nasser, B. Millet, and M. Vancoppenolle, 2024: Effects of improved tidal mixing in nemo one-degree global ocean model. *ESS Open Archive*, [doi](#).
- de Lavergne, C., C. Vic, G. Madec, F. Roquet, A. F. Waterhouse, C. B. Whalen, Y. Cuypers, P. Bouruet-Aubertot, B. Ferron, and T. Hibiya, 2020: A parameterization of local and remote tidal mixing. *Journal of Advances in Modeling Earth Systems*, **12** (5), e2020MS002065, [doi](#).
- Deardorff, J. W., 1980: Stratocumulus-capped mixed layers derived from a three-dimensional model. *Boundary-Layer Meteorology 1980 18:4*, **18**, 495–527, [doi](#), [URL](#).
- 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](#).
- Demange, J., L. Debreu, P. Marchesiello, F. Lemarié, E. Blayo, and C. Eldred, 2019: Stability analysis of split-explicit free surface ocean models: Implication of the depth-independent barotropic mode approximation. *Journal of Computational Physics*, **398**, 108875, [doi](#).
- 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](#).
- Ducouso, N., F. Lemarié, L. Debreu, and G. Madec, 2024: Stability and accuracy of runge–kutta-based split-explicit time-stepping algorithms for free-surface, quasi-eulerian vertical coordinates ocean models. *To be submitted to JAMES*.
- 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. Hershbach, 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](#).
- El-Tahan, M., S. Venkatesh, and H. El-Tahan, 1987: Validation and quantitative assessment of the deterioration mechanisms of arctic icebergs. *Journal of Offshore Mechanics and Arctic Engineering*, **109** (1), 102–108, [doi](#), [URL](#).
- 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. S. Godfrey, G. A. Wick, J. B. Edson, and G. S. Young, 1996: Cool-skin and warm-layer effects on sea surface temperature. *Journal of Geophysical Research: Oceans*, **101** (C1), 1295–1308, [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](#).
- Favier, L., N. C. Jourdain, A. Jenkins, N. Merino, G. Durand, O. Gagliardini, F. Gillet-Chaulet, and P. Mathiot, 2019: Assessment of sub-shelf melting parameterisations using the ocean–ice-sheet coupled model nemo(v3.6)–elmer/ice(v8.3). *Geoscientific Model Development*, **12** (6), 2255–2283, [doi](#), [URL](#).
- Fedorov, K. N., 1988: Layer thicknesses and effective diffusivities in “diffusive” thermohaline convection in the ocean. *Small-Scale Turbulence and Mixing in the Ocean*, Nihoul, J. and B. Jamart, Eds., Elsevier, Elsevier Oceanography Series, Vol. 46, 471–479, [doi](#), [URL](#).
- Ferreira, D., J. Marshall, and P. Heimbach, 2005: Estimating eddy stresses by fitting dynamics to observations using a residual-mean ocean circulation model and its adjoint. *Journal of Physical Oceanography*, **35** (10), 1891–1910, [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., G. Danabasoglu, R. Ferrari, S. Griffies, R. Hallberg, M. Holland, M. Maltrud, S. Peacock, and B. Samuels, 2011: Parameterization of mixed layer eddies. iii: Implementation and impact in global ocean climate simulations. *Ocean Modelling*, **39** (1), 61–78, [doi](#).
- 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](#).
- Giordani, H., R. Bourdallé-Badie, and G. Madec, 2020: An eddy-diffusivity mass-flux parameterization for modeling oceanic convection. *Journal of Advances in Modeling Earth Systems*, **12** (9), e2020MS002 078, [doi](#).
- Gladstone, R. M., G. R. Bigg, and K. W. Nicholls, 2001: Iceberg trajectory modeling and meltwater injection in the southern ocean. *Journal of Geophysical Research: Oceans*, **106** (C9), 19 903–19 915, [doi](#).
- Graham, F. S. and T. J. McDougall, 2013: Quantifying the nonconservative production of conservative temperature, potential temperature, and entropy. *Journal of Physical Oceanography*, **43** (5), 838–862, [doi](#).
- Grant, A. L. M., 2001: Cloud-base fluxes in the cumulus-capped boundary layer. *Quarterly Journal of the Royal Meteorological Society*, **127** (572), 407–421, [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](#).
- Grosfeld, K., R. Gerdes, and J. Determann, 1997: Thermohaline circulation and interaction between ice shelf cavities and the adjacent open ocean. *Journal of Geophysical Research: Oceans*, **102** (C7), 15 595–15 610, [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](#).
- Hellerman, S. and M. Rosenstein, 1983: Normal monthly wind stress over the world ocean with error estimates. *Journal of Physical Oceanography*, **13**, 1093–1104, [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](#).
- 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](#).
- Irrmann, G., S. Masson, E. Maisonnave, D. Guibert, and E. Raffin, 2022: Improving ocean modeling software nemo 4.0 benchmarking and communication efficiency. *Geoscientific Model Development*, **15** (4), 1567–1582, [doi](#), [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., H. H. Hellmer, and D. M. Holland, 2001: The role of meltwater advection in the formulation of conservative boundary conditions at an ice–ocean interface. *Journal of Physical Oceanography*, **31** (1), 285–296, [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](#).
- Josey, S. A., S. Gulev, and L. Yu, 2013: Exchanges through the ocean surface. *Ocean Circulation and Climate*, 115–140, [doi](#).
- Jourdain, N. C., P. Mathiot, N. Merino, G. Durand, J. Le Sommer, P. Spence, P. Dutrieux, and G. Madec, 2017: Ocean circulation and sea-ice thinning induced by melting ice shelves in the amundsen sea. *Journal of Geophysical Research: Oceans*, **122** (3), 2550–2573, [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](#).
- Kevlahan, N.-R., T. Dubos, and M. Aechtner, 2015: Adaptive wavelet simulation of global ocean dynamics using a new brinkman volume penalization. *Geoscientific Model Development*, **8**, 3891–3909, [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.
- Kondo, J., 1975: Air-sea bulk transfer coefficients in diabatic conditions. *Boundary-Layer Meteorology*, **9**, 91–112, [doi](#).
- Kraus, E. B., 1990: Diapycnal mixing. *Climate-Ocean Interaction*, Springer, 269–293.
- Kraus, E. B. and J. A. Businger, 1996: Atmosphere-ocean interaction. *Quarterly Journal of the Royal Meteorological Society*, **122** (529), 324–325, [doi](#).
- Kraus, E. B. and J. S. Turner, 1967: A one-dimensional model of the seasonal thermocline ii. the general theory and its consequences. *Tellus*, **19** (1), 98–106, [doi](#).
- Lac, C., J. P. Chaboureaud, V. Masson, J. P. Pinty, P. Tulet, J. Escobar, M. Leriche, C. Barthe, B. Aouizerats, C. Augros, P. Aumond, F. Auguste, P. Bechtold, S. Berthet, S. Bielli, F. Bosseur, O. Caumont, J. M. Cohard, J. Colin, F. Couvreur, J. Cuxart, G. Delautier, T. Dauhut, V. Ducrocq, J. B. Filippi, D. Gazen, O. Geoffroy, F. Gheusi, R. Honnert, J. P. Lafore, C. L. Brossier, Q. Libois, T. Lunet, C. Mari, T. Maric, P. Mascart, M. Mog’è, G. Molini’è, O. Nuissier, F. Pantiillon, P. Peyrill’è, J. Pergaud, E. Perraud, J. Pianezze, J. L. Redelsperger, D. Ricard, E. Richard, S. Riette, Q. Rodier, R. Schoetter, L. Seyfried, J. Stein, K. Suhre, M. Taufour, O. Thouron, S. Turner, A. Verrelle, B. Vi’è, F. Visentin, V. Vionnet, and P. Wautelet, 2018: Overview of the meso-nh model version 5.4 and its applications. *Geoscientific Model Development*, **11**, 1929–1969, [doi](#).
- 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](#).
- , 2009: The global climatology of an interannually varying air-sea flux data set. *Climate Dynamics*, **33** (2-3), 341–364, [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 biharmonic isoneutral mixing operators in ocean models. *Ocean Modelling*, **52–53**, 9–35, [doi](#).
- Lemarié, F., G. Samson, J. L. Redelsperger, H. Giordani, T. Brivoal, and G. Madec, 2021: A simplified atmospheric boundary layer model for an improved representation of air-sea interactions in eddy oceanic models: Implementation and first evaluation in nemo (4.0). *Geoscientific Model Development*, **14**, 543–572, [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](#).
- Levier, 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](#).
- Loset, S., 1993: Thermal energy conservation in icebergs and tracking by temperature. *Journal of Geophysical Research: Oceans*, **98** (C6), 10001–10012, [doi](#).
- Lott, F., G. Madec, and J. Verron, 1990: Topographic experiments in an ocean general circulation model. *Ocean Modelling*, **88**, 1–4.
- Love, A. E. H., 1909: The yielding of the earth to disturbing forces. *Proceedings of the Royal Society of London*, **82** (551), 73–88, [doi](#).
- Lucazeau, F., 2019: Analysis and mapping of an updated terrestrial heat flow data set. *Geochemistry, Geophysics, Geosystems*, **20** (8), 4001–4024, [doi](#).
- 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](#).
- Madec, G., F. Lemarié, S. Téchené, J. Chanut, N. Ducouso, M. Bell, A. Coward, L. Debreu, S. Masson, and D. Storkey, 2024: Implementation of a runge-kutta-based time-stepping algorithm in the nemo ocean model: formulation, robustness and efficiency. *To be submitted to JAMES*.

- Mak, J., A. Avdis, T. W. David, H. S. Lee, Y. Na, Y. Wang, and F. E. Yan, 2022a: On constraining the mesoscale eddy energy dissipation time-scale. *Journal of Advances in Modeling Earth Systems*, **14** (11), e2022MS003 223, [doi](#).
- Mak, J., D. P. Marshall, G. Madec, and J. R. Maddison, 2022b: Acute sensitivity of global ocean circulation and heat content to eddy energy dissipation timescale. *Geophysical Research Letters*, **49** (8), e2021GL097 259, [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 freshwater 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](#).
- Merino, N., J. Le Sommer, G. Durand, N. C. Jourdain, G. Madec, P. Mathiot, and J. Tournadre, 2016: Antarctic icebergs melt over the southern ocean: Climatology and impact on sea ice. *Ocean Modelling*, **104**, 99 – 110, [doi](#), [URL](#).
- 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](#).
- Oddo, P., M. Adani, N. Pinardi, C. Fratianni, M. Tonani, , and D. Pettenuzzo, 2009: A nested atlantic-mediterranean sea general circulation model for operational forecasting. *Ocean Sciences*, **5**, 1–13, [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](#).
- Payne, R. E., 1972: Albedo of the sea surface. *Journal of the Atmospheric Sciences*, **29**, 959–970, [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  $1/4^\circ$  global ocean simulations. *Ocean Science*, **3** (4), 509–524, [doi](#).
- Pergaud, J., V. Masson, S. Malardel, and F. Couvreur, 2009: Quantifying the nonconservative production of conservative temperature, potential temperature, and entropy. *Boundary-Layer Meteorology*, **132** (1), 838–862, [doi](#).
- Qiao, F., Y. Yuan, T. Ezer, C. Xia, Y. Yang, X. Lu, and Z. Song, 2010: A three-dimensional surface wave–ocean 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](#).
- Reed, R. K., 1977: On estimating insolation over the ocean. *Journal of Physical Oceanography*, **1**, 874–971, [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., Inter-science 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](#).
- Shapiro, G., M. Luneva, J. Pickering, and D. Storkey, 2013: The effect of various vertical discretization schemes and horizontal diffusion parameterization on the performance of a 3-d ocean model: the black sea case study. *Ocean Science*, **9** (2), 377–390, [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, 2003: A method for computing horizontal pressure-gradient force in an oceanic model with a nonaligned vertical coordinate. *Journal of Geophysical Research: Oceans*, **108** (C3), 3090, [doi](#).
- , 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](#).
- Soares, P. M. M., P. M. A. Miranda, A. P. Siebesma, and J. Teixeira, 2004: An eddy-diffusivity/mass-flux parametrization for dry and shallow cumulus convection. *Quarterly Journal of the Royal Meteorological Society*, **130**, 3365–3383, [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](#).
- 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](#).
- Sverdrup, H. U., M. W. Johnson, and R. H. Fleming, 1942: *The Oceans, Their Physics, Chemistry, and General Biology*. Prentice-Hall, New York, 1087 pp.
- Takaya, Y., J.-R. Bidlot, A. C. M. Beljaars, and P. A. E. M. Janssen, 2010: Refinements to a prognostic scheme of skin sea surface temperature. *Journal of Geophysical Research*, **115** (C6), C06 009, [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](#).
- Tintó Prims, O., M. C. Acosta, A. M. Moore, M. Castrillo, K. Serradell, A. Cortés, and F. J. Doblas-Reyes, 2019: How to use mixed precision in ocean models: exploring a potential reduction of numerical precision in nemo 4.0 and roms 3.6. *Geoscientific Model Development*, **12**, 3135–3148, [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](#).
- Tsamados, M., 2014: Impact of variable atmospheric and oceanic form drag on simulations of arctic sea ice. *Journal of Physical Oceanography*, **44** (5), 1329–1353, [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](#).
- Wicker, L. J. and W. C. Skamarock, 2002: Time-splitting methods for elastic models using forward time schemes. *Mon. Weather Rev.*, **130** (8), 2088–2097, [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](#).
- Wise, A., J. Harle, D. Bruciaferri, E. O’Dea, and J. Polton, 2021: The effect of vertical coordinates on the accuracy of a shelf sea model. *Ocean Modelling*, 101935, [doi](#).
- Zalesak, S. T., 1979: Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, **31** (3), 335–362, [doi](#).
- Zeng, X. and A. Beljaars, 2005: A prognostic scheme of sea surface skin temperature for modeling and data assimilation. *Geophysical Research Letters*, **32** (14), [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

<code>&amp;nam_asminc</code> .....	216, 217	<code>&amp;namsbc_sas</code> .....	85
<code>&amp;nam_diadct</code> .....	185	<code>&amp;namsbc_ssr</code> .....	103
<code>&amp;nam_tide</code> .....	96, 97, 134	<code>&amp;namsbc_wave</code> .....	100
<code>&amp;nambbc</code> .....	71	<code>&amp;namsto</code> .....	221
<code>&amp;namdbl</code> .....	71	<code>&amp;namtra_adv</code> .....	60
<code>&amp;nambdy</code> .....	127, 130, 132, 133	<code>&amp;namtra_dmp</code> .....	73, 197
<code>&amp;nambdy_dta</code> .....	127, 130	<code>&amp;namtra_eiv</code> .....	142
<code>&amp;nambdy_tide</code> .....	134	<code>&amp;namtra_ldf</code> .....	63, 136, 266
<code>&amp;namberg</code> .....	115	<code>&amp;namtra_mle</code> .....	144
<code>&amp;namcfg</code> ...	32, 37, 124, 182, 231, 232, 235, 292	<code>&amp;namtra_qsr</code> .....	68
<code>&amp;namctl</code> .....	192, 227	<code>&amp;namtrc_trd</code> .....	183, 185
<code>&amp;namdom</code> .....	37, 57, 73, 190	<code>&amp;namtrd</code> .....	183
<code>&amp;namdrg</code> .....	162, 165	<code>&amp;namtsd</code> .....	23, 73
<code>&amp;namdrg_bot</code> .....	162	<code>&amp;namusr_def</code> .....	235
<code>&amp;namdrg_top</code> .....	162	<code>&amp;namtke</code> .....	150
<code>&amp;namdyn_adv</code> .....	45, 48	<code>&amp;namzdf</code> 24, 54, 66, 146, 159–161, 167	
<code>&amp;namdyn_hpg</code> .....	50	<code>&amp;namzdf_gls</code> .....	153
<code>&amp;namdyn_ldf</code> .....	52, 136	<code>&amp;namzdf_ric</code> .....	147
<code>&amp;namdyn_spg</code> .....	39, 52	<code>&amp;namzdf_tke</code> .....	151
<code>&amp;namdyn_vor</code> .....	45	<code>&amp;namzgr_isf</code> .....	114, 297
<code>&amp;nameos</code> .....	74		
<code>&amp;namisf</code> .....	108		
<code>&amp;namlbc</code> .....	122		
<code>&amp;namldf_eke</code> .....	143		
<code>&amp;nammpp</code> .....	124–126		
<code>&amp;namnc4</code> .....	183		
<code>&amp;namobs</code> .....	194, 195, 209		
<code>&amp;namobs_dta</code> .	194, 195, 197, 202		
<code>&amp;namrun</code> .....	23, 181		
<code>&amp;namsao</code> .....	209		
<code>&amp;namsbc</code> .	79, 92, 94, 98–100, 102, 103, 105		
<code>&amp;namsbc_abl</code> .....	93		
<code>&amp;namsbc_apr</code> .....	94		
<code>&amp;namsbc_blk</code> .....	91, 93		
<code>&amp;namsbc_cpl</code> .....	94, 100		
<code>&amp;namsbc_flx</code> .....	85		
<code>&amp;namsbc_fwb</code> .....	106		
<code>&amp;namsbc_rnf</code> .....	98		

key_agrif .....	234	key_qco .....	33–35, 51, 52, 57, 59, 60, 304
key_asminc .....	216	key_RK3 .....	18
key_diahth .....	190	key_si3 .....	105
key_diainstant .....	170	key_single .....	307
key_isf .....	51	key_top .....	94
key_linssh... 33–35, 45, 52, 57, 59, 60, 67, 68, 74, 189		key_trdmxl_trc .....	185
key_mpi2 .....	125	key_trdtrc .....	185
key_mpi_off .....	125, 309	key_vco .....	34
key_nosignedzero .....	226	key_vco_1d .....	34, 35
key_oa3mct_v1v2 .....	94	key_vco_1d3d .....	34, 35
key_oasis3 .....	94	key_vco_3d .....	34, 35
		key_xios .....	26, 170, 185
		key_xios3 .....	170

<i>closea.F90</i> .....	X, 224, 226, 227	<i>iom.F90</i> .....	85, 173, 178	<i>tide_mod.F90</i> .....	96
<i>daymod.F90</i> .....	85, 189	<i>isfcav.F90</i> .....	108	<i>traadv.F90</i> .....	59, 76
<i>dia25h.F90</i> .....	191	<i>isfcavgam.F90</i> .....	111	<i>traadv_cen.F90</i> .....	61
<i>diar5.F90</i> .....	189, 191	<i>isfcavmlt.F90</i> .....	108	<i>traadv_fct.F90</i> .....	61
<i>diadct.F90</i> .....	185	<i>isfcpl.F90</i> .....	113	<i>traadv_mus.F90</i> .....	62
<i>diadetide.F90</i> .....	191, 192	<i>isfdiags.F90</i> .....	112	<i>traadv_qck.F90</i> .....	63
<i>diahth.F90</i> .....	190	<i>isfload.F90</i> .....	114	<i>traadv_ubs.F90</i> .....	62, 63
<i>diamlr.F90</i> .....	189	<i>isfpar.F90</i> .....	112	<i>traatf.F90</i> .....	73, 74
<i>diaobs.F90</i> .....	202	<i>isfparmlt.F90</i> .....	112	<i>trabbc.F90</i> .....	69
<i>diaptr.F90</i> .....	191	<i>isftbl.F90</i> .....	111	<i>trabbl.F90</i> .....	67, 71
<i>diawri.F90</i> .....	85	<i>istate.F90</i> .....	23, 29	<i>tradmp.F90</i> .....	73
<i>divhor.F90</i> .....	44, 98, 111	<i>lbcnk.F90</i> .....	123–125, 281–288	<i>traisf.F90</i> .....	59, 66, 111
<i>dom_oce.F90</i> .....	189	<i>lbcnfd.F90</i> .....	124	<i>traldf.F90</i> .....	63
<i>domclo.F90</i> .....	297	<i>ldfc1d_c2d.F90</i> .....	140	<i>traldf_iso.F90</i> .....	65, 66
<i>domhgr.F90</i> .....	33, 290	<i>ldfdyn.F90</i> .....	139	<i>traldf_lev.F90</i> .....	65
<i>domisf.F90</i> .....	114	<i>ldfslp.F90</i> .....	63, 136	<i>traldf_triad.F90</i> .....	65
<i>domzgr.F90</i> .....	33, 292, 293	<i>ldftra.F90</i> .....	139	<i>tramle.F90</i> .....	59
<i>dtastd.F90</i> .....	127	<i>lib_fortran.F90</i> .....	226, 227	<i>tranpc.F90</i> .....	59
<i>dtatsd.F90</i> .....	23, 29	<i>lib_mpp.F90</i> .....	124	<i>traqsr.F90</i> .....	67–69, 80
<i>dynadv.F90</i> .....	49	<i>mppini.F90</i> .....	127	<i>trasbc.F90</i> .....	66, 67, 80, 98
<i>dynadv_up3.F90</i> .....	50	<i>nemogcm.F90</i> .....	85, 194, 209	<i>trazdf.F90</i> .....	18, 25, 66, 74, 146
<i>dynatf.F90</i> .....	57	<i>phycst.F90</i> .....	59, 74	<i>trc_oce.F90</i> .....	69
<i>dynhpg.F90</i> .....	50	<i>sbcabl.F90</i> .....	91	<i>trddyn.F90</i> .....	183
<i>dynkeg.F90</i> .....	48	<i>sbcapr.F90</i> .....	94	<i>trdtra.F90</i> .....	183
<i>dynldf.F90</i> .....	44, 52	<i>sbcblk.F90</i> .....	86	<i>trdtrc.F90</i> .....	183
<i>dynspg.F90</i> .....	52, 96	<i>sbcctl.F90</i> .....	94	<i>userdef_istate.F90</i> .....	23
<i>dynspg_ts.F90</i> .....	II, 38, 40	<i>sbcscy.F90</i> .....	102	<i>usrdef_fmash.F90</i> .....	224
<i>dynvor.F90</i> .....	45, 48	<i>sbcflx.F90</i> .....	85	<i>usrdef_hgr.F90</i> .....	32, 33, 224, 291
<i>dynzad.F90</i> .....	48	<i>sbcfwb.F90</i> .....	105	<i>usrdef_hrg.F90</i> .....	235
<i>dynzdf.F90</i> .....	18, 25, 44, 54, 57, 80, 146, 163–166	<i>sbcice_if.F90</i> .....	105	<i>usrdef_nam.F90</i> .....	32, 124
<i>eosbn2.F90</i> .....	59, 74–76, 162	<i>sbcmod.F90</i> .....	67, 80, 85	<i>usrdef_zgr.F90</i> .....	32, 34, 56, 57, 235
<i>fldread.F90</i> .....	73, 81, 130	<i>sbcnfn.F90</i> .....	97	<i>wet_dry.F90</i> .....	56
<i>geo2ocean.F90</i> .....	103	<i>sbcssm.F90</i> .....	85	<i>zdfddm.F90</i> .....	161
<i>icbclv.F90</i> .....	115, 117	<i>sbcssr.F90</i> .....	96, 103	<i>zdfdrg.F90</i> .....	162–165
<i>icbdia.F90</i> .....	118	<i>sbcwave.F90</i> .....	99	<i>zdfgls.F90</i> .....	146
<i>icbdyn.F90</i> .....	117	<i>sshwzv.F90</i> .....	39, 44, 45	<i>zdfiwm.F90</i> .....	166
<i>icbini.F90</i> .....	117	<i>step.F90</i> .....	85, 194	<i>zdfmal.F90</i> .....	155
<i>icblbc.F90</i> .....	118	<i>stopar.F90</i> .....	221	<i>zdfosm.F90</i> .....	146
<i>icbthm.F90</i> .....	117	<i>stopts.F90</i> .....	221	<i>zdfphy.F90</i> .....	146
<i>icbtrj.F90</i> .....	119	<i>storng.F90</i> .....	221	<i>zdfric.F90</i> .....	146
		<i>stp2d.F90</i> .....	39, 52	<i>zdftrc.F90</i> .....	146, 166
		<i>stpctl.F90</i> .....	85		

## Namelist parameters

)	85	ln_dyn_trd	185
cn_altbiasfile	197	ln_dynadv_cen2	49
cn_bath	294	ln_dynadv_up3	49, 136
cn_bathlvl	294	ln_dynadv_vec	57
cn_bathy	114	ln_dynhpg_zco	50
cn_coords_file	128	ln_dynldf_bilap	54
cn_dir	81, 91, 130	ln_dynldf_blp	136
cn_domcfg	32, 124, 224	ln_dynldf_hor	53, 136, 137
cn_domcfg_out	37	ln_dynldf_iso	53
cn_draft	114	ln_dynldf_lap	53, 136
cn_dyn3d	131	ln_dynldf_OFF	50, 136
cn_fisfd	294	ln_dynspg_exp	21, 39, 52
cn_gammablack	111	ln_dynspg_ts	21, 39, 52
cn_gridsearchfile	195	ln_dynvor_eeen	46, 252, 258
cn_groupname	195	ln_dynvor_ene	46, 252
cn_ice	128	ln_dynvor_ens	46, 257
cn_isfcav_mlt	108, 110, 112	ln_dynvor_enT	49
cn_isfdir	112	ln_dynvor_mix	46
cn_isfload	114	ln_dynvor_msk	45
cn_isfpar_mlt	112	ln_e3_dep	292
cn_mask_file	132	ln_ECMWF	88, 89, 100
cn_obsbiasfile_varname	197	ln_enabled	195
cn_obsbiasfiles	197	ln_eos80	75
cn_obsfiles	195, 196	ln_flg	85, 102, 103
cn_obstypes	195–197, 202	ln_fp_indegs	196
cn_ocerst_in	23	ln_full_vel	130
cn_ocerst_out	23	ln_geos_winds	93
cn_resto	73	ln_glo_trd	183
cn_storst_in	221	ln_grid_global	195
cn_storst_out	221	ln_grid_search_lookup	195
cn_tide_load	97	ln_hpg_djc	51
cn_topo	294	ln_hpg_isf	51, 114
cn_topolvl	294	ln_hpg_prj	51
cn_tra	128, 131	ln_hpg_sco	51, 114
cn_type_to_biascorrect	197	ln_hpgls_frc	93
cn_u2d	128	ln_humi_dpt	91
cn_u3d	128	ln_humi_rlh	91
cn_visfd	294	ln_humi_sph	91
cp_cfg	293, 294	ln_hvolg_var	106
ctypebody	132	ln_icb_grd	117
filtide	134	ln_ice_embd	80
jp_cfg	294	ln_icebergs	115
jperio	293	ln_ignmis	196
jphgr_msh	290, 291	ln_iscpl	113
jpni	126, 127	ln_isf	41, 44, 108
jpnj	126, 127	ln_isfcav	51, 114, 164, 292, 294, 297
l_sasread	85	ln_isfcav_mlt	108
ldbletanh	292		
lk_asminc	.AND. ln_sshinc		
	.AND. ln_asmiau	42,	
	44		
ln_3d_uve	85		
ln_abl	91, 92		
ln_all_at_all	196		
ln_altbias	197		
ln_apr_dyn	42, 94		
ln_apr_obc	96		
ln_asmdin	216		
ln_asmiau	216		
ln_bdy	128		
ln_bdytide_2ddta	134		
ln_bench	235		
ln_bergdia	118		
ln_bern_srfc	101		
ln_bkl	92		
ln_blk	86, 102, 103		
ln_boost	163, 164		
ln_botmix_triad	66, 266, 272, 273, 277		
ln_bound_reject	196		
ln_breivikFV_2016	100, 101		
ln_bt_auto	39		
ln_cdgw	100		
ln_cfmeta	181		
ln_charn	100		
ln_closea	224, 226		
ln_COARE	88		
ln_COARE_3p0	89		
ln_convmix	154		
ln_coords_file	128		
ln_cpl	100		
ln_ctl	119		
ln_Cx_ice_cst	90		
ln_Cx_ice_frm	90		
ln_Cx_ice_LG15	90		
ln_Cx_ice_LU12	90		
ln_dia_osm	155		
ln_diacfl	192		
ln_diadct	185		
ln_diaobs	194		
ln_dm2dc	102		
ln_drg_OFF	163		
ln_drgimp	164, 165		
ln_dyn3d_dmp	131		
ln_dyn_mx1	185		

ln_isfchannel	115	ln_seos	74, 75, 137	ln_wd_dl_ramp	56	
ln_isfcheminey	115	ln_shuman	76	ln_write_cfg	37	
ln_isfconnect	115	ln_sigcrit	297	ln_xios_read	23, 182	
ln_isfcpl	108	ln_skin_cs	88, 89	ln_zad_Aimp	24–26, 45, 55, 62	
ln_isfcpl_cons	41, 114	ln_skin_wl	88, 89	ln_zco	50, 64, 65, 294	
ln_isfpar_mlt	108, 112	ln_ssr	103	ln_zdfcst	147	
ln_isfsubgl	115	ln_ssr_bnd	104	ln_zdfddm	66, 161	
ln_KE_trd	185	ln_stcor	101	ln_zdfevd	160, 161	
ln_kpprimix	154	ln_sto_eos	221	ln_zdfgl	152, 153, 161	
ln_lc	151	ln_sto_hpg	221	ln_zdfiwm	166	
ln_ldfeiv	142	ln_sto_ldf	221	ln_zdfmfc	160	
ln_length_lim	154	ln_sto_pstar	221	ln_zdfnpc	159, 160	
ln_lin	163	ln_sto_trc	221	ln_zdfosm	154, 161	
ln_linssh	51, 297	ln_sto_trd	221	ln_zdftric	147	
ln_listonly	126	ln_stshear	102	ln_zdfswm	168	
ln_loglayer	164	ln_surf	195	ln_zdftke	148, 161	
ln_M2016	117, 118	ln_tair_pot	91	ln_zinterp	130	
ln_mask_csundef	226	ln_tauoc	102	ln_zps	51, 65, 294	
ln_mask_file	132	ln_taw	102	nb_bdy	128	
ln_maskcs	224, 226	ln_teos10	74	nbdybeg	132	
ln_meshmask	37	ln_teos80	74	nbdyend	132	
ln_mevr	166	ln_tide	42, 96, 134	nbdyind	132	
ln_MFS	88, 100	ln_tide_pot	42, 96	niaufn	216	
ln_mixcpl	84	ln_tide_ramp	97	nn_1dint	196	
ln_mldw	147	ln_time_mean_bkg	196	nn_2dint	196, 203	
ln_mle	143, 144	ln_tra_dmp	131	nn_aei_ijk_t	142	
ln_mskland	127	ln_tra_mx1	59, 185	nn_ahm_ijk_t	139–141	
ln_mus_ups	62	ln_tra_trd	59, 185	nn_aht_ijk_t	139–141	
ln_mxhsw	150	ln_traadv_cen	61	nn_amxl	94	
ln_mx10	150	ln_traadv_fct	25, 61	nn_atfp	57	
ln_nc4zip	183	ln_traadv_mus	25, 62	nn_avb	146	
ln_NCAR	88, 100	ln_traadv_OFF	59	nn_ave	155	
ln_nea	195	ln_traadv_qck	63	nn_bathy	293, 294	
ln_night	194, 196	ln_traadv_ubs	62	nn_bbl_adv	72	
ln_nnogather	124	ln_trabbc	69, 71, 146	nn_bbl_ldf	71, 272	
ln_non_lin	164	ln_trabbl	71, 272	nn_bc_bot	153	
ln_obsbias	197	ln_tradmp	73, 197	nn_bc_surf	150, 153	
ln_output_clim	197	ln_traldf_blp	64, 65, 136	nn_btflt	40, 41	
ln_passive_mode	118	ln_traldf_eiv	266	nn_cen_h	61	
ln_PE_trd	185	ln_traldf_gdia	277	nn_cen_v	61	
ln_phioc	102, 150	ln_traldf_hor	64, 65, 136, 137	nn_chldta	69	
ln_prec_met	91	ln_traldf_iso	64, 65, 136, 266	nn_clo	153	
ln_prof	195	ln_traldf_lap	64, 65, 136	nn_closea	298	
ln_qsr_2bd	69	ln_traldf_lev	64, 65, 136	nn_comm	125, 126	
ln_qsr_5bd	69	ln_traldf_msc	63, 66, 266	nn_components	85	
ln_qsr_bio	69	ln_traldf_OFF	64, 136	nn_dct	185	
ln_qsr_rgb	69	ln_traldf_triad	64–66, 136–138, 142, 143, 266, 275	nn_dctwri	186	
ln_read_cfg	32, 124, 235	ln_tranpc	159	nn_divdmp	217	
ln_read_load	97	ln_traqsr	67, 68, 80	nn_drown	113	
ln_restart	23	ln_triad_iso	66, 266, 273, 274, 277	nn_dyn2d_dta	130	
ln_rnf	41, 44, 67, 98	ln_tsd_dmp	73	nn_dynkeg	45	
ln_rnf_depth	98	ln_tsd_init	23, 73	nn_dynldf_typ	53	
ln_rnf_depth_ini	98	ln_tsdiff	66, 166	nn_e	39–41	
ln_rnf_icb	98	ln_use_osm_la	155	nn_ediff	148	
ln_rnf_mouth	99, 105	ln_usr	79	nn_ediss	148	
ln_rnf_sal	98	ln_vol	133	nn_eeen_e3f	47	
ln_rnf_tem	98	ln_vor_trd	185	nn_eice	151, 152	
ln_rst_list	23	ln_vortex_force	101	nn_eosflt	221	
ln_rstseed	221	ln_wave	76, 99, 150, 168	nn_eos_ord	221	
ln_rststo	221	ln_wave_AND	ln_bern_srfc	42	nn_etau	152
ln_s_SF12	295	ln_wave_test	100	nn_euler	23	
ln_s_sf12	295	ln_wd_dl	55	nn_evdm	160	
ln_s_sh94	295	ln_wd_dl_bc	56	nn_fct_h	61, 62	
ln_scal_load	97			nn_fct_imp	62	
ln_sco	51, 64, 65, 137, 294			nn_fct_v	61, 62	
ln_sdw	41, 76, 100, 150, 168			nn_frm	90	

nn_fsbc	79, 80, 82, 86, 173, 179	rn_alpha	296	rn_ice_age	130
nn_fwb	106	rn_atfp	19, 74	rn_ice_apnd	130
nn_fwb_voltype	106	rn_avglamscl	196	rn_ice_hlid	130
nn_geoflx	71	rn_avgphiscl	196	rn_ice_hpnd	130
nn_GYRE	235	rn_avm0	146	rn_ice_sal	130
nn_havtb	146	rn_avmri	147	rn_ice_tem	130
nn_hls	IX, 124, 125, 235	rn_avt0	146	rn_initial_mass	115
nn_htau	152	rn_avt_rnf	99	rn_initial_thickness	115
nn_hvolg_mth	106	rn_b0	75	rn_isfdep_min	114, 115
nn_ice	105	rn_bb	295	rn_isfhw_min	114
nn_ice_dta	128	rn_bits_erosion_fraction	118	rn_isfload_s	114
nn_ice_embd	42	rn_bt_alpha	40	rn_isfload_t	114
nn_mle	144	rn_bt_cmax	39	rn_isfpar_bg03_gt0	112
nn_msh	127	rn_Cd0	163, 164	rn_ke0	164
nn_msshc	197	rn_Cd_ia	90	rn_kisfmax	115
nn_mx1	149, 150	rn_Cdmax	164	rn_lambda1	75
nn_nit000	117	rn_Ce_ia	90	rn_lambda2	75
nn_npc	159	rn_Cf_ia	90	rn_lat_opnsea	298
nn_obsgroups	195	rn_Cf_io	90	rn_lc	151
nn_osm_wave	154, 155	rn_Ch_ia	90	rn_Ld	140
nn_pdl	148	rn_charn	153	rn_ldyn_max	93
nn_profdavtypes	194, 196	rn_clim_galp	154	rn_ldyn_min	93
nn_ref_apr	96	rn_Cr_ia	90	rn_Le	142
nn_ric	147	rn_Cr_io	90	rn_lon_opnsea	298
nn_rimwidth	130–132	rn_crban	153	rn_LoW_ratio	117
nn_rnf_depth_file	98	rn-Cs_ia	90	rn_ltra_max	93
nn_rstctl	23	rn-Cs_io	90	rn_ltra_min	93
nn_sample_rate	119	rn_csmc	141	rn_Lv	140
nn_secdebug	186	rn_deds	104	rn_m_la	154
nn_ssr	104	rn_dep_max	98	rn_mass_scaling	115
nn_ssr_ice	104	rn_dif_conv	154	rn_maxfac	142
nn_sstr	103	rn_difri	154	rn_mdtcorr	197
nn_stab_func	153	rn_distribution	117	rn_mdtcutoff	197
nn_sto_eos	221	rn_dobsend	195	rn_minfac	142
nn_stock	23	rn_dobsini	195	rn_mldmax	147
nn_stocklist	23	rn_dqdt	103	rn_mldmin	147
nn_test_icebergs	117	rn_Dt	25–27, 191	rn_mu1	75
nn_tide_var	96	rn_e3zps_min	294	rn_mu2	75
nn_tra_dta	128, 130	rn_e3zps_rat	294	rn_mx10	150
nn_trd	183	rn_ebb	148, 150	rn_nu	75
nn_u2d_dta	128	rn_efac	91	rn_osm_dstokes	154, 155
nn_u3d_dta	128	rnEFR	152	rn_osm_hbl0	155
nn_ubs_v	63	rn_eke_dis	143	rn_par	69
nn_verbose_level	119	rn_eke_lap	143	rn_pfac	91
nn_verbose_write	119	rn_eke_min	143	rn_radar_snow_penetr	197
nn_volctl	133	rn_ekmfc	147	rn_rdt	179
nn_write	23	rn_emin	149	rn_rho_bergs	117
nn_wxios	23, 182	rn_emin0	149	rn_riinfy	154
nn_zdmp	73	rn_eos_lim	221	rn_rmax	295
ppa0	292	rn_eos_stdxy	221	rn_rnf_max	98
ppa1	292	rn_eos_stdz	221	rn_sbot_max	295
ppacr	292, 293	rn_eos_tcor	221	rn_sbot_min	295
ppdzmin	293	rn_evd	160	rn_scal_load	97
ppe1_deg	291	rn_fwb0	106	rn_SFmax	143
ppe1_m	291	rn_gambbl	72	rn_SFmin	143
ppe2_deg	291	rn_gammas0	111	rn_shlat	122, 123
ppe2_m	291	rn_gammat0	111	rn_si0	68, 69
ppglam0	291	rn_geoflx_cst	71	rn_slpmax	66, 138, 266
ppgphi0	291	rn_geom	143	rn_speed_limit	118
pphmax	292–294	rn_glhv_min	114	rn_ssr_bnd	104
ppkth	292, 293	rn_gridsearchres	195	rn_sw_triad	66, 266
ppsur	292	rn_hc	295	rn_test_box	117
rn_a0	75	rn_hrnf	99	rn_theta	295
rn_abs	68, 69	rn_htbl	108, 111	rn_tide_gamma	97
rn_ahtbbl	71	rn_hvolg_amp	106	rn_tide_ramp_dt	97
rn_alp	147	rn_hvolg_trd	106	rn_time_dmp	131

rn_time_dmp_out .....	131	sn_dep_rnf .....	98, 99	sn_ssh .....	85
rn_time_mean_period .....	196	sn_e3t .....	85	sn_sss .....	104
rn_Uc0 .....	163	sn_frq .....	85	sn_sst .....	103
rn_Ud .....	140	sn_hpgi .....	93	sn_t_rnf .....	98
rn_Ue .....	142	sn_hpgj .....	93	sn_tair .....	91, 93
rn_Uv .....	140	sn_humi .....	91, 93	sn_tdif .....	91
rn_vfac .....	91	sn_i_rnf .....	98	sn_tem .....	23, 73, 85
rn_wdmin1 .....	56	sn_icb .....	117	sn_tide_cnames .....	96, 97, 134
rn_wtmix .....	147	sn_isfcav_fwf .....	108	sn_ustp .....	85
rn_wvmix .....	147	sn_isfpar_fwf .....	112	sn_vsp .....	85
rn_z0 .....	164	sn_isfpar_Leff .....	112	sn_wndi .....	91, 93
rn_zb_a .....	296	sn_isfpar_zmax .....	112	sn_wndj .....	91, 93
rn_zb_b .....	296	sn_isfpar_zmin .....	112	zdrag_ia .....	90
rn_zisfmax .....	115	sn_lake .....	297, 298	zdrag_ia_floe .....	90
rn_zqt .....	88, 91	sn_prec .....	91	zdrag_ia_rdg .....	90
rn_zs .....	296	sn_qlw .....	91	zdrag_ia_skin .....	90
rn_zu .....	88, 91	sn_qsr .....	91, 103	zdrag_io .....	90
sn_apr .....	94	sn_rnf .....	98	zdrag_io_floe .....	90
sn_apref .....	96	sn_s_rnf .....	98	zdrag_io_rdg .....	90
sn_cfctl .....	228	sn_sal .....	23, 73, 85	zdrag_io_skin .....	90
sn_cnf .....	99	sn_snow .....	91		



## Fortran subroutines

C	
ctl_stop .....	170
D	
div_hor .....	217
domqco_r3c .....	34
dyn_asm_inc .....	216
F	
fld_read .....	81, 83
H	
hgr_read .....	290
I	
init_locglo .....	127
iom_put .....	170, 171, 173
iom_set_axis_attr .....	173
iom_set_domain_attr .....	173
iom_set_field_attr .....	173
isf_hdiv .....	111
L	
lbc_lnk .....	123, 125, 281–288
ldf_eiv_trp .....	142
ldf_mle_trp .....	144
ldf_slp_init .....	137
S	
sbc_blk_init .....	81
sbc_rnf_div .....	98
sbcbk_algo_ecmwf .....	100
set_xmlatt .....	178
ssh_asm_div .....	217
sto_par .....	221
sto_par_init .....	221
sto_rst_write .....	221
stp_ctl .....	85
stp_mlf .....	216
stp_rk3_stg3 .....	216
T	
tra_adv .....	142, 144
tra_adv_trp .....	76
tra_asm_inc .....	216
tra_dmp_init .....	73
tra_ldf_blp .....	65
tra_ldf_lap .....	65
tra_sbc .....	217
traqsr .....	67
trc_oce_rgb .....	69
trd_dyn .....	183
trd_tra .....	183
trd_trc .....	183
U	
usr_def_istate .....	23
usrdef_nam .....	124
Z	
zdfcke .....	102
zgr_bat .....	293