



Nucleus for European Modelling of the Ocean



Sea Ice modelling Integrated Initiative (SI³)

The NEMO sea ice engine

Version 4.0.1 - October, 2019

Yevgeny Aksenov
Ed Blockley
Matthieu Chevallier
Danny Feltham
Thierry Fichefet
Gilles Garric
Paul Holland
Dorotea Iovino
Gurvan Madec
François Massonnet
Jeff Ridley
Clément Rousset
David Salas
David Schroeder
Steffen Tietsche
Martin Vancoppenolle

Abstract

“Sea Ice Modelling Integrated Initiative” (*SI³*) is the sea ice engine of the *NEMO* ocean model (“Nucleus for European Modelling of the Ocean”). It is intended to be a flexible tool for studying the sea ice dynamics and thermodynamics, brine inclusions and subgrid-scale thickness variations (“white ocean”), as well as its interactions with the other components of the Earth climate system over a wide range of space and time scales. *SI³* is interfaced with the *NEMO* ocean engine, and, via the *OASIS* coupler, with several atmospheric general circulation models. It also supports two-way grid embedding by means of the *AGRIF* software.

Designed for global to regional applications up to 10 km of effective resolution, *SI³* is a curvilinear grid, finite-difference implementation of the classical AIDJEX model (Arctic Ice Dynamics Joint EXperiment), combining the conservation of momentum for viscous-plastic continuum, energy and salt-conserving halo-thermodynamics, an explicit representation of subgrid-scale ice thickness variations, snow and melt ponds. An option to switch back to the *single-category* (or *2-level*) framework provides a cheap sea ice modelling solution.



Community

Ocean

Model

Disclaimer






Like all components of the modelling framework, the SI3 core engine is developed under the **CECILL** license, which is a French adaptation of the GNU GPL (General Public License). Anyone may use it freely for research purposes, and is encouraged to communicate back to the development team its own developments and improvements.

The model and the present document have been made available as a service to the community. We cannot certify that the code and its manual are free of errors. Bugs are inevitable and some have undoubtedly survived the testing phase. Users are encouraged to bring them to our attention.

The authors assume no responsibility for problems, errors, or incorrect usage of *NEMO*.

Other resources

Additional information can be found on:

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

Citation

Reference for papers and other publications is as follows:

“Sea Ice modelling Integrated Initiative (SI³) – The NEMO sea ice engine”, *Scientific Notes of Climate Modelling Center*, **31** — ISSN 1288-1619, Institut Pierre-Simon Laplace (IPSL), doi:10.5281/zenodo.1471689

List of Figures

1.1. Representation of the ice pack, using multiple categories with specific ice concentration ($a_l, l = 1, 2, \dots, L$), thickness (h_l^i), snow depth (h_l^s), vertical temperature and salinity profiles (T_{kl}^i, S_{kl}^*) and a single ice velocity vector (u).	3
1.2. Representation of the relation between real thickness profiles and the ice thickness distribution function $g(h)$	5
1.3. Elliptical yield curve used in the VP rheologies, drawn in the space of the principal components of the stress tensor (σ_1 and σ_2).	7
1.4. Thermal properties of sea ice vs temperature for different bulk salinities: brine fraction, specific enthalpy, thermal conductivity, and effective specific heat.	9
2.1. Schematic representation of time stepping in SI^3 , assuming $nn_fsc = 5$	12
2.2. Vertical grid of the model, used to resolve vertical temperature and salinity profiles	13
2.3. Boundaries of the model ice thickness categories (m) for varying number of categories and prescribed mean thickness (\bar{h}). The formerly used \tanh formulation is also depicted.	14
6.1. Partitioning of solar radiation in the snow-ice system, as represented in SI^3	24
6.2. Albedo correction $\Delta\alpha$ as a function of overcast sky (diffuse light) albedo α_{os} , from field observations (Grenfell, 2004, their Table 3) (squares) and 2nd-order fit (Eq. 6.3). Red squares represent the irrelevant data points excluded from the fit. For indication, the amplitude of the correction used in the ocean component is also depicted (blue circle).	25
6.3. Example albedo dependencies on ice thickness, snow depth and pond depth, as parameterized in SI^3	26
6.4. Framing solar radiation transfer through sea ice into the atmosphere-ice-ocean context.	27
7.1. Scheme of the estimate of the heat budget of the first ocean level.	30

List of Namelists

List of Tables

1.1. Thermodynamic constants of the model.	4
1.2. LIM global variables.	4
1.3. Intensive variables of practical use.	5
1.4. Main model parameters.	7

[July 2018]

SI^3 is the result of the recommendation of the Sea Ice Working Group (SIWG) to reduce duplication and better use development resources. SI^3 merges the capabilities of the 3 formerly sea ice models used in *NEMO* (CICE, GELATO and LIM). The 3 in SI^3 refers either to the three formerly used sea ice models and linkages between 3 different media (ocean-ice-snow). The model would be pronounced as “SI cube” for short (or “Sea Ice cubed” for slightly longer), otherwise it can be spelt “SI three” in situations where the superscript could be problematic.

In order to handle all the subsequent required subjective choices, we applied the following guidelines or principles:

- Sea ice is frozen seawater that is in tight interaction with the underlying ocean. This close connexion suggests that the sea ice and ocean model components must be as consistent as possible. In practice, this is materialized by the close match between LIM and *NEMO*, in terms of numerical choices, regarding the grid (Arakawa C-type) and the numerical discretization (finite differences with *NEMO* scale factors).
- It is useful to be able to either prescribe the atmospheric state or to use an atmospheric model. For consistency and simplicity of the code, we choose to use formulations as close as possible in both cases.
- Different resolutions and time steps can be used. There are parameters that depend on such choices. We thrived to achieve a resolution and time-step independent code, by imposing a priori scaling on the resolution / time step dependence of such parameters.
- Energy, mass and salt must be conserved as much as possible.

This manual is organised as follows.

List of chapters...

There are no more CPP keys in the code.

Namelists and output management follow *NEMO* guidelines.

Changes between releases.

The list of people that should be acknowledged is too long, but a great number of people or more exactly a number of great people contributed to the code and should be gratefully acknowledged. As for today, the SIWG members are (*July 2018*).

- Yevgeny Aksenov (NOCS, Southampton, UK)
- Ed Blockley (Met Office, Exeter, UK, co-chair)
- Matthieu Chevallier (CNRM-GAME, Météo France, Toulouse)
- Danny Feltham (CPOM, Reading, UK)
- Thierry Fichefet (UCL, Louvain-la-Neuve, Belgium)
- Gilles Garric (Mercator-Océan, Toulouse, France)

-
- Paul Holland (BAS, Cambridge, UK)
 - Dorotea Iovino (CMCC, Bologna, Italy)
 - Gurvan Madec (LOCEAN, CNRS, Paris, France)
 - François Massonnet (UCL, Louvain-la-Neuve, Belgium)
 - Jeff Ridley (Met Office, Exeter, UK)
 - Clément Rousset (LOCEAN, CNRS, Paris, France)
 - David Salas (CNRM-GAME, Météo France, Toulouse)
 - David Schroeder (CPOM, Reading, UK)
 - Steffen Tietsche (ECMWF, Reading, UK)
 - Martin Vancoppenolle (LOCEAN, CNRS, Paris, France, co-chair)

1. Model Basics	2
1.1. Rationale and assumptions	3
1.1.1. Scales, thermodynamics and dynamics	3
1.1.2. Subgrid scale variations	3
1.1.3. Thermodynamic formulation	4
1.1.4. Dynamic formulation	4
1.2. Thickness distribution framework	5
1.3. Governing equations	6
1.4. Ice Dynamics	7
1.5. Ice thermodynamics	8
1.5.1. Transport in thickness space	8
1.5.2. Thermodynamic source and sink terms	8
2. Time, space and thickness space domain	11
2.1. Time domain	12
2.2. Spatial domain	12
2.3. Thickness space domain	13
3. Ice dynamics	15
4. Ice transport	17
4.1. Second order moments conserving (Prather 1986) scheme (<i>ln_adv_Pra</i>)	18
4.2. 5 th order flux-corrected transport scheme (UM5)	18
5. Ridging and rafting	19
5.1. Dynamical inputs	20
5.2. The two deformation modes: ridging and rafting	21
5.3. Participation functions	21
5.4. Transfer functions	21
5.5. Ridging shift	22
5.6. Mechanical redistribution for other global ice variables	22
6. Radiative transfer	23
6.1. Solar radiation partitioning in the snow-ice system	24
6.1.1. Surface albedo	24
6.1.2. Transmission below the snow/ice surface	26
6.1.3. Attenuation and transmission below the ice/ocean interface	27
6.2. Solar radiation: framing sea ice at the ocean-atmosphere boundary	27
6.2.1. Forced mode	27
6.2.2. Coupled mode	27
7. Ice thermodynamics	29
7.1. Open water and new ice formation	30
7.2. Diffusion of heat	31

7.3.	Vertical growth and melt	31
7.4.	Desalination	31
7.5.	Remapping	31
7.6.	Transport in thickness space	31
7.7.	True lateral melting	31
8.	Ice-atmosphere and ice-ocean interfaces	32
8.1.	Ice-ocean interface	33
8.2.	Ice-atmosphere interface	33
9.	Output and diagnostics	34
9.1.	SIMIP diagnostics	35
9.1.1.	Missing SIMIP fields	35
9.1.2.	Links	35
9.2.	Conservation checks	35
10.	Using SI³ with a single category	36
10.1.	Enhanced conduction	37
10.2.	Virtual thin ice melting (fake lateral melting)	37
10.3.	Ridging and rafting with a single ice category	37
11.	BDY and AGRIF with SI³	38
12.	Miscellaneous topics	40
A.	Coding Rules	42
A.1.	Introduction	43
A.2.	Overview and general conventions	43
A.3.	Architecture	44
A.4.	Style rules	44
A.4.1.	Argument list format	44
A.4.2.	Array syntax	44
A.4.3.	Case	44
A.4.4.	Comments	45
A.4.5.	Continuation lines	45
A.4.6.	Declaration of arguments and local variables	45
A.4.7.	F90 Standard	45
A.4.8.	Free-Form Source	45
A.4.9.	Indentation	46
A.4.10.	Loops	46
A.4.11.	Naming Conventions: files	46
A.4.12.	Naming Conventions: modules	46
A.4.13.	Naming Conventions: variables	46
A.4.14.	Operators	46
A.4.15.	Pre processor	46
A.5.	Content rules	47
A.5.1.	Configurations	47
A.5.2.	Constants	47
A.5.3.	Declaration for variables and constants	47
A.5.4.	Headers	48
A.5.5.	Interface blocks	48
A.5.6.	I/O Error Conditions	48
A.5.7.	PRINT - ASCII output files	48
A.5.8.	Precision	49
A.5.9.	Structures	49
A.6.	Packages coding rules	49
A.6.1.	Bounds checking	49
A.6.2.	Communication	49
A.6.3.	Error conditions	49
A.6.4.	Memory management	50
A.6.5.	Optimisation	51
A.6.6.	Package attribute: PRIVATE , PUBLIC , USE , ONLY	51

A.6.7. Parallelism using MPI	51
A.7. Features to be avoided	51
Bibliography	53

To-do-list for documentation

- Documentation infrastructure ready by Jul 19, 2018.
- Abstract: v1 written.
- Disclaimer: v1 written.
- Introduction: list of chapters and change between releases to be written (1h)
- 1. Model basics: v1 written.
- Add namelist infrastructure as in NEMO doc (done)
- 2. Time, space and thickness space domain
- 3. Ice dynamics: to be written (Clem & Martin, 3h)
- 4. Ice transport: UM5 description to be written (Clem, 3h)
- 5. Ridging and rafting: V1 from previous doc. Should be refreshed. A bit too long. All namelist parameters are not explained (namdyn_rdrft).
- 6. Radiative transfer: Full rewriting required (Martin & Clem, 3-6h). Available elements from previous documentation, but mostly obsolete.
- 7. Thermodynamics: Lots of rewriting required. Available elements from previous documentation, but mostly obsolete (Martin, 6h)
- 8. Interfaces: Full writing required
- 9. Output and diagnostics: Full writing required
- 10. Single category (Martin, 3h). Full writing required.
- 11. BDY-AGRIF (Clem, 3h). Full writing required.
- 12. Miscellaneous. Full writing required.
- Melt ponds are not currently considered in the plan.
- Make sure all namelist parameters are properly described.
- Peer-review by SIWG members.

Table of contents

1.1. Rationale and assumptions	3
1.1.1. Scales, thermodynamics and dynamics	3
1.1.2. Subgrid scale variations	3
1.1.3. Thermodynamic formulation	4
1.1.4. Dynamic formulation	4
1.2. Thickness distribution framework	5
1.3. Governing equations	6
1.4. Ice Dynamics	7
1.5. Ice thermodynamics	8
1.5.1. Transport in thickness space	8
1.5.2. Thermodynamic source and sink terms	8

1.1. Rationale and assumptions

- Separation of horizontal dynamics and vertical thermodynamics;
- Dynamics: sea ice is a viscous-plastic continuum;
- Thermodynamics: sea ice is a mushy layer covered by snow
- Subgrid-scale physics

1.1.1. Scales, thermodynamics and dynamics

Because sea ice is much wider – $\mathcal{O}(100\text{-}1000\text{ km})$ – than thick – $\mathcal{O}(1\text{ m})$ – ice drift can be considered as purely horizontal: vertical motions around the hydrostatic equilibrium position are negligible. The same scaling argument justifies the assumption that heat exchanges are purely vertical*. It is on this basis that thermodynamics and dynamics are separated and rely upon different frameworks and sets of hypotheses: thermodynamics use the ice thickness distribution (Thorndike et al., 1975) and the mushy-layer (Worster, 1992) frameworks, whereas dynamics assume continuum mechanics (e.g., Leppäranta, 2011). Thermodynamics and dynamics interact by two means: first, advection impacts state variables; second, the horizontal momentum equation depends, among other things, on the ice state.

1.1.2. Subgrid scale variations

Sea ice properties – in particular ice thickness – feature important changes at horizontal scales $\mathcal{O}(1\text{m})$ (Thorndike et al., 1975). An explicit representation of these variations is not and will not be – at least in the next twenty years or so – accessible to large-scale sea ice models. Yet important features, such as energy exchanges through the ice, quite non-linearly depend on ice thickness (Maykut, 1986); whereas ice motion depends on the presence of open water, thin and thick ice at the very least, suggesting that subgrid-scale variations in ice properties must be accounted for, at least in a statistical fashion (Maykut and Thorndike, 1973).

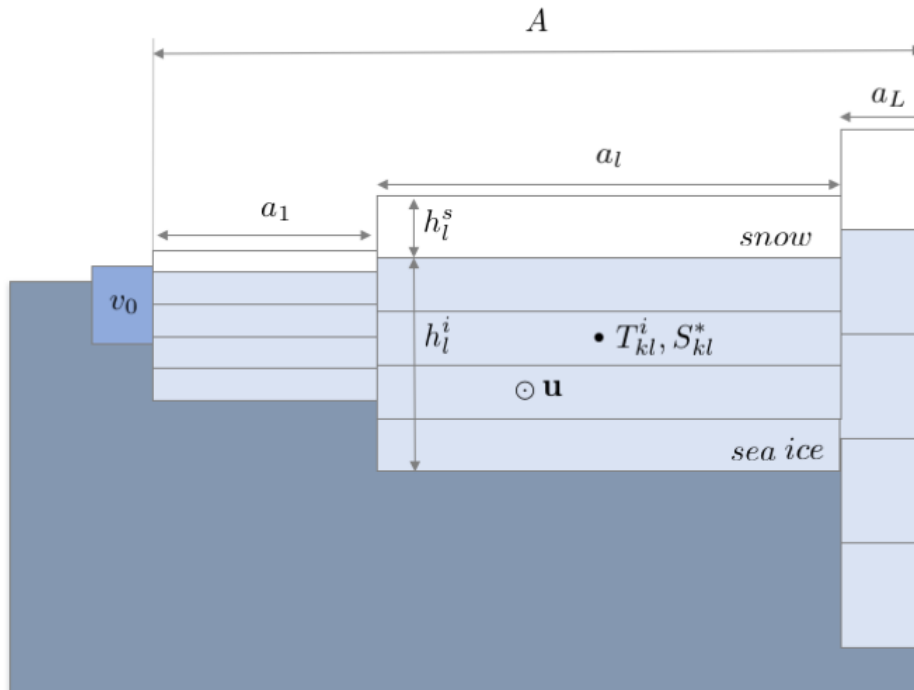


Figure 1.1.: Representation of the ice pack, using multiple categories with specific ice concentration ($a_l, l = 1, 2, \dots, L$), thickness (h_l^i), snow depth (h_l^s), vertical temperature and salinity profiles (T_{kl}^i, S_{kl}^*) and a single ice velocity vector (\mathbf{u}).

*The latter assumption is probably less valid, because the horizontal scales of temperature variations are $\mathcal{O}(10\text{-}100\text{ m})$

The *multi-category* framework (Maykut and Thorndike, 1973) addresses this issue by treating the ice thickness as an independent variable next to spatial coordinates and time, and introducing a thickness distribution[†] $g(h)$ as the main prognostic model field. In the discrete world, the thickness distribution is converted into L thickness categories. Ice thickness categories occupy a fraction of each grid cell, termed ice concentration ($a_l, l = 1, 2, \dots, L$), with specific thickness and properties.

The *single-category* framework (Hibler, 1979) tackles the subgrid-scale issue by drastically simplifying the ice thickness distribution. The grid cell is divided into open water and sea ice characterized by a single ice concentration A and mean thickness H . Single-category models (in particular LIM2) typically add parameterizations to represent the effects of unresolved ice thickness distribution on ice growth and melt (see, e.g. Mellor and Kantha, 1989; Fichefet and Maqueda, 1997).

SI³ provides the choice between either a multi- or a single-category framework. The default mode is multi-category. The single-category mode can be activated by setting the number of categories ($jpl = 1$) and by activating the virtual thickness distribution parameterizations ($nn_monocat=1$).

1.1.3. Thermodynamic formulation

Table 1.1.: Thermodynamic constants of the model.

	Description	Value	Units	Ref
c_i (cpic)	Pure ice specific heat	2067	J/kg/K	?
c_w (rcp)	Seawater specific heat	3991	J/kg/K	IOC, SCOR and IAPSO (2010)
L (lfus)	Latent heat of fusion (0°C)	334000	J/kg/K	Bitz and Lipscomb (1999)
ρ_i (rhoic)	Sea ice density	917	kg/m ³	Bitz and Lipscomb (1999)
ρ_s (rhosn)	Snow density	330	kg/m ³	Maykut and Untersteiner (1971)
μ (tmut)	Linear liquidus coefficient	0.054	°C/(g/kg)	Assur (1958)

Ice thermodynamics are formulated assuming that sea ice is covered by snow. Within each thickness category, both snow and sea ice are horizontally uniform, hence each thickness category has a specific ice thickness (h_i^l) and snow depth (h_s^l). Snow is assumed to be fresh, with constant density and thermal conductivity. Sea ice is assumed to be a *mushy layer*[‡] (Worster, 1992) of constant density, made of pure ice and brine in thermal equilibrium, related by a linear liquidus relationship (Bitz and Lipscomb, 1999). A vertically-averaged bulk salinity S_l uniquely characterizes brine fraction for each thickness category, and changes through time from a simple parametrization of brine drainage. The linear vertical salinity profile (S_{kl}^*) is reconstructed from the vertical mean (Vancoppenolle et al., 2009). The diffusion of heat affects the vertical temperature profile, discretized on a unique layer of snow and multiple ice layers (typically 2-5) for each category, whereas thermal properties depend on local brine fraction. Growth and melt rates are computed, also for each ice category. The choice of the main thermodynamic constants is described in Tab. 1.1.

Table 1.2.: LIM global variables.

Symbol	Description	Units	Code name
u	Sea ice velocity	[$m \cdot s^{-1}$]	$u_ice, v_ice(ji, jj)$
σ	Stress tensor	[$Pa \cdot m$]	$stress1_i, stress2_i$ $stress12_i(ji, jj)$
a_l	Concentration of sea ice in category l	[-]	$a_i(ji, jj, jl)$
v_i^i	Volume of sea ice per unit area in category l	[m]	$v_i(ji, jj, jl)$
v_l^s	Volume of snow per unit area in category l	[m]	$v_s(ji, jj, jl)$
e_{kl}^i	Sea ice enthalpy per unit area in layer k and category l	[$J \cdot m^{-2}$]	$e_i(ji, jj, jk, jl)$
e_l^s	Snow enthalpy per unit area in category l	[$J \cdot m^{-2}$]	$e_s(ji, jj, jl)$
M_l^s	Sea ice salt content in category l	[$g/kg \cdot m$]	$smv_i(ji, jj, jl)$

Temperature, salinity, ice thickness, and snow depth are not extensive variables and therefore not conservative. Hence, conservative, extensive variables, must be introduced to ensure mass, salt and energy conservation. There are several back-and-forth conversions from extensive (conservative) state variables (see Table 1.2) to intensive state variables of practical use (Table 1.3).

1.1.4. Dynamic formulation

The formulation of ice dynamics is based on the continuum approach. The latter holds provided the drift ice particles are much larger than single ice floes, and much smaller than typical gradient scales. This compromise is rarely achieved in

[†] $g(h)$, termed the *ice thickness distribution* is the density of probability of ice thickness (Thorndike et al., 1975).

[‡] Mushy layers are two-phase, two-component porous media.

Table 1.3.: Intensive variables of practical use.

Symbol	Description	Units
$h_l^i = v_l^i / g_l^i$	Ice thickness	[m]
$h_l^s = v_l^s / g_l^i$	Snow depth	[m]
$q_{m,kl}^i = e_{i,k}^i / (h_l^i / N)$	Ice specific enthalpy	[J.kg ⁻¹]
$q_{m,l}^s = e_l^s / h_l^s$	Snow specific enthalpy	[J.kg ⁻¹]
$T_{kl}^i = T(q_{kl}^i)$	Ice temperature	[K]
$T_l^s = T(q_l^s)$	Snow temperature	[K]
$\bar{S}_l^i = M_l^s / v_l^i$	Vertically-averaged bulk ice salinity	[g/kg]
S_{kl}^*	Depth-dependent ice salinity	[g/kg]
ϕ_{kl}	Brine fraction	[-]

practice (Leppäranta, 2011). Yet the continuum approach generates a convenient momentum equation for the horizontal ice velocity vector $u = (u, v)$, which can be solved with classical numerical methods (here, finite differences on the NEMO C-grid). The most important term in the momentum equation is internal stress. We follow the viscous-plastic (VP) rheological framework (Hibler, 1979), assuming that sea ice has no tensile strength but responds to compressive and shear deformations in a plastic way. In practice, the elastic-viscous-plastic (EVP) technique of (Bouillon et al., 2013) is used, more convenient numerically than VP. It is well accepted that the VP rheology and its relatives are the minimum complexity to get reasonable ice drift patterns (Kreyscher et al., 2000), but fail at generating the observed deformation patterns (Girard et al., 2009). This is a long-lasting problem: what is the ideal rheological model for sea ice and how it should be applied are still being debated (see, e.g. Weiss, 2013).

1.2. Thickness distribution framework

We first present the essentials of the thickness distribution framework (Thorndike et al., 1975). Consider a given region of area R centered at spatial coordinates (x) at a given time t . R could be e.g. a model grid cell. The ice thickness distribution $g(x, t, h)$ is introduced as follows:

$$g(h) = \lim_{dh \rightarrow 0} \frac{dA(h, h + dh)}{dh}, \quad (1.1)$$

where $dA(h, h + dh)$ is the surface fraction of all parts of R with ice thickness between h and $h + dh$. Using this definition, the spatial structure of ice thickness is lost (see Fig. 1.2), and h becomes an extra independent variable, next to spatial coordinates and time, that can be thought as random. g is by definition normalized to 1. The conservation of area, expressed in terms of $g(h)$, is given by (Thorndike et al., 1975):

$$\frac{\partial g}{\partial t} = -\nabla \cdot (gu) - \frac{\partial}{\partial h} (fg) + \psi, \quad (1.2)$$

where the terms on the right hand side refer to horizontal transport, thermodynamic transport in thickness space (f , m/s is the growth/melt rate), and mechanical redistribution rate, e.g. by ridging and rafting, where ψ must conserve ice area and volume by construction.

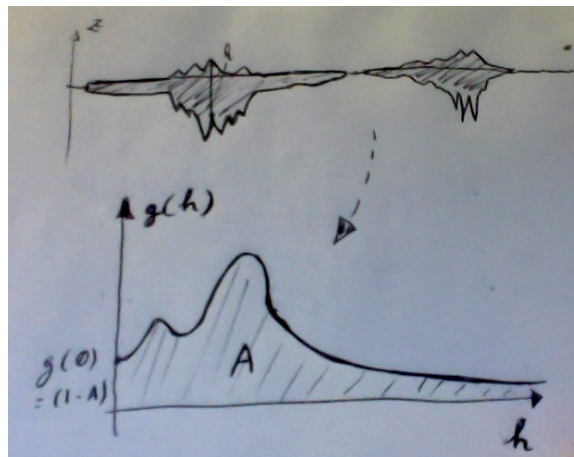


Figure 1.2.: Representation of the relation between real thickness profiles and the ice thickness distribution function $g(h)$

In numerical implementations, the thickness distribution is discretized into several thickness categories, with specific ice concentration a_l and ice volume per area v_l^i :

$$a_l = \int_{H_{l-1}^*}^{H_l^*} dh \cdot g(h), \quad (1.3)$$

$$v_l^i = \int_{H_{l-1}^*}^{H_l^*} dh \cdot h \cdot g(h). \quad (1.4)$$

Ice volume per area is the extensive counterpart for ice thickness, connected with volume through $h_l^i = v_l^i/a_l$. Evolution equations for extensive variables can be readily derived from equation 1.5 by integration between thickness boundaries of the l^{th} category (Bitz et al., 2001). This applies to all model extensive variables (see Table 1.2). For ice area, this reads:

$$\frac{\partial a_l}{\partial t} = -\nabla \cdot (a_l \mathbf{u}) + \Theta_l^a + \int_{H_{l-1}^*}^{H_l^*} dh \psi. \quad (1.5)$$

where Θ_l^a refers to the effect of thermodynamics. Enthalpy is a particular case because it also has a vertical depth dependence z , which corresponds to K vertical layers of equal thickness. The solution adopted here, following from Zhang and Rothrock (2001), is that enthalpy from the individual layers are conserved separately. This is a practical solution, for lack of better.

One of the major actions of LIM is to resolve conservation equations for all extensive variables that characterize the ice state. Let us now connect this detailed information with classical sea ice fields. The ice concentration A and the ice volume per area[§] V_i (m) directly derive from g :

$$A(\mathbf{x}, t) = \int_{0^+}^{\infty} dh \cdot g(h, \mathbf{x}, t) \sim A_{ij} = \sum_{l=1}^L a_{ijl}, \quad (1.6)$$

$$V_i(\mathbf{x}, t) = \int_0^{\infty} dh \cdot g(h, \mathbf{x}, t) \cdot h \sim V_{ij}^i = \sum_{l=1}^L v_{ijl}^i. \quad (1.7)$$

$$(1.8)$$

where the 0^+ boundary implies that the means exclude open water. The mean ice thicknesses H_i (m) is:

$$H_i = V_i/A, \quad (1.9)$$

whereas the open water fraction is simply $1 - A$.

1.3. Governing equations

Let us now readily present the set of the LIM "governing" equations in the framework of the assumptions developed above. The conservation of horizontal momentum reads:

$$m \frac{\partial \mathbf{u}}{\partial t} = \nabla \cdot \boldsymbol{\sigma} + A(\boldsymbol{\tau}_a + \boldsymbol{\tau}_w) - m f \mathbf{k} \times \mathbf{u} - m g \nabla \eta, \quad (1.10)$$

where $m = \rho_i V_i + \rho_s V_s$ is the ice and snow mass per unit area, \mathbf{u} is the ice velocity, $\boldsymbol{\sigma}$ is the internal stress tensor, $\boldsymbol{\tau}_a$ and $\boldsymbol{\tau}_w$ are the air and ocean stresses, respectively, f is the Coriolis parameter, \mathbf{k} is a unit vector pointing upwards, g is the gravity acceleration and η is the ocean surface elevation. The EVP approach used in LIM (Bouillon et al., 2013) gives the stress tensor as a function of the strain rate tensor $\dot{\epsilon}$ and some of the sea ice state variables:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\dot{\epsilon}, \text{ice state}). \quad (1.11)$$

To the exception of velocity and internal stress, all extensive variables in Table 1.2 follow a conservation equation of the form:

$$\frac{\partial X}{\partial t} = -\nabla \cdot (\mathbf{u}X) + \Theta^X + \Psi^X, \quad (1.12)$$

including the effects of transport, thermodynamics (Θ^X) and mechanical redistribution (Ψ^X). Solving these *jpl.(4+2.jpj)* equations gives the temporal evolution of \mathbf{u} , $\boldsymbol{\sigma}$ and the rest of the global (extensive) variables listed in Table 1.2.

[§]Ice volume per area is equivalent to the grid-cell averaged ice thickness.

1.4. Ice Dynamics

Dynamical processes include the conservation of momentum, rheology, transport and mechanical redistribution. To resolve the momentum equation, atmospheric stress is taken either as forcing or from an atmospheric model, oceanic stress and sea surface elevation from the ocean model, the Coriolis term is trivial. The last term, the divergence of the internal stress tensor σ , is the most critical term in the momentum equation and requires a rheological formulation. The EVP approach used in LIM gives the stress tensor components as (Bouillon et al., 2013):

$$\sigma_{ij} = \frac{P}{2(\Delta + \Delta_{min})} \left[(\dot{\epsilon}_{kk} - \Delta)\delta_{ij} + \frac{1}{e^2} (2\dot{\epsilon}_{ij} - \dot{\epsilon}_{kk}\delta_{ij}) \right], \quad (1.13)$$

where Δ is a particular measure of the deformation rate, Δ_{min} a parameter determining a smooth transition from pure viscous flow ($\Delta \ll \Delta_{min}$) to pure plastic flow ($\Delta \gg \Delta_{min}$), and e is a parameter giving the ratio between the maximum compressive stress and twice the maximum shear stress. In the pure plastic regime, the stress principal components should lie on the edge of an elliptical yield curve (Fig. 7.1). In the viscous regime, they are within the ellipse. The ice strength P determines the plastic failure criterion and connects the momentum equation with the state of the sea ice. P is not well constrained and must be parameterized. The heuristic option of Hibler (1979) was here adopted as a reference formulation:

$$P = P^* V_i e^{-C(1-A)}, \quad (1.14)$$

where P^* and C are empirical constants (see Table 1.4 for the values of the main model parameters).

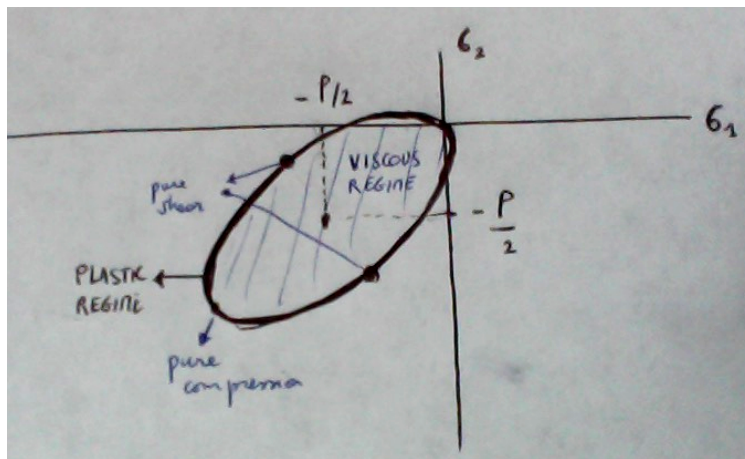


Figure 1.3.: Elliptical yield curve used in the VP rheologies, drawn in the space of the principal components of the stress tensor (σ_1 and σ_2).

Table 1.4.: Main model parameters.

	Description	Value	Units	Ref
P^* (rn_pstar)	ice strength thickness param.	20000	N/m ²	-
C (rn_crhg)	ice strength concentration param.	20	-	(Hibler, 1979)
H^* (rn_hstar)	maximum ridged ice thickness param.	25	m	(Lipscomb et al., 2007)
p (rn_por_rdg)	porosity of new ridges	0.3	-	(Leppäranta et al., 1995)
$amax$ (rn_amax)	maximum ice concentration	0.999	-	-
h_0 (rn_hnewice)	thickness of newly formed ice	0.1	m	-

Transport connects the horizontal velocity fields and the rest of the ice properties. LIM assumes that the ice properties in the different thickness categories are transported at the same velocity. The scheme of Prather (1986), based on the conservation of 0, 1st and 2nd order moments in x - and y -directions, is used, with some numerical diffusion if desired. Whereas this scheme is accurate, nearly conservative, it is also quite expensive since, for each advected field, five moments need to be advected, which proves CPU consuming, in particular when multiple categories are used. Other solutions are currently explored.

The dissipation of energy associated with plastic failure under convergence and shear is accomplished by rafting (overriding of two ice plates) and ridging (breaking of an ice plate and subsequent piling of the broken ice blocks into pressure ridges). Thin ice preferentially rafts whereas thick ice preferentially ridges (Tuhkuri, 2002). Because observations of these processes are limited, their representation in LIM is rather heuristic. The amount of ice that rafts/ridges depends on the strain rate tensor invariants (shear and divergence) as in (Flato and Hibler, 1995), while the ice categories involved

are determined by a participation function favouring thin ice (Lipscomb et al., 2007). The thickness of ice being deformed (h') determines whether ice rafts ($h' < 0.75$ m) or ridges ($h' > 0.75$ m), following Haapala (2000). The deformed ice thickness is $2h'$ after rafting, and is distributed between $2h'$ and $2\sqrt{H^*h'}$ after ridging, where $H^* = 25$ m (Lipscomb et al., 2007). Newly ridged ice is highly porous, effectively trapping seawater. To represent this, a prescribed volume fraction (30%) of newly ridged ice (Leppäranta et al., 1995) incorporates mass, salt and heat are extracted from the ocean. Hence, in contrast with other models, the net thermodynamic ice production during convergence is not zero in LIM, since mass is added to sea ice during ridging. Consequently, simulated new ridges have high temperature and salinity as observed (Høyland, 2002). A fraction of snow (50 %) falls into the ocean during deformation.

1.5. Ice thermodynamics

In this section, we develop the underlying principles of the thermodynamic formulation, summarized in the term Θ^X , where X refers to all extensive state variables. Θ^X includes the contributions of transport in thickness space and thermodynamic source and sink terms.

1.5.1. Transport in thickness space

Transport in thickness space describes how vertical growth and melt moves ice state variables among the different thicknesses at a velocity $f(h)$, the net ice growth/melt rate, which needs to be first computed. In discretized form, this term moves ice properties between neighbouring categories. The linear remapping scheme of Lipscomb (2001) is used. This scheme is semi-lagrangian, second-order, is less diffusive and converges faster than other options.

1.5.2. Thermodynamic source and sink terms

Since heat, salt and mass are strongly inter-dependent for sea ice, the thermodynamic source and sink terms are treated together. They include the changes in extensive sea ice state associated with thermodynamic processes. The latter are separated in two main parts: (i) open water fraction processes, where atmosphere and ocean are in direct interaction; and (ii) vertical ice thermodynamic processes, driven by surface snow/ice-atmosphere and basal ice-ocean exchanges, for each thickness category. For each part, first, the energy available or lost is specified. Then the impact on mass exchanges is evaluated. The latter part requires to specify how sea ice and snow responds to energy supply or loss, which is achieved through the enthalpy formulation.

Enthalpy formulation

A first overarching aspect of the thermodynamic calculations is the specification of the response of sea ice to energy supply. This is achieved by defining the internal energy (or enthalpy[¶]). This ultimately relies on the response of the phase composition to salinity and temperature changes. The enthalpy formulation used in LIM is based on the following assumptions:

- Sea ice is gas-free, composed solely of pure ice and saline brine, characterized by brine fraction ϕ ;
- brine and pure ice are in thermodynamic equilibrium;
- the salinity-dependence of the freezing point is linear (linear liquidus);
- the density of the sea ice (ice+brine) medium is constant (ρ_i).

Based on these, brine fraction reduces to $\phi = -\mu S/T$ (see Fig. 1.4), where μ relates the freezing point of brine to salinity, and one can derive the specific enthalpy $q_m(S, T)$, defined as the energy required to warm and melt a unit control volume of sea ice at temperature T (in Celsius) and salinity S until 0°C , taken as a reference zero-energy level (Bitz and Lipscomb, 1999; Schmidt et al., 2004):

$$q_m(S, T) = \left[c_i(T + \mu S) - L \left(1 + \frac{\mu S}{T} \right) - c_w \mu S \right] \quad (1.15)$$

where c_i is pure ice specific heat, L is latent heat of fusion at 0°C , and c_w is water specific heat. The first term expresses the warming of solid ice. The second term expresses internal change in brine fraction, which is often the largest because the Stefan number ($c_i T/L$) is generally small. The last term gives the warming of the remaining water from $T_{fr} = -\mu S$ until 0°C . Similar, but simpler and linear expressions for snow and water can be derived.

The second overarching aspect is that all growth and melt processes must be calculated consistently with the enthalpy formulation. Energetics of phase transitions are handled using the formalism of Schmidt et al. (2004). For each phase

[¶]Wording it internal energy or enthalpy is equivalent since pressure effects are not considered.

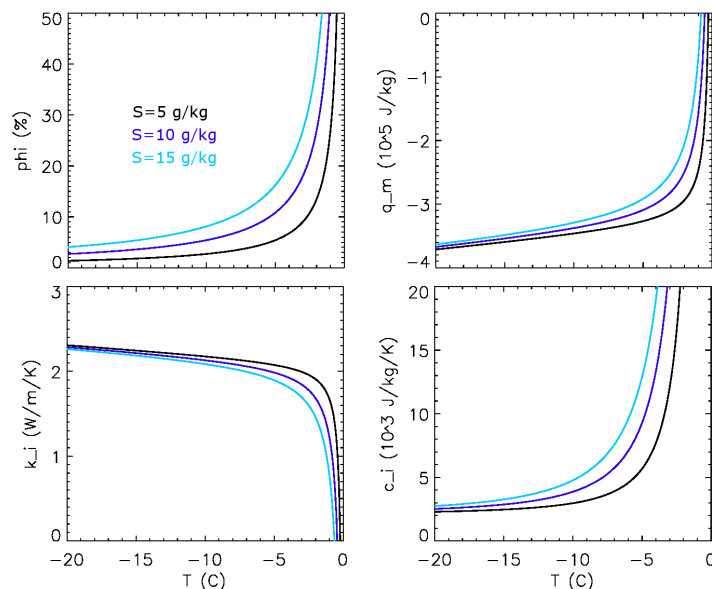


Figure 1.4.: Thermal properties of sea ice vs temperature for different bulk salinities: brine fraction, specific enthalpy, thermal conductivity, and effective specific heat.

transition, initial and final states (temperature and salinity) are defined, and the ice-to-ocean mass flux to the ice F_m (kg/s) relates to the energy gain or loss ΔQ through:

$$\Delta Q / \Delta q_m = F_m, \quad (1.16)$$

where Δq_m is the change in specific enthalpy involved in the considered phase transition, from initial to final state.

Open water processes

As part of the sea ice thermodynamic calculations, a heat budget estimate for the uppermost ocean level (B^{opw}) must be included, to compute the rate of new ice formation or the contribution of sensible heat to bottom melting. B^{opw} includes:

- the absorption of a fraction f_1^{qst} of solar radiation (given by radiative transfer component of the ocean model);
- the non-solar heat flux absorbed at the surface;
- the sensible heat content of precipitation
- the sensible heat flux from the ocean to the sea ice ($A.F_w$)

Other contributions are not assumed not to contribute. The ocean-to-ice sensible heat flux is formulated the bulk formula of (McPhee, 1992).

If B^{opw} is such that the SST would decrease below the freezing point, the remainder of the heat is used to form new ice. The heat loss is converted into a volume of new ice v_0 . The thickness h_0 of the new ice grown during a sea ice time step depends on unresolved small-scale currents and waves and is prescribed. The fraction $a_0 = v_0/h_0$ is computed accordingly. The salinity of this new ice S_0 is given by the salinity-thickness empirical relationship of Kovacs (1996). The temperature assumed for this new ice is the local freezing point. If by contrast B^{opw} is positive and there still is ice in the grid cell, then B^{opw} is directly redirected to bottom melting. This argument follows from Maykut and MCPhee (1995), who found that most of solar heat absorbed in the surface waters is converted into melting. In practise, this prevents the SST to be above freezing as long ice is present.

B^{opw} can be seen as a predictor of the heat budget of the first ocean level. As such, it only helps to compute new ice formation and the extra bottom melt in summer, but is not part of the conservation of heat in the model. To ensure heat conservation, the heat effectively contributing to changing sea ice is removed from the non-solar flux sent to the ocean. This includes: (i) the heat loss used for ice formation, (ii) the heat gain used to melt ice, and (iii) the sensible heat given by the ocean to the ice. Finally, because ice dynamics are not able to maintain the small amount of open water that is observed, a maximum ice fraction (*amax*, < 1) is prescribed.

Vertical ice thermodynamic processes

The second part of the computations regard the computation of purely vertical processes in the ice-covered part of the grid cell, similarly for each ice category.

Surface melt, basal growth and melt and diffusion of heat. The surface melt rate, as well as the basal growth / melt rate depend on the energy budget at the upper and lower interfaces, respectively, between the external fluxes either from the atmosphere or the ocean, and the internal conduction fluxes. The internal conduction fluxes depend on the internal temperature profile, which is determined by solving the enthalpy equation:

$$\rho \frac{\partial q_m}{\partial t} = - \frac{\partial}{\partial z} (F_c + F_r). \quad (1.17)$$

which state that the local change in enthalpy is given by the divergence of the vertical conduction ($F_c = -k(S, T)\partial T/\partial z$) and radiation (F_r) fluxes. ρ is the density of ice or snow. Re-expressed as a function of temperature, this becomes the heat diffusion equation. This equation is non-linear in T , because of q and k , and its main specificity is that internal melting requires large amounts of energy near the freezing point. The thermal conductivity is formulated following Pringle et al. (2007), empirically accounting for the reduction of thermal conductivity at large brine fractions.

At the ice base, we assume that the temperature is at the local freezing point. Ice grows or melt if the heat balance between the oceanic sensible heat flux (F_w) and internal conduction is negative or positive.

At the ice surface, the boundary condition on the heat diffusion equation is:

$$Q^{sr} + Q^{ns}(T_{su}) = F_c + Q^{sum}. \quad (1.18)$$

where Q^{sr} and Q^{ns} are the net downwelling atmospheric solar and non-solar flux components. If the solution of this equation without melting gives a surface temperature (T_{su}) below 0° C, then there is no melting and the heat available for surface melting $Q^{sum} = 0$. Otherwise T_{su} is capped at 0° C and Q^{sum} is calculated as a residual.

Radiation. Radiation contributes to the surface and internal heat budget. The radiative transfer scheme is currently basic, composed of surface albedo, transmission through the ice interior and attenuation with vertical depth. The albedo is computed empirically as a function of ice thickness, snow depth and surface temperature, using a reformulation of the parameterization of Shine and Henderson-Sellers (1985). When snow is present, all the absorbed radiation is transformed into sensible heat available for conduction or melting. Over snow-free ice, a fraction of solar radiation is transmitted below the surface and attenuates exponentially with depth, until it reaches the base of the ice.

Growth and melt processes. Snow grows from precipitation and loses mass from melting and snow-ice conversion once the snow base is below sea level. Sea ice grows and melts by various means. Ice forms by congelation or melt at the base, can melt at the surface and form from snow-to-ice conversion at the snow-ice interface if the latter is below sea level. Some new ice is also added to the system when seawater is trapped into newly formed pressure ridges.

Salt dynamics. Bulk salinity is empirically parameterized, as a function of salt uptake during growth, gravity drainage and flushing. The shape of the vertical profile depends on the bulk salinity (Vancoppenolle et al., 2009).

Single-category parameterizations. If the single-category representation is adopted, then two parameterizations can be activated, following (Fichefet and Maqueda, 1997). First, the thermal conductivity of both ice and snow is multiplied by a factor > 1 accounting for the unresolved thin ice, effectively increasing the ice growth rate. Second, to account for the loss of thin ice in summer, the ice concentration is reduced in proportion to the loss of ice thickness. Both parameterizations have been tuned to match the results in multi-category mode.

Time, space and thickness space domain

Table of contents

2.1. Time domain	12
2.2. Spatial domain	12
2.3. Thickness space domain	13

Having defined the model equations in previous Chapter, we need now to choose the numerical discretization. In the present chapter, we provide a general description of the SI³ discretization strategy, in terms of time, space and thickness, which is considered as an extra independent variable.

Sea ice state variables are typically expressed as:

$$X(ji, jj, jk, jl). \quad (2.1)$$

ji and jj are x-y spatial indices, as in the ocean. $jk = 1, \dots, nlay_i$ corresponds to the vertical coordinate system in sea ice (ice layers), and only applies to vertically-resolved quantities (ice enthalpy and salinity). $jl = 1, \dots, jpl$ corresponds to the ice categories, discretizing thickness space.

2.1. Time domain

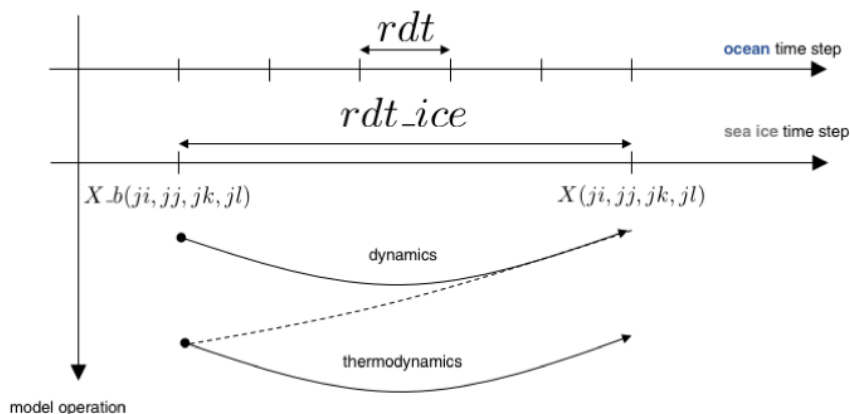


Figure 2.1.: Schematic representation of time stepping in SI³, assuming $nn_fsbc = 5$.

The sea ice time stepping is synchronized with that of the ocean. Because of the potentially large numerical cost of sea ice physics, in particular rheology, SI³ can be called every nn_fsbc time steps ($namsbc$ in *namelist_ref*). The sea ice time step is therefore $rdt_ice = rdt * nn_fsbc$. In terms of quality, the best value for nn_fsbc is 1, providing full consistency between sea ice and oceanic fields. Larger values (typically 2 to 5) can be used but numerical instabilities can appear because of the progressive decoupling between the state of sea ice and that of the ocean, hence changing nn_fsbc must be done carefully.

Ice dynamics (rheology, advection, ridging/rafting) and thermodynamics are called successively. To avoid pathological situations, thermodynamics were chosen to be applied on fields that have been updated by dynamics, in a somehow semi-implicit procedure.

There are a few iterative / subcycling procedures throughout the code, notably for rheology, advection, ridging/ rafting and the diffusion of heat. In some cases, the arrays at the beginning of the sea ice time step are required. Those are referred to as X_b .

2.2. Spatial domain

The horizontal indices ji and jj are handled as for the ocean in NEMO, assuming C-grid discretization and in most cases a finite difference expression for scale factors.

The vertical index $jk = 1, \dots, nlay_i$ is used for enthalpy (temperature) and salinity. In each ice category, the temperature and salinity profiles are vertically resolved over $nlay_i$ equally-spaced layers. The number of snow layers can currently only be set to $nlay_s = 1$ (Fig. 2.2).

To increase numerical efficiency of the code, the two horizontal dimensions of an array $X(ji, jj, jk, jl)$ are collapsed into one (array $X_1d(ji, jk, jl)$) for thermodynamic computations, and re-expanded afterwards.

```

!-----
&nampar      !   Generic parameters
!-----
jpl          = 5           ! number of ice categories
nlay_i      = 2           ! number of ice layers

```

Figure 2.2.: Vertical grid of the model, used to resolve vertical temperature and salinity profiles

```

nlay_s      = 1           ! number of snow layers (only 1 is working)
ln_virtual_itd = .false.  ! virtual ITD mono-category parameterization (jpl=1 only)
                    ! i.e. enhanced thermal conductivity & virtual thin ice melting

ln_icedyn   = .true.     ! ice dynamics (T) or not (F)
ln_icethd   = .true.     ! ice thermo (T) or not (F)
rn_amax_n   = 0.997      ! maximum tolerated ice concentration NH
rn_amax_s   = 0.997      ! maximum tolerated ice concentration SH
cn_icerst_in  = "restart_ice" ! suffix of ice restart name (input)
cn_icerst_out = "restart_ice" ! suffix of ice restart name (output)
cn_icerst_indir = "."     ! directory to read input ice restarts
cn_icerst_outdir = "."    ! directory to write output ice restarts
/

```

2.3. Thickness space domain

```

!-----
&namitd      ! Ice discretization
!-----
ln_cat_hfn   = .true.     ! ice categories are defined by a function following rn_himean**(-0.05)
rn_himean    = 2.0        ! expected domain-average ice thickness (m)
ln_cat_usr   = .false.    ! ice categories are defined by rn_catbnd below (m)
rn_catbnd    = 0.,0.45,1.1,2.1,3.7,6.0
rn_himin     = 0.1        ! minimum ice thickness (m) used in remapping
/

```

Thickness space is discretized using $j_l = 1, \dots, j_{pl}$ thickness categories, with prescribed boundaries $hi_max(j_l - 1)$, $hi_max(j_l)$. Following Lipscomb (2001), ice thickness can freely evolve between these boundaries. The number of ice categories j_{pl} can be adjusted from the namelist (*nampar*).

There are two means to specify the position of the thickness boundaries of ice categories. The first option (*ln_cat_hfn*) is to use a fitting function that places the category boundaries between 0 and $3\bar{h}$, with \bar{h} the expected mean ice thickness over the domain (namelist parameter *rn_himean*), and with a greater resolution for thin ice (Fig. 2.3). More specifically, the upper limits for ice in category $j_l = 1, \dots, j_{pl} - 1$ are:

$$hi_max(j_l) = \left(\frac{j_l \cdot (3\bar{h} + 1)^\alpha}{(j_{pl} - j_l)(3\bar{h} + 1)^\alpha + j_l} \right)^{\alpha^{-1}} - 1, \quad (2.2)$$

with $h_{i_max}(0)=0$ m and $\alpha = 0.05$. The last category has no upper boundary, so that it can contain arbitrarily thick ice.

Figure 2.3.: Boundaries of the model ice thickness categories (m) for varying number of categories and prescribed mean thickness (\bar{h}). The formerly used *tanh* formulation is also depicted.

The other option (`ln_cat_usr`) is to specify category boundaries by hand using `rn_catbnd`. The first category must always be thicker than `rn_himin` (0.1 m by default).

The choice of ice categories is important, because it constrains the ability of the model to resolve the ice thickness distribution. The latest study (Massonnet et al., 2018) recommends to use at least 5 categories, which should include one thick ice with lower bounds at ~ 4 m and ~ 2 m for the Arctic and Antarctic, respectively, for allowing the storage of deformed ice.

With a fixed number of cores, the cost of the model linearly increases with the number of ice categories. Using `jpl = 1` single ice category is also much cheaper than with 5 categories, but seriously deteriorates the ability of the model to grow and melt ice. Indeed, thin ice thickens faster than thick ice, and shrinks more rapidly as well. When `nn_virtual_itd=1` (`jpl = 1` only), two parameterizations are activated to compensate for these shortcomings. Heat conduction and areal decay of melting ice are adjusted to closely approach the 5 categories case.

Table of contents

In this chapter, we describe how the momentum equation is solved.

I guess we could be brief. Gurvan and Clem should take the lead to write this.

- 1) aEVP approach, revisited by Bouillon et al (2013).
- 2) Short description of the numerical method. NEMO scale factors.
- 3) Landfast ice option
- 4) Boundary conditions (free-slip, no-slip, ...)
- 4) Practical use of namelist parameters (nev, telast, ...).

Table of contents

4.1. Second order moments conserving (Prather 1986) scheme (<i>ln_adv_Pra</i>)	18
4.2. 5 th order flux-corrected transport scheme (UM5)	18

As soon as ice dynamics are activated (*In_dyn_xxx*), all extensive state variables are to be advected following the horizontal velocity field.

4.1. Second order moments conserving (Prather 1986) scheme (*In_adv_Pra*)

The scheme of Prather (1986) explicitly computes the conservation of second-order moments of the spatial distribution of global sea ice state variables. This scheme preserves positivity of the transported variables and is practically non-diffusive. It is also computationally expensive, however it allows to localize the ice edge quite accurately. As the scheme is conditionally stable, the time step is split into two parts if the ice drift is too fast, based on the CFL criterion.

State variables per unit grid cell area are first multiplied by grid cell area. Then, for each state variable, the 0^{th} (mean), 1^{st} (x, y) and 2^{nd} (xx, xy, yy) order moments of the spatial distribution are transported. At 1st time step, all moments are zero (if prescribed initial state); or read from a restart file, and then evolve through the course of the run. Therefore, for each global variable, 5 additional tracers have to be kept into memory and written in the restart file, which significantly increases the required memory. Advection following x and y are computed independently. The succession order of x - and y - advection is reversed every day.

4.2. 5^{th} order flux-corrected transport scheme (UM5)

Table of contents

5.1. Dynamical inputs	20
5.2. The two deformation modes: ridging and rafting	21
5.3. Participation functions	21
5.4. Transfer functions	21
5.5. Ridging shift	22
5.6. Mechanical redistribution for other global ice variables	22

This chapter focuses on how LIM solves the red part of the general equation:

$$\frac{\partial X}{\partial t} = -\nabla \cdot (\mathbf{u}X)\Theta^X + \Psi^X, \quad (5.1)$$

where X refers to any global sea ice state variable.

Divergence and shear open the ice pack and create ice of zero thickness. Convergence and shear consumes thin ice and create thicker ice by mechanical deformation. The redistribution functions Ψ^X describe how opening and mechanical deformation redistribute the global ice state variables into the various ice thickness categories.

The fundamental redistribution function is Ψ^g , which accounts for area redistribution. The other redistribution functions Ψ^X associated with other state variables will derive naturally. The redistribution function Ψ^g should first ensure area conservation. By integrating the evolution equation for $g(h)$ over all thicknesses, recalling that $\int_0^\infty g(h) = 1$, and that the total areal change due to thermodynamics must be zero, e.g. $\int_0^\infty \partial(fg)/\partial h = 0$, then the area conservation reads:

$$\int_0^\infty h\Psi^g dh = \nabla \cdot \mathbf{u}. \quad (5.2)$$

Second, we must say something about volume conservation, and this will be done more specifically later. Following Thorndike et al. (1975), we separate the Ψ^X 's into **(i) dynamical inputs**, **(ii) participation functions**, i.e., how much area of ice with a given thickness participates to mechanical deformation **(iii) transfer functions**, i.e., where in thickness space the ice is transferred after deformation.

5.1. Dynamical inputs

A general expression of Ψ^g , the mechanical redistribution function associated to the ice concentration, was proposed by Thorndike et al. (1975):

$$\Psi^g = |\dot{\epsilon}|[\alpha_o(\theta)\delta(h) + \alpha_d(\theta)w_d(h, g)], \quad (5.3)$$

which is convenient to separate the dependence in \mathbf{u} from those in g and h . The first and second terms on the right-hand side correspond to opening and deformation, respectively. $|\dot{\epsilon}| = (\dot{\epsilon}_I^2 + \dot{\epsilon}_{II}^2)^{1/2}$, where $\dot{\epsilon}_I = \nabla \cdot \mathbf{u}$ and $\dot{\epsilon}_{II}$ are the strain rate tensor invariants; $\theta = \text{atan}(\dot{\epsilon}_{II}/\dot{\epsilon}_I)$. $w_d(h, g)$, the deformation mode will be discussed in the next section. $|\dot{\epsilon}|\alpha_o$ and $|\dot{\epsilon}|\alpha_d$ are called the lead opening and closing rates, respectively.

The **dynamical** inputs of the mechanical redistribution in LIM are:

- $|\dot{\epsilon}|\alpha_o$, the opening rate,
- $|\dot{\epsilon}|\alpha_d$, the net closing rate.

Following Thorndike et al. (1975), we choose $\int_0^\infty w_d(h, g) = -1$. In order to satisfy area conservation, the relation $|\dot{\epsilon}|\alpha_o - |\dot{\epsilon}|\alpha_d = \nabla \cdot \mathbf{u}$ must be verified. In the model, there are two ways to compute the divergence of the velocity field. A first way is to use the velocity components ($\dot{\epsilon}_I = \nabla \cdot \mathbf{u}^{rhg}$) as computed after the rheology (superscript *rhg*). Another way is to derive it from the horizontal transport of ice concentration and open water fraction. In principle, the equality $A^o + \sum_{l=1}^L g_L^i = 1$ should always be verified. However, after ice transport (superscript *trp*), this is not the case, and one can diagnose a velocity divergence using the departure from this equality: $\nabla \cdot \mathbf{u}^{trp} = (1 - A^o - \sum_{l=1}^L g_L^i)/\Delta t$. In general, we will use $\dot{\epsilon}_I$ unless otherwise stated.

The **net closing rate** is written as a sum of two terms representing the energy dissipation by shear and convergence (Flato and Hibler, 1995):

$$|\dot{\epsilon}|\alpha_d(\theta) = C_s \frac{1}{2} (\Delta - |\dot{\epsilon}_I|) - \min(\dot{\epsilon}_I, 0), \quad (5.4)$$

where Δ is a measure of deformation (defined in the rheology section). The factor $C_s = 0.5$ (C_s in *namelist_ice*) is added to allow for energy sinks other than ridge building (e.g., sliding friction) during shear. In case of convergence, the closing rate must be large enough to satisfy area conservation after ridging, so we take:

$$|\dot{\epsilon}|\alpha_d(\theta) = \max(|\dot{\epsilon}|\alpha_d(\theta), -\nabla \cdot \mathbf{u}^{trp}) \quad \text{if } \nabla \cdot \mathbf{u} < 0. \quad (5.5)$$

The **opening rate** is obtained by taking the difference:

$$|\dot{\epsilon}|\alpha_o = |\dot{\epsilon}|\alpha_d = \nabla \cdot \mathbf{u}^{trp} \quad (5.6)$$

5.2. The two deformation modes: ridging and rafting

The deformation mode is separated into ridging w^{ri} and rafting w^{ra} modes:

$$w^d(h, g) = w^{ri}(g, h) + w^{ra}(g, h). \quad (5.7)$$

Rafting is the piling of two ice sheets on top of each other. Rafting doubles the participating ice thickness and is a volume-conserving process. Babko (2002) concluded that rafting plays a significant role during initial ice growth in fall, therefore we included it into the model.

Ridging is the piling of a series of broken ice blocks into pressure ridges. Ridging redistributes participating ice on a various range of thicknesses. Ridging does not conserve ice volume, as pressure ridges are porous. Therefore, the volume of ridged ice is larger than the volume of new ice being ridged. In the model, newly ridged ice has a prescribed porosity $p = 30\%$ (*ridge_por* in *namelist_ice*), following observations (Leppäranta et al., 1995; Høyland, 2002). The importance of ridging is now since the early works of (Thorndike et al., 1975).

The deformation modes are formulated using **participation** and **transfer** functions with specific contributions from ridging and rafting:

$$w_d(h, g) = -[b^{ra}(h) + b^{ri}(h)]g(h) + n^{ra}(h) + n^{ri}(h). \quad (5.8)$$

$b^{ra}(h)$ and $b^{ri}(h)$ are the rafting and ridging participation functions. They determine which regions of the ice thickness space participate in the redistribution. $n^{ra}(h)$ and $n^{ri}(h)$, called transfer functions, specify how thin, deformation ice is redistributed onto thick, deformed ice. Participation and transfer functions are normalized in order to conserve area.

5.3. Participation functions

We assume that the participation of ice in redistribution does not depend upon whether the deformation process is rafting or ridging. Therefore, the participation functions can be written as follows:

$$b^{ra}(h) = \beta(h)b(h), \quad (5.9)$$

$$b^{ri}(h) = [1 - \beta(h)]b(h), \quad (5.10)$$

where $b(h)$ is an exponential weighting function with an e-folding scale a^* (Lipscomb et al., 2007) (*astar* in *namelist_ice*) which preferentially apportions the thinnest available ice to ice deformation:

$$b(h) = \frac{\exp[-G(h)/a^*]}{a^*[1 - \exp(-1/a^*)]}, \quad (5.11)$$

It is numerically more stable than the original version of Thorndike et al. (1975). This scheme is still present in the code and can be activated using *partfun_swi* from *namelist_ice*, with the associated parameter *Gstar*.

$\beta(h)$ partitions deformation ice between rafted and ridged ice. $\beta(h)$ is formulated following Haapala (2000), using the Parmeter (1975) law, which states that, under a critical participating ice thickness h_P , ice rafts, otherwise it ridges:

$$\beta(h) = \frac{\tanh[-C_{ra}(h - h_P)] + 1}{2}, \quad (5.12)$$

where $C_{ra} = 5 \text{ m}^{-1}$ (*Craft* in *namelist_ice*) and $h_P = 0.75 \text{ m}$ (*hparameter* in *namelist_ice*) (Haapala, 2000; Babko, 2002). The *tanh* function is used to smooth the transition between ridging and rafting. If *namelist* parameter *raftswi* is set to 0, ice only ridges and does not raft.

5.4. Transfer functions

The rafting transfer function assumes a doubling of ice thickness :

$$n^{ra}(h) = \frac{1}{2} \int_0^\infty \delta(h - 2h')b(h')g(h')dh, \quad (5.13)$$

where δ is the Dirac delta function.

The ridging transfer function is :

$$n^{ri}(h) = \int_0^\infty \gamma(h', h)(1 + p)b(h')g(h')dh. \quad (5.14)$$

The redistributor $\gamma(h', h)$ specifies how area of thickness h' is redistributed on area of thickness h . We follow (Hibler, 1980) who constructed a rule, based on observations, that forces all ice participating in ridging with thickness h' to be

linearly distributed between ice that is between $2h'$ and $2\sqrt{H^*h'}$ thick, where $H^* = 100$ m ($Hstar$ in *namelist_ice*). This in turn determines how to construct the ice volume redistribution function Ψ^v . Volumes equal to participating area times thickness are removed from thin ice. They are redistributed following Hibler's rule. The factor $(1 + p)$ accounts for initial ridge porosity p (*ridge_por* in *namelist_ice*, defined as the fractional volume of seawater initially included into ridges. In many previous models, the initial ridge porosity has been assumed to be 0, which is not the case in reality since newly formed ridges are porous, as indicated by in-situ observations (Leppäranta et al., 1995; Høyland, 2002). In other words, LIM3 creates a higher volume of ridged ice with the same participating ice.

For the numerical computation of the integrals, we have to compute several temporary values:

- The thickness of rafted ice $h_l^{ra} = 2h_l^i$
- The mean thickness of ridged ice $h_l^{ri,mean} = \max(\sqrt{H^*h_l^i}, h_l^i \cdot 1.1)$
- The minimum thickness of ridged ice $h_l^{ri,min} = \min[2 * h_l^i, 0.5 \cdot (h_l^{ri,mean} + h_l^i)]$
- The maximum thickness of ridged ice $h_l^{ri,max} = 2h_l^{ri,mean} - h_l^{ri,min}$
- The mean rate of thickening of ridged ice $k_l^{ri} = h_l^{ri,mean} / h_l^i$

5.5. Ridging shift

The numerical computation of the impact of mechanical redistribution on ice concentration involves:

- A normalization factor that ensures volume conservation (*aksum*)
- The removal of ice participating in deformation (including the closing of open water)
- The addition of deformed ice

For ice concentrations, the numerical procedure reads:

$$\Delta g_l^i = C^{net} \Delta t \left[- (b_l^{ri} + b_l^{ra}) + \sum_{l_2=1}^L \left(f_{l,l_2}^{ra} \frac{b_{l_2}^{ra}}{k^{ra}} + f_{l,l_2}^{ri} \frac{b_{l_2}^{ri}}{k_{l_2}^{ri}} \right) \right] \quad (5.15)$$

- C^{net} is the normalized closing rate ($|\dot{\epsilon}| \alpha^d / aksum$)
- b_l^{ri} and b_l^{ra} are the area participating into redistribution for category l
- f_{l,l_2}^{ra} and f_{l,l_2}^{ri} are the fractions of are of category l being redistributed into category l_2
- k^{ra} is the rate of thickening of rafted ice (=2)

Because of the nonlinearities involved in the integrals, the ridging procedure has to be iterated until $A^* = A^{ow} + \sum_{l=1}^L g_l^i = 1$.

5.6. Mechanical redistribution for other global ice variables

The other global ice state variables redistribution functions Ψ^X are computed based on Ψ^g for the ice age content and on Ψ^{v^i} for the remainder (ice enthalpy and salt content, snow volume and enthalpy). The general principles behind this derivation are described in Appendix A of Bitz et al. (2001). A fraction $f_s = 0.5$ (*fsnowrdg* and *fsnowrft* in *namelist_ice*) of the snow volume and enthalpy is assumed to be lost during ridging and rafting and transferred to the ocean. The contribution of the seawater trapped into the porous ridges is included in the computation of the redistribution of ice enthalpy and salt content (i.e., Ψ^{e^i} and Ψ^{M^s}). During this computation, seawater is supposed to be in thermal equilibrium with the surrounding ice blocks. Ridged ice desalination induces an implicit decrease in internal brine volume, and heat supply to the ocean, which accounts for ridge consolidation as described by Høyland (2002). The inclusion of seawater in ridges does not imply any net change in ocean salinity. The energy used to cool down the seawater trapped in porous ridges until the seawater freezing point is rejected into the ocean.

Table of contents

6.1. Solar radiation partitioning in the snow-ice system	24
6.1.1. Surface albedo	24
6.1.2. Transmission below the snow/ice surface	26
6.1.3. Attenuation and transmission below the ice/ocean interface	27
6.2. Solar radiation: framing sea ice at the ocean-atmosphere boundary	27
6.2.1. Forced mode	27
6.2.2. Coupled mode	27

Radiative transfer in SI³ currently reduces to the parameterization of solar radiation partitioning through the snow/ice/open water system, treated using a single wavelength band. This will likely be improved in future versions of the code. In this chapter, we first explain how solar radiation is partitioned in the snow-ice system, then describe how, solar radiation-wise, the snow-ice system is framed in the context of the atmosphere-ice-ocean boundary.

6.1. Solar radiation partitioning in the snow-ice system

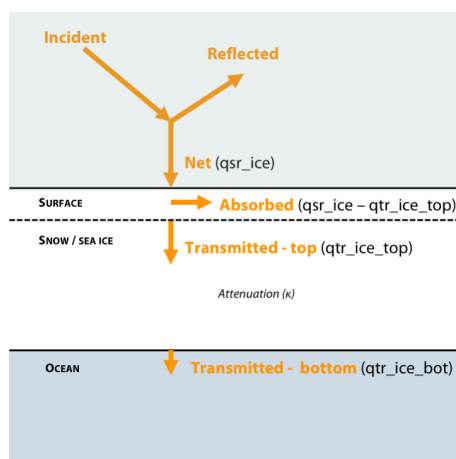


Figure 6.1.: Partitioning of solar radiation in the snow-ice system, as represented in SI³.

Solar radiation in the snow-ice system is represented following the principles of Maykut and Untersteiner (1971), see Fig.6.4, using a unique band of solar radiation. Incident solar radiation (W/m^2 , counted per unit ice area - not per grid cell area) is specified in the SBC routines and is a priori category dependent, because multiple atmosphere-surface reflexions are frequent in polar regions imply that incident radiation depends on the surface albedo and therefore surface state.

Net solar radiation $qsr_ice(i,j,l)$ is obtained by subtracting the reflected part of the incident radiation using the surface albedo $\alpha(i,j,l)$, parameterized as a function of environmental conditions.

The subsequent attenuation of solar radiation through the snow-ice system is represented assuming the presence of a highly diffusive surface scattering layer, absorbing a fraction i_o of net solar radiation, which is transformed into sensible heat, contributing to the surface energy balance.

The remainder of solar radiation, $qtr_ice_top(i,j,l)$, is transmitted below the surface and attenuates following Beer-Lambert law. The part of solar radiation that is absorbed on its path to the base of the ice is given as sensible heat to the snow/ice system, via a source term in the heat diffusion equation. The rest of solar radiation that reaches the ice base, $qtr_ice_bot(i,j,l)$, is transmitted to the ocean.

In the rest of this section, we describe how the albedo, the surface transmission parameter i_o and the attenuation of solar radiation are parameterized.

6.1.1. Surface albedo

The surface albedo determines the amount of solar radiation that is reflected by the ice surface, hence also net solar radiation. The philosophy of the parameterization of surface albedo is the following: each ice category has its own albedo value $\alpha(i,j,l)$, determined as a function of cloud fraction, ice thickness, snow depth, melt pond fraction and depth, using observation-based empirical fits.

The original Shine and Henderson-Sellers (1985) parameterization had a few inconsistencies and flaws that the revisited parameterization described hereafter fixes. In particular, the dependencies of albedos on ice thickness, snow depth and cloud fraction have been revised in the light of recent observational constraints (Brandt et al., 2005; Grenfell, 2004). In addition, the asymptotic properties of albedo are better specified and now fully consistent with oceanic values. Finally, the effect of melt ponds has been included (Lecomte et al., 2015).

The user has control on 5 reference namelist values, which describe the asymptotic values of albedo of snow and ice for dry and wet conditions, as well as the deep ponded-ice albedo. Observational surveys, in particular during SHEBA in the Arctic (Perovich, 2002) and further additional experiments (Grenfell, 2004), as well as by Brandt et al. (2005) in the Antarctic, have provided relatively strong constraints on the surface albedo. In this context, the albedo can hardly be used as the main model tuning parameter, at least outside of these observation-based bounds (see namalb for reference values).

```

!-----
&namalb      !   albedo parameters
!-----
!
! rn_alb_sdry   = 0.85      ! dry snow albedo           ! obs range (cloud-sky)
! rn_alb_smlt   = 0.75      ! melting snow albedo      : 0.85 -- 0.87
! rn_alb_idry   = 0.60      ! dry ice albedo           : 0.72 -- 0.82
! rn_alb_imlt   = 0.50      ! bare puddled ice albedo  : 0.54 -- 0.65
! rn_alb_dpnd   = 0.27      ! ponded ice albedo       : 0.49 -- 0.58
! rn_alb_dpnd   = 0.27      ! ponded ice albedo       : 0.10 -- 0.30
/
    
```

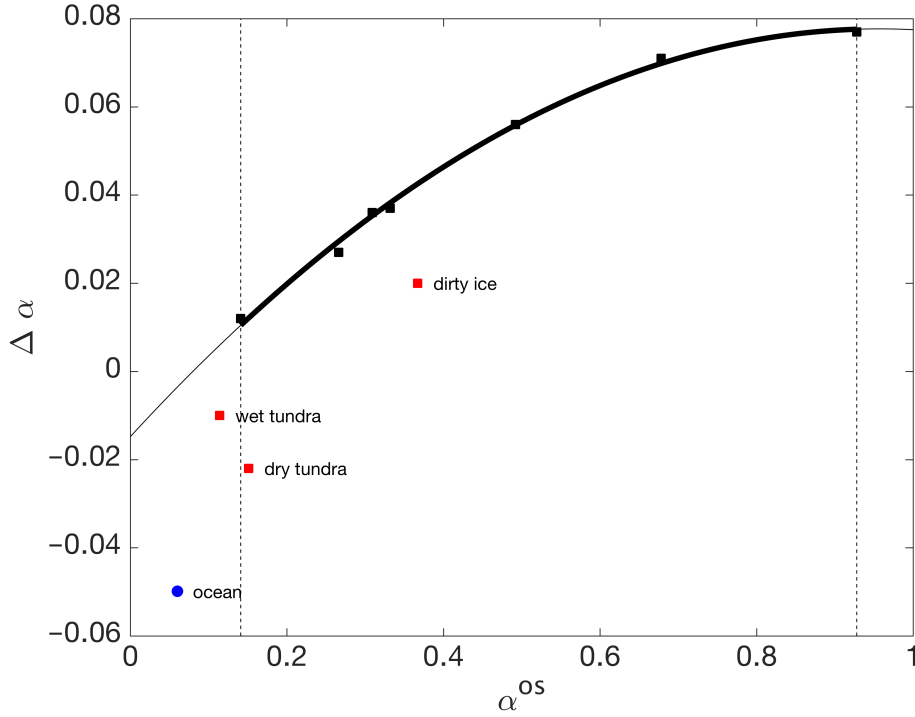


Figure 6.2.: Albedo correction $\Delta\alpha$ as a function of overcast sky (diffuse light) albedo α_{os} , from field observations (Grenfell, 2004, their Table 3) (squares) and 2nd-order fit (Eq. 6.3). Red squares represent the irrelevant data points excluded from the fit. For indication, the amplitude of the correction used in the ocean component is also depicted (blue circle).

Because the albedo is not an intrinsic optical property, it depends on the type of light (diffuse or direct), which is practically handled by weighting the clear (cs) and overcast (os) skies values by cloud fraction $c(i, j)$ (Fichefet and Maqueda, 1997):

$$\alpha(i, j, l) = [1 - c(i, j)] \cdot \alpha_{cs}(i, j, l) + c(i, j) \cdot \alpha_{os}(i, j, l). \quad (6.1)$$

For concision, we drop the spatial and category indices hereafter. Grenfell (2004) observations at Point Barrow, on the Alaskan Coast, suggest that clear and overcast sky albedos are directly related through

$$\alpha_{cs} = \alpha_{os} - \Delta\alpha(\alpha_{os}). \quad (6.2)$$

The relation between $\Delta\alpha$ and α_{os} can well be handled using a 2nd-order polynomial fit (Fig. 6.2):

$$\Delta\alpha = (-0.1010 \cdot \alpha_{os}^2 + 0.1933 \cdot \alpha_{os} - 0.0148). \quad (6.3)$$

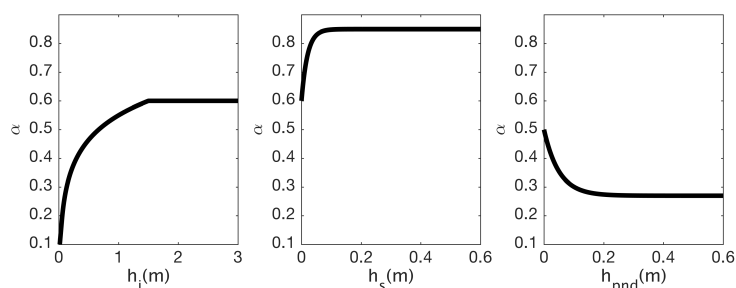
Overcast sky surface albedo is used as a reference, from which the clear-sky value is derived.

The second important parameter that controls surface albedo is surface type. In each category, we assume that three types of surfaces can coexist (bare, snow-covered and ponded ice), with respective fractions f_{ice} , f_{snw} and f_{pnd} summing to 1. Then the overcast albedo is expressed as

$$\alpha_{os}(i, j, l) = f_{ice} \cdot \alpha_{ice} + f_{snw} \cdot \alpha_{snw} + f_{pnd} \cdot \alpha_{pnd} \quad (6.4)$$

with a specific albedo value for each surface type.

The surface fractions f_{ice} , f_{snw} and f_{pnd} are currently crudely parameterized: if snow is present ($h_s > 0$), then $f_{snw} = 1$ and $f_{ice} = f_{pnd} = 0$. In the absence of snow, f_{pnd} is either specified or calculated (depending on melt pond

Figure 6.3.: Example albedo dependencies on ice thickness, snow depth and pond depth, as parameterized in SI³.

options in nampond), and $f_{ice} = 1. - f_{pnd}$. Admittedly, more refined parameterizations of f_{snow} could improve the realism of the model. Note finally that the dependence of surface albedo on the presence of melt ponds can be included or not (namelist parameter ln_pnd_alb). If the latter is set to false, f_{pnd} is always assumed zero in the albedo computations.

Works by Brandt et al. (2005) and references therein, indicate that the dependence of the albedo of bare ice on ice thickness depends is linear/logarithmic/constant from thin to thick ice. Hence, the following expressions capture the essence of their works:

$$\alpha_{ice} = \begin{cases} \alpha_{ice}^{\infty} & \text{if } h_i > 1.5, \\ \alpha_{ice}^{\infty} + (0.18 - \alpha_{ice}^{\infty}) \cdot \frac{\ln(1.5) - \ln(h_i)}{\ln(1.5) - \ln(0.05)} & \text{if } 0.05 < h_i, \leq 1.5 \\ \alpha_{oce} + (0.18 - \alpha_{oce})h_i/0.05 & \text{if } h_i < 0.05. \end{cases} \quad (6.5)$$

The thick-ice constant albedo value depends on whether the surface is dry or melting:

$$\alpha_{ice}^{\infty} = \begin{cases} \alpha_{i,dry} & \text{if } T_{su} < T_{fr} \\ \alpha_{i,mlt} & \text{if } T_{su} = T_{fr}, \end{cases} \quad (6.6)$$

values that are to be specified from the namelist.

Grenfell (2004) suggest that the dependence of surface albedo on snow depth is exponential,

$$\alpha_{snow} = \alpha_{snow}^{\infty} - (\alpha_{snow}^{\infty} - \alpha_{ice}) \cdot \exp(-h_s/h_s^{ref}), \quad (6.7)$$

where $h_s^{ref} = 0.02$ (0.03) m for dry (wet) snow. As for bare ice, the deep-snow asymptotic albedo also depends on whether the surface is dry or melting:

$$\alpha_{snow}^{\infty} = \begin{cases} \alpha_{s,dry} & \text{if } T_{su} < T_{fr} \\ \alpha_{s,mlt} & \text{if } T_{su} = T_{fr}, \end{cases} \quad (6.8)$$

values that are to be specified from the namelist.

Based on ideas developed from melt ponds on continental ice (Zuo and Oerlemans, 1996), the albedo of ponded ice was proposed to follow (Lecomte et al., 2011):

$$\alpha_{pnd} = \alpha_{dpnd} - (\alpha_{dpnd} - \alpha_{ice}) \cdot \exp(-h_{pnd}/0.05) \quad (6.9)$$

α_{dpnd} is a namelist parameter. Ebert and Curry (1993) also use such dependency for their multi-spectral albedo.

The dependencies of surface albedo on ice thickness, snow depth and pond depth are illustrated in Fig. 6.3.

6.1.2. Transmission below the snow/ice surface

The transmitted solar radiation below the surface is represented following Fichfet and Maqueda (1997) and Maykut and Untersteiner (1971):

$$qtr_ice_top(i, j, l) = i_o(i, j)qsr_ice(i, j, l), \quad (6.10)$$

where $i_o = 0$ in presence of snow, and depends on cloud fraction otherwise, based on works of Grenfell and Maykut (1977). This parameterization needs to be re-evaluated and likely updated.

6.1.3. Attenuation and transmission below the ice/ocean interface

Attenuation of solar radiation through the ice follows Beer-Lambert law. In practise, we assume that irradiance below layer k is given by

$$radtr_i(i, j, k, l) = qtr_ice_top(i, j, l) \cdot exp(-\kappa_i z), \quad (6.11)$$

where $\kappa_i = 1 \text{ m}^{-1}$ is the exponential attenuation coefficient (namelist parameter `rn_kappa_i`). Hence, at the ice base, remains below the l^{th} category a transmitted flux:

$$qtr_ice_bot(i, j, l) = qtr_ice_top(i, j, l) \cdot exp(-\kappa_i h_i). \quad (6.12)$$

6.2. Solar radiation: framing sea ice at the ocean-atmosphere boundary

How solar radiation transfer through sea ice is framed into the atmosphere-ice-ocean is nearly identical but not exactly the same in forced and coupled mode (see Fig. 6.4).

The basic principle of the computation is that the irradiant flux given to the ocean model (`qsr`) is computed as the average flux per grid cell area (`qsr_tot`) minus what is given to the sea ice ($\sum a(l)qsr_ice(l)$), plus what is transmitted below sea ice $\sum qtr_ice_bot(jl)$ (see at the base of Fig. 6.4). Such formulation ensures heat conservation by construction.

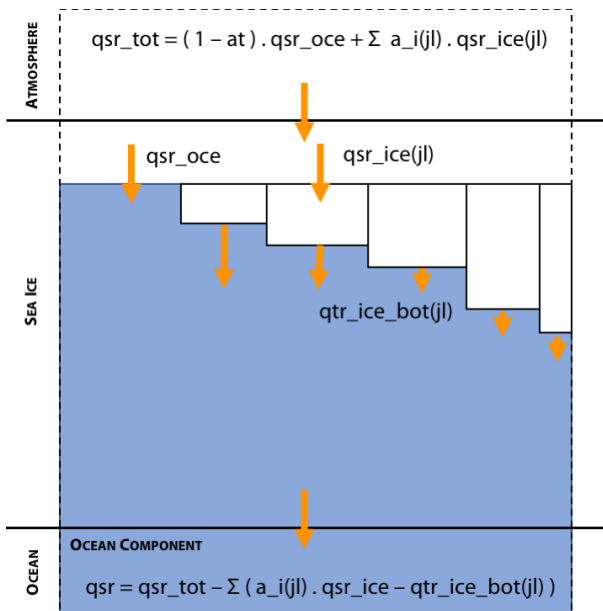


Figure 6.4.: Framing solar radiation transfer through sea ice into the atmosphere-ice-ocean context.

6.2.1. Forced mode

In forced-atmosphere mode, it is the incoming solar irradiance fluxes above the ocean and sea ice (categories) that are specified (from files) or computed (from bulk formulae), and constitute the basis of solar radiation transfer computations. Then the net solar fluxes above open water (`qsr_oce`) and ice categories (`qsr_ice`) are obtained by multiplication by $1 - \alpha$. `qsr_tot` is then diagnosed as a weighted sum of `qsr_oce` and the `qsr_ice(jl)`'s.

6.2.2. Coupled mode

In coupled-atmosphere mode, `qsr_tot` and `qsr_ice` have to be provided by the atmospheric model, whereas `qsr_oce` is diagnosed from `qsr_ice` and `qsr_tot`.

Some atmospheric models enable *tiling* and can provide solar fluxes over individual ice categories. For such atmospheric models, net solar radiation fluxes are directly useable by SI³ (`nn_fxldist = -1`). Other models cannot do tiling, being only able to provide a net solar flux above all ice categories, seen as a single surface type. For such models a first option is to give the net solar flux above sea ice identically to all sea ice categories (`nn_fxldist = 0`). Yet a better option is to redistribute

the mean solar flux above sea ice $\langle qsr_{ice} \rangle$ above categories ($nn_flxdist = 2$) using the following scaling, conserving heat by construction:

$$qsr_{ice}(jl) = \langle qsr_{ice} \rangle \frac{1 - \alpha(jl)}{1 - \langle \alpha \rangle} \quad (6.13)$$

where $\langle \alpha \rangle$ is the albedo averaged over the ice categories. Note that for testing, the flux redistributor can be emulated in forced mode ($nn_flxdist = 1$).

Table of contents

7.1. Open water and new ice formation	30
7.2. Diffusion of heat	31
7.3. Vertical growth and melt	31
7.4. Desalination	31
7.5. Remapping	31
7.6. Transport in thickness space	31
7.7. True lateral melting	31

Mass, energy and salt are intertwined for sea ice. Referred to as thermodynamics, or more exactly halo-thermodynamics. Mushy-layer theory.

7.1. Open water and new ice formation

As part of the sea ice computations, a heat budget of the uppermost oceanic level is estimated. This heat budget is used if negative to compute the production of new ice or, if positive, for bottom melting.

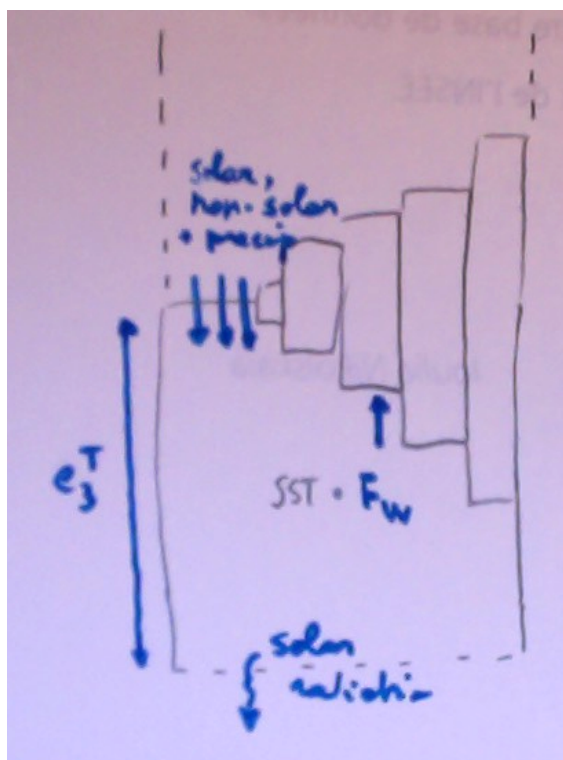


Figure 7.1.: Scheme of the estimate of the heat budget of the first ocean level.

We estimate the heat budget of the first ocean level (B^{opw}) assuming four contributions, namely:

- The absorption of a fraction f_1^{qsr} of solar radiation (given by radiative transfer component of the ocean model);
- The non-solar heat flux absorbed at the surface;
- The sensible heat content of precipitation
- The sensible heat loss to the sea ice

The estimated heat budget thus reads:

$$B^{opw} = Q^{sr}(1 - A)f_1^{qsr} + Q^{ns}(1 - A) + Q^{emp} - AF_w, \quad (7.1)$$

Hence there is no consideration of the entrainment of heat at the base of the first ocean level, or of solar radiation transmitted below the ice. The ocean-to-ice turbulent sensible heat flux is formulated following (McPhee, 1992)

$$F_w = \rho_0 c_w C_h u^* (SST - T_b) \quad (7.2)$$

where ρ_0 is the reference ocean density, c_w is the seawater specific heat, $C_h = 7.5 \times 10^{-3}$ is a heat transfer coefficient, and $u^* = \sqrt{\tau_{iw}/\rho_0}$. There are two additional conditions, the oceanic heat flux cannot be negative. Second, F_w cannot exceed the heat content of the first ocean level.

If the B^{opw} is such that the SST would decrease below the freezing point, the remainder of the heat is used to form new ice. The heat loss is converted into mass through [1.16], giving by multiplication by density a volume of new ice v_0 . The thickness h_0 of the new ice grown during a sea ice time step depends on unresolved small currents and waves

and is prescribed. The fraction $a_0 = v_0/h_0$ is computed accordingly. The salinity of this new ice S_0 is given by the S-h empirical relationship of Kovacs (1996). The temperature assumed for this new ice is the local freezing point.

If there is ice in the grid cell and that the B^{opw} is positive, it is directly given attributed to the heat available for bottom melting. This argument follows from Maykut and McPhee (1995), who found that most of solar heat absorbed in the surface waters is converted into melting. In practise, this means that the SST cannot go above freezing as long ice is present.

The heat loss used for ice formation, heat gain used to melt ice and the sensible heat given by the ocean to the ice, are all removed from the non-solar heat flux transmitted to the ocean.

Because ice dynamics are not able to maintain the small amount of open water that is observed, a maximum ice fraction ($amax, < 1$) is prescribed.

7.2. Diffusion of heat

7.3. Vertical growth and melt

7.4. Desalination

7.5. Remapping

7.6. Transport in thickness space

7.7. True lateral melting

Ice-atmosphere and ice-ocean interfaces

Table of contents

8.1. Ice-ocean interface	33
8.2. Ice-atmosphere interface	33

In this chapter, we give information on the representation of interfaces.

8.1. Ice-ocean interface

8.2. Ice-atmosphere interface

Drags?

Blowing snow parameter

Flux redistributor

Jules coupling

Table of contents

9.1. SIMIP diagnostics	35
9.1.1. Missing SIMIP fields	35
9.1.2. Links	35
9.2. Conservation checks	35

9.1. SIMIP diagnostics

The SIMIP protocol (Notz et al., 2016) was designed for CMIP6, to standardize sea ice model outputs in climate simulations. We tried to follow the data request as closely as possible. Outputs are in most cases directly managed with XIOS2 in `limwri.F90`, but not always. In the code, output fields keep their native LIM reference name.

A corresponding entry exists in `field_def_nemo-lim.xml`, where fields are given their SIMIP specifications (standard name, long name, units). At the end of the file the fields are gathered in the field groups `SIday_fields`, `SImon_fields` and `SImon_scalar` for separation of the daily (SIday) and monthly (SImon) requests.

In `file_def_nemo-lim.xml`, the daily, monthly and scalar output files are created.

In the reference xml files, the largest possible SIMIP-based diagnostics with LIM are distributed among the field groups. If some fields are to be discarded, the best way to do so is to remove them from the field groups in `field_def_nemo-lim.xml`.

9.1.1. Missing SIMIP fields

About 90% of the SIMIP fields can be output. Below is the list of the missing fields and why they are missing.

1. Fields that are not part of the sea ice representation in LIM3.6

- `sisnconc` (snow area fraction), `siitdsnconc` (snow area fractions in thickness categories);
- `simpconc` (meltpond area fraction), `simplmass` (melt pond mass per area), `simprefrozen` (thickness of refrozen ice on ponds);
- `sirdgconc` (ridged ice area fraction), `sirdgmass` (ridged ice thickness);
- `sidmasslat` (lateral sea ice melt rate);
- `sndmasswindrif` (snow mass change through wind drift of snow);

2. Fields which value is trivial

- `sipr` (rainfall over sea ice): all rain falls in open water;
- `sidragtop` (atmospheric drag over sea ice): namelist parameter;
- `sidragbot` (oceanic drag over sea ice): namelist parameter

3. Fields that belong to the atmospheric component

- `siflswdtop`, `siflswutop`, `siflswdbot`, `sifllwdtop`, `sifllwutop`, `siflsenstop`, `sifflatstop` (surface energy budget components)

Ice thickness and snow depth were masked below 5% ice concentration, because below this value, they become meaninglessly large in LIM. This is notably because of the Prather advection scheme. We hope to fix these issues for our next release. For similar reasons, the ice age is masked below 15% concentration.

Fluxes through straits and passages were not directly implemented. Instead, ice mass, snow mass, and ice area transports were implemented as 2D arrays, for x- and y- directions. A python script is available to derive the fluxes through straits and passages from full 2D arrays for ORCA2 and eORCA1 grids.

9.1.2. Links

- Paper of Notz et al;
- SIMIP CMIP6 data request page;
- SIMIP description on CliC website.

9.2. Conservation checks

Table of contents

10.1. Enhanced conduction	37
10.2. Virtual thin ice melting (fake lateral melting)	37
10.3. Ridging and rafting with a single ice category	37

10.1. Enhanced conduction

10.2. Virtual thin ice melting (fake lateral melting)

10.3. Ridging and rafting with a single ice category

Table of contents

Table of contents

getting started, namelists, configs, ...



Table of contents

A.1. Introduction	43
A.2. Overview and general conventions	43
A.3. Architecture	44
A.4. Style rules	44
A.4.1. Argument list format	44
A.4.2. Array syntax	44
A.4.3. Case	44
A.4.4. Comments	45
A.4.5. Continuation lines	45
A.4.6. Declaration of arguments and local variables	45
A.4.7. F90 Standard	45
A.4.8. Free-Form Source	45
A.4.9. Indentation	46
A.4.10. Loops	46
A.4.11. Naming Conventions: files	46
A.4.12. Naming Conventions: modules	46
A.4.13. Naming Conventions: variables	46
A.4.14. Operators	46
A.4.15. Pre processor	46
A.5. Content rules	47
A.5.1. Configurations	47
A.5.2. Constants	47
A.5.3. Declaration for variables and constants	47
A.5.4. Headers	48
A.5.5. Interface blocks	48
A.5.6. I/O Error Conditions	48
A.5.7. PRINT - ASCII output files	48
A.5.8. Precision	49
A.5.9. Structures	49
A.6. Packages coding rules	49
A.6.1. Bounds checking	49
A.6.2. Communication	49
A.6.3. Error conditions	49
A.6.4. Memory management	50
A.6.5. Optimisation	51
A.6.6. Package attribute: PRIVATE , PUBLIC , USE , ONLY	51
A.6.7. Parallelism using MPI	51
A.7. Features to be avoided	51

A "model life" is more than ten years. Its software, composed of a few hundred modules, is used by many people who are scientists or students and do not necessarily know every aspect of computing very well. Moreover, a well thought-out program is easier to read and understand, less difficult to modify, produces fewer bugs and is easier to maintain. Therefore, it is essential that the model development follows some rules:

- well planned and designed
- well written
- well documented (both on- and off-line)
- maintainable
- easily portable
- flexible.

To satisfy part of these aims, *NEMO* is written with a coding standard which is close to the ECMWF rules, named DOCTOR (?). These rules present some advantages like:

- to provide a well presented program
- to use rules for variable names which allow recognition of their type (integer, real, parameter, local or shared variables, etc.).

This facilitates both the understanding and the debugging of an algorithm.

A.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.

A.2. Overview and general conventions

NEMO has 3 major components: ocean dynamics (`./src/OCE`), sea-ice (`./src/ICE`) and marine biogeochemistry (`./src/MBG`). In each directory, one will find some FORTRAN files and/or subdirectories, one per functionality of the code: `./src/OCE/BDY` (boundaries), `./src/OCE/DIA` (diagnostics), `./src/OCE/DOM` (domain), `./src/OCE/DYN` (dynamics), `./src/OCE/LDF` (lateral diffusion), etc...

All name are chosen to be as self-explanatory as possible, in English, all prefixes are 3 digits.

English is used for all variables names, comments, and documentation.

Physical units are MKS. The only exception to this is the temperature, which is expressed in degrees Celsius, except in bulk formulae and part of SI³ sea-ice model where it is in Kelvin. See `./src/OCE/DOM/physcst.F90` files for conversions.

*UCAR conventions

A.3. Architecture

Within each directory, organisation of files is driven by orthogonality, *i.e.* one functionality of the code is intended to be in one and only one directory, and one module and all its related routines are in one file. The functional modules are:

- SBC surface module
- IOM management of the I/O
- NST interface to AGRIF (nesting model) for dynamics and biogeochemistry
- OBC, BDY management of structured and unstructured open boundaries
- C1D 1D (vertical) configuration for dynamics, sea-ice and biogeochemistry
- OFF off-line module: passive tracer or biogeochemistry alone
- ...

For example, the file *domain.F90* contains the module `domain` and all the subroutines related to this module (`dom_init`, `dom_nam`, `dom_ctl`).

A.4. Style rules

A.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 )
```

A.4.2. Array syntax

Except for long loops (see below), array notation should be used if possible. To improve readability the array shape must be shown in brackets, *e.g.*:

```
onedarraya(:) = onedarrayb(:) + onedarrayc(:)
twodarray(:, :) = scalar * another twodarray(:, :)
```

When accessing sections of arrays, for example in finite difference equations, do so by using the triplet notation on the full array, *e.g.*:

```
twodarray(:, 2:len2) = scalar &
& * ( twodarray2(:, 1:len2-1) &
& - twodarray2(:, 2:len2) )
```

For long, complicated loops, explicitly indexed loops should be preferred. In general when using this syntax, the order of the loops indices should reflect the following scheme (for best usage of data locality):

```
DO jk = 1, jpk
  DO jj = 1, jpj
    DO ji = 1, jpi
      threedarray(ji, jj, jk) = ...
    END DO
  END DO
END DO
```

A.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.

A.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

```

A.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).

A.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).[†]

A.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, `zdfake`, where "zdf" stands for vertical diffusion, and "ake" 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.

A.4.13. Naming Conventions: variables

All variable should be named as explicitly as possible in English. The naming convention concerns prefix letters of these name, in order to identify the variable type and status.

Never use a FORTRAN keyword as a routine or variable name.

The table below lists the starting letter(s) to be used for variable naming, depending on their type and status:

A.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 >`

A.4.15. Pre processor

Where the use of a language pre-processor is required, it will be the C pre-processor (`cpp`).

The `cpp` key is the main feature used, allowing to ignore some useless parts of the code at compilation step.

The advantage is to reduce the memory use; the drawback is that compilation of this part of the code isn't checked.

The `cpp` key feature should only be used for a few limited options, if it reduces the memory usage. In all cases, a logical variable and a FORTRAN **IF** should be preferred. When using a `cpp` key `key_optionname`, a corresponding logical variable `lk_optionname` should be declared to allow FORTRAN **IF** tests in the code and a FORTRAN module with the same name (*i.e.* `optionname.F90`) should be defined. This module is the only place where a '#if defined' command appears, selecting either the whole FORTRAN code or a dummy module. For example, the TKE vertical physics, the module name is `zdfake.F90`, the CPP key is `key_zdfake` and the associated logical is `lk_zdfake`.

The following syntax:

[†]For example, if routine A "USE"s module B, then "make" must be told of the dependency relation which requires B to be compiled before A. If one can assume that module B resides in file B.o, building a tool to generate this dependency rule (*e.g.* A.o: B.o) is quite simple. Put another way, it is difficult (to say nothing of CPU-intensive) to search an entire source tree to find the file in which module B resides for each routine or module which "USE"s B.

Type / Status	integer	real	logical	character	double precision	complex
public or module variable	m n <i>but not nn_</i>	a b e f g h o q to x <i>but not fs rn_</i>	l <i>but not lp ld ll ln_</i>	c <i>but not cp cd cl cn_</i>	d <i>but not dp dd dl dn_</i>	y <i>but not yp yd yl</i>
dummy argument	k <i>but not kf</i>	p <i>but not pp pf</i>	ld	cd	dd	yd
local variable	i	z	ll	cl	cd	yl
loop control	j <i>but not jp</i>					
parameter	jp	pp	lp	cp	dp	yp
namelist	nn_	rn_	ln_	cn_	dn_	
CPP macro	kf	sf				

```
#if defined key_optionname
!! Part of code conditionally compiled if cpp key key_optionname is active
#endif
```

Is to be used rather than the #ifdef abbreviate form since it may have conflicts with some Unix scripts. Tests on cpp keys included in *NEMO* at compilation step:

- The CPP keys used are compared to the previous list of cpp keys (the compilation will stop if trying to specify a non-existing key)
- If a change occurs in the CPP keys used for a given experiment, the whole compilation phase is done again.

A.5. Content rules

A.5.1. Configurations

The configuration defines the domain and the grid on which *NEMO* is running. It may be useful to associate a CPP key and some variables to a given configuration, although the part of the code changed under each of those keys should be minimized. As an example, the "ORCA2" configuration (global ocean, 2 degrees grid size) is associated with the cpp key `key_orca2` for which

```
cp_cfg = "orca"
jp_cfg = 2
```

A.5.2. Constants

Physical constants (*e.g.* π , gas constants) must never be hard-wired into the executable portion of a code. Instead, a mnemonically named variable or parameter should be set to the appropriate value, in the setup routine for the package. We realize that many parameterizations rely on empirically derived constants or fudge factors, which are not easy to name. In these cases it is not forbidden to leave such factors coded as "magic numbers" buried in executable code, but comments should be included referring to the source of the empirical formula. Hard-coded numbers should never be passed through argument lists.

A.5.3. Declaration for variables and constants

Rules

Variables used as constants should be declared with attribute **PARAMETER** and used always without copying to local variables, in order to prevent from using different values for the same constant or changing it accidentally.

- Usage of the **DIMENSION** statement or attribute is required in declaration statements

- The “:” notation is quite useful to show that this program unit declaration part is written in standard FORTRAN syntax, even if there are no attributes to clarify the declaration section. Always use the notation <blank>::<three blanks> to improve readability.
- Declare the length of a character variable using the **CHARACTER** (len=xxx) syntax[‡]
- For all global data (in contrast to module data, that is all data that can be access by other module) must be accompanied with a comment field on the same line[§]. For example:

```
REAL(wp), DIMENSION(jpi,jpj,jpk) :: ua ! i-horizontal velocity (m/s)
```

Implicit None

All subroutines and functions will include an **IMPLICIT NONE** statement. Thus all variables must be explicitly typed. It also allows the compiler to detect typographical errors in variable names. For modules, one **IMPLICIT NONE** statement in the modules definition section is needed. This also removes the need to have **IMPLICIT NONE** statements in any routines that are **CONTAINS**'ed in the module. Improper data initialisation is another common source of errors[¶]. To avoid problems, initialise variables as close as possible to where they are first used.

Attributes

PRIVATE / PUBLIC: All resources of a module are **PUBLIC** by default. A reason to store multiple routines and their data in a single module is that the scope of the data defined in the module can be limited to the routines which are in the same module. This is accomplished with the **PRIVATE** attribute.

INTENT: All dummy arguments of a routine must include the **INTENT** clause in their declaration in order to improve control of variables in routine calls.

A.5.4. Headers

Prologues are not used in *NEMO* for now, although it may become an interesting tool in combination with ProTeX auto documentation script in the future. Rules to code the headers and layout of a module or a routine are illustrated in the example module available with the code: `./src/OCE/module_example`

A.5.5. Interface blocks

Explicit interface blocks are required between routines if optional or keyword arguments are to be used. They also allow the compiler to check that the type, shape and number of arguments specified in the **CALL** are the same as those specified in the subprogram itself. FORTRAN 95 compilers can automatically provide explicit interface blocks for routines contained in a module.

A.5.6. I/O Error Conditions

I/O statements which need to check an error condition will use the `iostat=<integer variable>` construct instead of the outmoded `end=` and `err=`.

Note that a 0 value means success, a positive value means an error has occurred, and a negative value means the end of record or end of file was encountered.

A.5.7. PRINT - ASCII output files

Output listing and errors are directed to `numout` logical unit =6 and produces a file called *ocean.output* (use `ln_prt` to have one output per process in MPP). Logical `lwp` variable allows for less verbose outputs. To output an error from a routine, one can use the following template:

```
IF( nstop /= 0 .AND. lwp ) THEN ! error print
  WRITE(numout,cform_err)
  WRITE(numout,*) nstop, ' error have been found'
ENDIF
```

[‡]The len specifier is important because it is possible to have several kinds for characters (e.g. Unicode using two bytes per character, or there might be a different kind for Japanese e.g. NEC).

[§]This allows a easy research of where and how a variable is declared using the unix command: “`grep var *90 | grep !:`”.

[¶]A variable could contain an initial value you did not expect. This can happen for several reasons, e.g. the variable has never been assigned a value, its value is outdated, memory has been allocated for a pointer but you have forgotten to initialise the variable pointed to.

A.5.8. Precision

Parameterizations should not rely on vendor-supplied flags to supply a default floating point precision or integer size. The F95 `KIND` feature should be used instead. In order to improve portability between 32 and 64 bit platforms, it is necessary to make use of kinds by using a specific module `./src/OCE/par_kind.F90` declaring the "kind definitions" to obtain the required numerical precision and range as well as the size of `INTEGER`. It should be noted that numerical constants need to have a suffix of `_kindvalue` to have the according size.

Thus `wp` being the "working precision" as declared in `./src/OCE/par_kind.F90`, declaring real array `zpc` will take the form:

```
REAL(wp), DIMENSION(jpi,jpj,jpk) :: zpc      ! power consumption
```

A.5.9. Structures

The `TYPE` structure allowing to declare some variables is more often used in *NEMO*, especially in the modules dealing with reading fields, or interfaces. For example:

```
! Definition of a tracer as a structure
TYPE PTRACER
  CHARACTER(len = 20) :: sname ! short name
  CHARACTER(len = 80) :: lname ! long name
  CHARACTER(len = 20) :: unit  ! unit
  LOGICAL             :: lini  ! read in a file or not
  LOGICAL             :: lsav  ! ouput the tracer or not
END TYPE PTRACER

TYPE(PTRACER), DIMENSION(jptr) :: tracer
```

Missing rule on structure name??

A.6. Packages coding rules

A.6.1. Bounds checking

NEMO is able to run when an array bounds checking option is enabled (provided the `cpp` key `key_vectopt_loop` is not defined).

Thus, constructs of the following form are disallowed:

```
REAL(wp) :: arr(1)
```

where "arr" is an input argument into which the user wishes to index beyond 1. Use of the (*) construct in array dimensioning is forbidden also because it effectively disables array bounds checking.

A.6.2. Communication

A package should refer only to its own modules and subprograms and to those intrinsic functions included in the Fortran standard.

All communication with the package will be through the argument list or namelist input. ^{||}

A.6.3. Error conditions

When an error condition occurs inside a package, a message describing what went wrong will be printed (see `PRINT - ASCII` output files). The name of the routine in which the error occurred must be included. It is acceptable to terminate execution within a package, but the developer may instead wish to return an error flag through the argument list, see *stpctl.F90*.

^{||} The point behind this rule is that packages should not have to know details of the surrounding model data structures, or the names of variables outside of the package. A notable exception to this rule is model resolution parameters. The reason for the exception is to allow compile-time array sizing inside the package. This is often important for efficiency.


```

    RETURN
  END IF
  !
  zwrk1 => wrk_3d_5(1:10,1:10,1:10)
  ...
END SUBROUTINE sub

```

Here, instead of “use associating” the variable `zwrk1` with the array `wrk_3d_5` (as in the first example), it is explicitly declared as a pointer to a 3D array. It is then associated with a sub-array of `wrk_3d_5` once the call to `wrk_in_use()` has completed successfully. Note that in F95 (to which *NEMO* conforms) it is not possible for either the upper or lower array bounds of the pointer object to differ from those of the target array.

In addition to the **REAL** (`KIND = wp`) workspace arrays, `wrk_nemo.F90` also contains 2D integer arrays and 2D REAL arrays with extent (`jpi`, `jpj`), *i.e.* `xz`. The utility routines for the integer workspaces are `iwrk_in_use()` and `iwrk_not_released()` while those for the `xz` workspaces are `wrk_in_use_xz()` and `wrk_not_released_xz()`.

Should a call to one of the `wrk_in_use()` family of utilities fail, an error message is printed along with a table showing which of the workspace arrays are currently in use. This should enable the developer to choose alternatives for use in the subroutine being worked on.

When compiling *NEMO* for production runs, the calls to `wrk_in_use()` / `wrk_not_released()` can be reduced to stubs that just return `.false.` by setting the `cpp` key `key_no_workspace_check`. These stubs may then be inlined (and thus effectively removed altogether) by setting appropriate compiler flags (*e.g.* “`-finline`” for the Intel compiler or “`-Q`” for the IBM compiler).

A.6.5. Optimisation

Considering the new computer architecture, optimisation cannot be considered independently from the computer type. In *NEMO*, portability is a priority, before any too specific optimisation.

Some tools are available to help: for vector computers, `key_vectopt_loop` allows to unroll a loop

A.6.6. Package attribute: **PRIVATE**, **PUBLIC**, **USE**, **ONLY**

Module variables and routines should be encapsulated by using the **PRIVATE** attribute. What shall be used outside the module can be declared **PUBLIC** instead. Use **USE** with the **ONLY** attribute to specify which of the variables, type definitions etc... defined in a module are to be made available to the using routine.

A.6.7. Parallelism using MPI

NEMO is written in order to be able to run on one processor, or on one or more using MPI (*i.e.* activating the `cpp` key `key_mpp_mpi`). The domain decomposition divides the global domain in cubes (see *NEMO* reference manual). Whilst coding a new development, the MPI compatibility has to be taken in account (see `./src/LBC/lib_mpp.F90`) and should be tested. By default, the `x-z` part of the decomposition is chosen to be as square as possible. However, this may be overridden by specifying the number of sub-domains in latitude and longitude in the `nammpp` section of the namelist file.

A.7. Features to be avoided

The code must follow the current standards of FORTRAN and ANSI C. In particular, the code should not produce any WARNING at compiling phase, so that users can be easily alerted of potential bugs when some appear in their new developments. Below is a list of features to avoid:

- **COMMON** block (use the declaration part of **MODULE** instead)
- **EQUIVALENCE** (use **POINTER** or derived data type instead to form data structure)
- Assigned and computed **GOTO** (use the **CASE** construct instead)
- Arithmetic **IF** statement (use the block **IF**, **ELSE**, **ELSEIF**, **ENDIF** or **SELECT CASE** construct instead)
- Labelled **DO** construct (use unlabelled **END DO** instead)
- **FORMAT** statement (use character parameters or explicit format- specifiers inside the **READ** or **WRITE** statement instead)
- **GOTO** and **CONTINUE** statements (use **IF**, **CASE**, **DO WHILE**, **EXIT** or **CYCLE** statements or a contained ?)

- **PAUSE**
- **ENTRY** statement: a sub-program must only have one entry point.
- **RETURN** is obsolete and so not necessary at the end of program units
- **FUNCTION** statement
- Avoid functions with side effects. **
- **DATA** and **BLOCK DATA** (use initialisers)

**First, the code is easier to understand, if you can rely on the rule that functions don't change their arguments. Second, some compilers generate more efficient code for PURE functions (in FORTRAN 95 there are the attributes PURE and ELEMENTAL), because they can store the arguments in different places. This is especially important on massive parallel and as well on vector machines.

Bibliography

- Assur, A., 1958: Composition of sea ice and its tensile strength. *Arctic Sea Ice*, **598**, 106–138.
- Babko, O., 2002: Role of rafting in the mechanical redistribution of sea ice thickness. *Journal of Geophysical Research*, **107** (C8), [doi](#), URL.
- Bitz, C. M., M. M. Holland, A. J. Weaver, and M. Eby, 2001: Simulating the ice-thickness distribution in a coupled climate model. *Journal of Geophysical Research: Oceans*, **106** (C2), 24412463, [doi](#), URL.
- Bitz, C. M. and W. H. Lipscomb, 1999: An energy-conserving thermodynamic model of sea ice. *Journal of Geophysical Research: Oceans*, **104** (C7), 15 66915 677, [doi](#), URL.
- Bouillon, S., T. Fichefet, V. Legat, and G. Madec, 2013: The elasticviscousplastic method revisited. *Ocean Modelling*, **71**, 212, [doi](#), URL.
- Brandt, R. E., S. G. Warren, A. P. Worby, and T. C. Grenfell, 2005: Surface albedo of the antarctic sea ice zone. *Journal of Climate*, **18** (17), 36063622, [doi](#), URL.
- Ebert, E. E. and J. A. Curry, 1993: An intermediate one-dimensional thermodynamic sea ice model for investigating ice-atmosphere interactions. *Journal of Geophysical Research*, **98** (C6), 10 085, [doi](#), URL.
- Fichefet, T. and M. A. M. Maqueda, 1997: Sensitivity of a global sea ice model to the treatment of ice thermodynamics and dynamics. *Journal of Geophysical Research: Oceans*, **102** (C6), 12 60912 646, [doi](#), URL.
- Flato, G. M. and W. D. Hibler, 1995: Ridging and strength in modeling the thickness distribution of arctic sea ice. *Journal of Geophysical Research*, **100** (C9), 18 611, [doi](#), URL.
- Girard, L., J. Weiss, J. M. Molines, B. Barnier, and S. Bouillon, 2009: Evaluation of high-resolution sea ice models on the basis of statistical and scaling properties of arctic sea ice drift and deformation. *Journal of Geophysical Research*, **114** (C8), [doi](#), URL.
- Grenfell, T. C., 2004: Seasonal and spatial evolution of albedo in a snow-ice-land-ocean environment. *Journal of Geophysical Research*, **109** (C1), [doi](#), URL.
- Grenfell, T. C. and G. A. Maykut, 1977: The optical properties of ice and snow in the arctic basin. *Journal of Glaciology*, **18** (80), 445463, [doi](#), URL.
- Haapala, J., 2000: On the modelling of ice-thickness redistribution. *Journal of Glaciology*, **46** (154), 427437, [doi](#), URL.
- Hibler, W. D., 1979: A dynamic thermodynamic sea ice model. *Journal of Physical Oceanography*, **9** (4), 815846, [doi](#), URL.
- , 1980: Modeling a variable thickness sea ice cover. *Monthly Weather Review*, **108** (12), 19431973, [doi](#), URL.
- Høyland, K. V., 2002: Consolidation of first-year sea ice ridges. *Journal of Geophysical Research*, **107** (C6), [doi](#), URL.
- IOC, SCOR and IAPSO, 2010: *The international thermodynamic equation of seawater - 2010: Calculation and use of thermodynamic properties*. Intergovernmental Oceanographic Commission, Manuals and Guides No. 56, UNESCO (English), URL.
- Kovacs, A., 1996: Sea ice. part 1. bulk salinity versus ice floe thickness,. Tech. rep., [doi](#), URL.
- Kreyscher, M., M. Harder, P. Lemke, and G. M. Flato, 2000: Results of the sea ice model intercomparison project: Evaluation of sea ice rheology schemes for use in climate simulations. *Journal of Geophysical Research: Oceans*, **105** (C5), 11 29911 320, [doi](#), URL.
- Lecomte, O., T. Fichefet, D. Flocco, D. Schroeder, and M. Vancoppenolle, 2015: Interactions between wind-blown snow redistribution and melt ponds in a coupled oceansea ice model. *Ocean Modelling*, **87**, 6780, [doi](#), URL.
- Lecomte, O., T. Fichefet, M. Vancoppenolle, and M. Nicolaus, 2011: A new snow thermodynamic scheme for large-scale sea-ice models. *Annals of Glaciology*, **52** (57), 337346, [doi](#), URL.
- Leppäranta, M., 2011: Drift ice material. *The Drift of Sea Ice*, 1163, [doi](#), URL.
- Leppäranta, M., M. Lensu, P. Kosloff, and B. Veitch, 1995: The life story of a first-year sea ice ridge. *Cold Regions Science and Technology*, **23** (3), 279290, [doi](#), URL.
- Lipscomb, W. H., 2001: Remapping the thickness distribution in sea ice models. *Journal of Geophysical Research: Oceans*, **106** (C7), 13 98914 000, [doi](#), URL.
- Lipscomb, W. H., E. C. Hunke, W. Maslowski, and J. Jakacki, 2007: Ridging, strength, and stability in high-resolution sea ice models. *Journal of Geophysical Research*, **112** (C3), [doi](#), URL.
- Massonnet, F., A. Barthélémy, K. Worou, T. Fichefet, M. Vancoppenolle, and C. Rousset, 2018: Insights on the discretization of the ice thickness distribution in large-scale sea ice models. *submitted*.

- Maykut, G. A., 1986: The surface heat and mass balance. *The Geophysics of Sea Ice*, 395463, [doi](#), [URL](#).
- Maykut, G. A. and M. G. McPhee, 1995: Solar heating of the arctic mixed layer. *Journal of Geophysical Research*, **100** (C12), 24 691, [doi](#), [URL](#).
- Maykut, G. A. and A. S. Thorndike, 1973: An approach to coupling the dynamics and thermodynamics of arctic sea ice. *AIDJEX Bulletin*, **21**, 23–29.
- Maykut, G. A. and N. Untersteiner, 1971: Some results from a time-dependent thermodynamic model of sea ice. *Journal of Geophysical Research*, **76** (6), 15501575, [doi](#), [URL](#).
- McPhee, M. G., 1992: Turbulent heat flux in the upper ocean under sea ice. *Journal of Geophysical Research*, **97** (C4), 5365, [doi](#), [URL](#).
- Mellor, G. L. and L. Kantha, 1989: An ice-ocean coupled model. *Journal of Geophysical Research*, **94** (C8), 10 937, [doi](#), [URL](#).
- Notz, D., A. Jahn, M. Holland, E. Hunke, F. Massonnet, J. Stroeve, B. Tremblay, and M. Vancoppenolle, 2016: The cmip6 sea-ice model intercomparison project (simip): understanding sea ice through climate-model simulations. *Geoscientific Model Development*, **9** (9), 34273446, [doi](#), [URL](#).
- Parmeter, R. R., 1975: A model of simple rafting in sea ice. *Journal of Geophysical Research*, **80** (15), 19481952, [doi](#), [URL](#).
- Perovich, D. K., 2002: Seasonal evolution of the albedo of multi-year arctic sea ice. *Journal of Geophysical Research*, **107** (C10), [doi](#), [URL](#).
- Prather, M. J., 1986: Numerical advection by conservation of second-order moments. *Journal of Geophysical Research*, **91** (D6), 6671, [doi](#), [URL](#).
- Pringle, D. J., H. Eicken, H. J. Trodahl, and L. G. E. Backstrom, 2007: Thermal conductivity of landfast antarctic and arctic sea ice. *Journal of Geophysical Research*, **112** (C4), [doi](#), [URL](#).
- Schmidt, G. A., C. M. Bitz, U. Mikolajewicz, and L.-B. Tremblay, 2004: Iceocean boundary conditions for coupled models. *Ocean Modelling*, **7** (1-2), 5974, [doi](#), [URL](#).
- Shine, K. P. and A. Henderson-Sellers, 1985: The sensitivity of a thermodynamic sea ice model to changes in surface albedo parameterization. *Journal of Geophysical Research*, **90** (D1), 2243, [doi](#), [URL](#).
- Thorndike, A. S., D. A. Rothrock, G. A. Maykut, and R. Colony, 1975: The thickness distribution of sea ice. *Journal of Geophysical Research*, **80** (33), 45014513, [doi](#), [URL](#).
- Tuhkuri, J., 2002: Laboratory tests on ridging and rafting of ice sheets. *Journal of Geophysical Research*, **107** (C9), [doi](#), [URL](#).
- Vancoppenolle, M., T. Fichefet, and H. Goosse, 2009: Simulating the mass balance and salinity of arctic and antarctic sea ice. 2. importance of sea ice salinity variations. *Ocean Modelling*, **27** (1-2), 5469, [doi](#), [URL](#).
- Weiss, J., 2013: Drift, deformation, and fracture of sea ice. *SpringerBriefs in Earth Sciences*, [doi](#), [URL](#).
- Worster, M. G., 1992: The dynamics of mushy layers. *Interactive Dynamics of Convection and Solidification*, 113138, [doi](#), [URL](#).
- Zhang, J. and D. Rothrock, 2001: A thickness and enthalpy distribution sea-ice model. *Journal of Physical Oceanography*, **31** (10), 29863001, [doi](#), [URL](#).
- Zuo, Z. and J. Oerlemans, 1996: Modelling albedo and specific balance of the greenland ice sheet: calculations for the søndre strømfjord transect. *Journal of Glaciology*, **42** (141), 305317, [doi](#), [URL](#).