CICE-Consortium Icepack Documentation

Release 0.0.1

Alice DuVivier

Dec 21, 2017

CONTENTS

CHAPTER

ONE

TABLE OF CONTENTS:

1.1 Introduction - Icepack

The column physics of the sea ice model CICE, "Icepack", is maintained by the CICE Consortium. This code includes several options for simulating sea ice thermodynamics, mechanical redistribution (ridging) and associated area and thickness changes. In addition, the model supports a number of tracers, including thickness, enthalpy, ice age, first-year ice area, deformed ice area and volume, melt ponds, and biogeochemistry.

Icepack is implemented in CICE as a git submodule. Icepack basically consists of three independent parts, the column physics code, the icepack driver that supports stand-alone testing of the column physics code, and the icepack scripts that build and test the Icepack model. The intent of Icepack is to provide the column physics model as a separate library for use in other models such as CICE. Development and testing of CICE and Icepack may be done together, but the repositories are independent.

This document uses the following text conventions: Variable names used in the code are `typewritten`. Subroutine names are given in *italic*. File and directory names are in **boldface**. Code and scripts are contained in a literal box or `typewritten`. A comprehensive *Index of primary variables and parameters*, including glossary of symbols with many of their values, appears at the end of this guide.

1.1.1 Quick Start

Download the model from the CICE-Consortium repository, https://github.com/CICE-Consortium/Icepack

Instructions for working in github with Icepack (and CICE) can be found in the CICE Git and Workflow Guide.

From your main Icepack directory, execute:

```
./icepack.create.case -c ~/mycase1 -m testmachine
cd ~/mycase1
./icepack.build
./icepack.submit
```

Note that testmachine is a generic machine name included with the icepack scripts. The local machine name will have to be substituted for testmachine and currently, there are working ports for several different machines. However, it may be necessary to port the model to a new machine. See *Porting* for more information about how to port and *Scripts* for more information about how to use the icepack.create.case script.

1.1.2 Major Icepack updates since CICE v5.1.2

This model release is Icepack version 1.0.

The column physics code was separated from CICE version 5.1.2 by removing all references to the horizontal grid and other infrastructural CICE elements (e.g. MPI tasks, calendar).

- A simplified driver was developed for Icepack, for testing purposes.
- Additional tests for the column physics are now available.
- This release includes the full vertical biogeochemistry code.

1.1.3 Acknowledgements

This work has been completed through the CICE Consortium and its members with funding through the Department of Energy, Department of Defense (Navy), Department of Commerce (NOAA), National Science Foundation and Environment and Climate Change Canada. Special thanks are due to the following people:

- Elizabeth Hunke, Nicole Jeffery, Adrian Turner and Chris Newman at Los Alamos National Laboratory
- David Bailey, Alice DuVivier and Marika Holland at the National Center for Atmospheric Research
- Rick Allard and Matt Turner at the Naval Research Laboratory, Stennis Space Center,
- Andrew Roberts of the Naval Postgraduate School,
- Jean-Francois Lemieux and Frederic Dupont of Environment and Climate Change Canada,
- Tony Craig and his supporters at the National Center for Atmospheric Research, the Naval Postgraduate School…CHECK,
- Cecilia Bitz at the University of Washington, for her column forcing data,
- and many others who contributed to previous versions of CICE.

1.1.4 Copyright

OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.2 Science Guide

1.2.1 Coupling with host models

The column physics is called from a host (driver) model on a gridpoint by gridpoint basis. Each gridpoint is independent and the host model stores and passes the model state and forcing to the column physics.

The column physics code interface

Subroutine calls and other linkages into Icepack from the host model should only need to access the **icepack_intfc*.F90** interface modules within the `columnphysics/` directory. The Icepack driver in the `configuration/driver/` directory is based on the CICE model and provides an example of the sea ice host model capabilities needed for inclusion of Icepack. In particular, host models will need to include code equivalent to that in the modules **icedrv_*_column.F90**. Calls into the Icepack interface routines are primarily from **icedrv_step_mod.F90** but there are others (search the driver code for `intfc`).

Guiding principles for the creation of Icepack include the following: CHECK THAT THESE ARE TRUE

- The column physics modules shall be independent of all sea ice model infrastructural elements that may vary from model to model. Examples include input/output, timers, references to CPUs or computational tasks, initialization other than that necessary for strictly physical reasons, and anything related to a horizontal grid.
- The column physics modules shall not call or reference any routines or code that reside outside of the **columnphysics/** directory.
- Any capabilities required by a host sea ice model (e.g. calendar variables, tracer flags, diagnostics) shall be implemented in the driver and passed into or out of the column physics modules via array arguments.

For more information, see the *Icepack Column Physics* section.

Atmosphere and ocean boundary forcing

Table 1: *External forcing data that are relevant to Icepack*

Table 1.1: Table 1

Variable	Description	External Interactions
z_o	Atmosphere level height	From *atmosphere model* **to** *sea ice model*
$\vec{U}_a$	Wind velocity	From *atmosphere model* **to** *sea ice model*
Q_a	Specific humidity	From *atmosphere model* **to** *sea ice model*
ρ_a	Air density	From *atmosphere model* **to** *sea ice model*
Θ_a	Air potential temperature	From *atmosphere model* **to** *sea ice model*
T_a	Air temperature	From *atmosphere model* **to** *sea ice model*
$F_{sw\downarrow}$	Incoming shortwave radiation (4 bands)	From *atmosphere model* **to** *sea ice model*
$F_{L\downarrow}$	Incoming longwave radiation	From *atmosphere model* **to** *sea ice model*
F_{rain}	Rainfall rate	From *atmosphere model* **to** *sea ice model*
F_{snow}	Snowfall rate	From *atmosphere model* **to** *sea ice model*

Continued on next page

Table 1.1 – continued from previous page

Variable	Description	External Interactions
F_{frzmlt}	Freezing/melting potential	From *ocean model* **to** *sea ice model*
T_w	Sea surface temperature	From *ocean model* **to** *sea ice model*
S	Sea surface salinity	From *ocean model* **to** *sea ice model*
$\vec{U}_w$	Surface ocean currents	From *ocean model* **to** *sea ice model* (available in Icepack driver, not used directly in column physics)
$\vec{\tau}_a$	Wind stress	From *sea ice model* **to** *atmosphere model*
F_s	Sensible heat flux	From *sea ice model* **to** *atmosphere model*
F_l	Latent heat flux	From *sea ice model* **to** *atmosphere model*
$F_{L\uparrow}$	Outgoing longwave radiation	From *sea ice model* **to** *atmosphere model*
F_{evap}	Evaporated water	From *sea ice model* **to** *atmosphere model*
α	Surface albedo (4 bands)	From *sea ice model* **to** *atmosphere model*
T_{sfc}	Surface temperature	From *sea ice model* **to** *atmosphere model*
$F_{sw\Downarrow}$	Penetrating shortwave radiation	From *sea ice model* **to** *ocean model*
F_{water}	Fresh water flux	From *sea ice model* **to** *ocean model*
F_{hocn}	Net heat flux to ocean	From *sea ice model* **to** *ocean model*
F_{salt}	Salt flux	From *sea ice model* **to** *ocean model*
$\vec{\tau}_w$	Ice-ocean stress	From *sea ice model* **to** *ocean model*
F_{bio}	Biogeochemical fluxes	From *sea ice model* **to** *ocean model*
a_i	Ice fraction	From *sea ice model* **to** both *ocean and atmosphere models*
T_a^{ref}	2m reference temperature (diagnostic)	From *sea ice model* **to** both *ocean and atmosphere models*
Q_a^{ref}	2m reference humidity (diagnostic)	From *sea ice model* **to** both *ocean and atmosphere models*
F_{swabs}	Absorbed shortwave (diagnostic)	From *sea ice model* **to** both *ocean and atmosphere models*

The ice fraction a_i (aice) is the total fractional ice coverage of a grid cell. That is, in each cell,

$$\begin{array}{rl} a_i = 0 & \text{if there is no ice} \\ a_i = 1 & \text{if there is no open water} \\ 0 < a_i < 1 & \text{if there is both ice and open water,} \end{array}$$

where a_i is the sum of fractional ice areas for each category of ice. The ice fraction is used by the flux coupler to merge fluxes from the ice model with fluxes from the other components. For example, the penetrating shortwave radiation flux, weighted by a_i, is combined with the net shortwave radiation flux through ice-free leads, weighted by $(1 - a_i)$, to obtain the net shortwave flux into the ocean over the entire grid cell. The CESM flux coupler requires the fluxes to be divided by the total ice area so that the ice and land models are treated identically (land also may occupy less than 100% of an atmospheric grid cell). These fluxes are "per unit ice area" rather than "per unit grid cell area."

In some coupled climate models (for example, recent versions of the U.K. Hadley Centre model) the surface air temperature and fluxes are computed within the atmosphere model and are passed to CICE for use in the column physics. In this case the logical parameter `calc_Tsfc` in *ice_therm_vertical* is set to false. The fields `fsurfn` (the net surface heat flux from the atmosphere), `flatn` (the surface latent heat flux), and `fcondtopn` (the conductive flux at the top surface) for each ice thickness category are copied or derived from the input coupler fluxes and are passed to the thermodynamic driver subroutine, *thermo_vertical*. At the end of the time step, the surface temperature and effective conductivity (i.e., thermal conductivity divided by thickness) of the top ice/snow layer in each category are returned to the atmosphere model via the coupler. Since the ice surface temperature is treated explicitly, the effective conductivity may need to be limited to ensure stability. As a result, accuracy may be significantly reduced, especially for thin ice or snow layers. A more stable and accurate procedure would be to compute the temperature profiles for both the atmosphere and ice, together with the surface fluxes, in a single implicit calculation. This was judged impractical, however, given that the atmosphere and sea ice models generally exist on different grids and/or processor sets.

Atmosphere

The wind velocity, specific humidity, air density and potential temperature at the given level height $z_\circ$ are used to compute transfer coefficients used in formulas for the surface wind stress and turbulent heat fluxes $\vec\tau_a$, F_s, and F_l, as described below. The sensible and latent heat fluxes, F_s and F_l, along with shortwave and longwave radiation, $F_{sw\downarrow}$, $F_{L\downarrow}$ and $F_{L\uparrow}$, are included in the flux balance that determines the ice or snow surface temperature when calc_Tsfc = true. As described in the *Thermodynamics* section, these fluxes depend nonlinearly on the ice surface temperature T_{sfc}. The balance equation is iterated until convergence, and the resulting fluxes and T_{sfc} are then passed to the flux coupler.

The snowfall precipitation rate (provided as liquid water equivalent and converted by the ice model to snow depth) also contributes to the heat and water mass budgets of the ice layer. Melt ponds generally form on the ice surface in the Arctic and refreeze later in the fall, reducing the total amount of fresh water that reaches the ocean and altering the heat budget of the ice; this version includes two new melt pond parameterizations. Rain and all melted snow end up in the ocean.

Wind stress and transfer coefficients for the turbulent heat fluxes are computed in subroutine *atmo_boundary_layer* following *[25]*. For clarity, the equations are reproduced here in the present notation.

The wind stress and turbulent heat flux calculation accounts for both stable and unstable atmosphere–ice boundary layers. Define the "stability"

$$\Upsilon = \frac{\kappa g z_\circ}{u^{*2}} \left(\frac{\Theta^*}{\Theta_a\left(1+0.606Q_a\right)} + \frac{Q^*}{1/0.606 + Q_a} \right), \tag{1.1}$$

where κ is the von Karman constant, g is gravitational acceleration, and u^*, Θ^* and Q^* are turbulent scales for velocity, temperature, and humidity, respectively:

$$\begin{aligned} u^* &= c_u \left|\vec{U}_a\right|, \\ \Theta^* &= c_\theta\left(\Theta_a - T_{sfc}\right), \\ Q^* &= c_q\left(Q_a - Q_{sfc}\right). \end{aligned} \tag{1.2}$$

The wind speed has a minimum value of 1 m/s. We have ignored ice motion in u^*, and T_{sfc} and Q_{sfc} are the surface temperature and specific humidity, respectively. The latter is calculated by assuming a saturated surface, as described in the *Thermodynamic surface forcing balance* section.

Neglecting form drag,the exchange coefficients c_u, c_θ and c_q are initialized as

$$\frac{\kappa}{\ln(z_{ref}/z_{ice})} \tag{1.3}$$

and updated during a short iteration, as they depend upon the turbulent scales. The number of iterations is set by the namelist variable `natmiter`. (For the case with form drag, see the *Variable exchange coefficients* section.) Here, z_{ref} is a reference height of 10m and z_{ice} is the roughness length scale for the given sea ice category. Υ is constrained to have magnitude less than 10. Further, defining $\chi = \left(1-16\Upsilon\right)^{0.25}$ and $\chi \geq 1$, the "integrated flux profiles" for momentum and stability in the unstable ($\Upsilon < 0$) case are given by

$$\begin{aligned} \psi_m &= 2\ln\left[0.5(1+\chi)\right] + \ln\left[0.5(1+\chi^2)\right] - 2\tan^{-1}\chi + \frac{\pi}{2}, \\ \psi_s &= 2\ln\left[0.5(1+\chi^2)\right]. \end{aligned} \tag{1.4}$$

In a departure from the parameterization used in [25], we use profiles for the stable case following [24],

$$\psi_m = \psi_s = -\left[0.7\Upsilon + 0.75\left(\Upsilon - 14.3\right)\exp\left(-0.35\Upsilon\right) + 10.7\right]. \tag{1.5}$$

The coefficients are then updated as

$$\begin{aligned} c_u' &= \frac{c_u}{1 + c_u\left(\lambda - \psi_m\right)/\kappa} \\ c_\theta' &= \frac{c_\theta}{1 + c_\theta\left(\lambda - \psi_s\right)/\kappa} \\ c_q' &= c_\theta' \end{aligned} \tag{1.6}$$

where $\lambda = \ln\left(z_\circ/z_{ref}\right)$. The first iteration ends with new turbulent scales from equations (1.2). After five iterations the latent and sensible heat flux coefficients are computed, along with the wind stress:

$$\begin{aligned} C_l &= \rho_a\left(L_{vap} + L_{ice}\right)u^* c_q \\ C_s &= \rho_a c_p u^* c_\theta^* + 1, \\ \vec{\tau}_a &= \frac{\rho_a u^{*2}\vec{U}_a}{|\vec{U}_a|}, \end{aligned} \tag{1.7}$$

where L_{vap} and L_{ice} are latent heats of vaporization and fusion, ρ_a is the density of air and c_p is its specific heat. Again following [24], we have added a constant to the sensible heat flux coefficient in order to allow some heat to pass between the atmosphere and the ice surface in stable, calm conditions.

The atmospheric reference temperature T_a^{ref} is computed from T_a and T_{sfc} using the coefficients c_u, c_θ and c_q. Although the sea ice model does not use this quantity, it is convenient for the ice model to perform this calculation. The atmospheric reference temperature is returned to the flux coupler as a climate diagnostic. The same is true for the reference humidity, Q_a^{ref}.

Additional details about the latent and sensible heat fluxes and other quantities referred to here can be found in the *Thermodynamic surface forcing balance* section.

Ocean

New sea ice forms when the ocean temperature drops below its freezing temperature. In the Bitz and Lipscomb thermodynamics, [6] $T_f = -\mu S$, where S is the seawater salinity and $\mu = 0.054$ °/ppt is the ratio of the freezing temperature of brine to its salinity (linear liquidus approximation). For the mushy thermodynamics, T_f is given by a piecewise linear liquidus relation. The ocean model calculates the new ice formation; if the freezing/melting potential F_{frzmlt} is positive, its value represents a certain amount of frazil ice that has formed in one or more layers of the ocean and floated to the surface. (The ocean model assumes that the amount of new ice implied by the freezing potential actually forms.)

If F_{frzmlt} is negative, it is used to heat already existing ice from below. In particular, the sea surface temperature and salinity are used to compute an oceanic heat flux F_w ($|F_w| \leq |F_{frzmlt}|$) which is applied at the bottom of the ice. The portion of the melting potential actually used to melt ice is returned to the coupler in F_{hocn}. The ocean model adjusts its own heat budget with this quantity, assuming that the rest of the flux remained in the ocean.

In addition to runoff from rain and melted snow, the fresh water flux F_{water} includes ice melt water from the top surface and water frozen (a negative flux) or melted at the bottom surface of the ice. This flux is computed as the net change of fresh water in the ice and snow volume over the coupling time step, excluding frazil ice formation and newly accumulated snow. Setting the namelist option update_ocn_f to true causes frazil ice to be included in the fresh water and salt fluxes.

There is a flux of salt into the ocean under melting conditions, and a (negative) flux when sea water is freezing. However, melting sea ice ultimately freshens the top ocean layer, since the ocean is much more saline than the ice. The ice model passes the net flux of salt F_{salt} to the flux coupler, based on the net change in salt for ice in all categories.

In the present configuration, ice_ref_salinity is used for computing the salt flux, although the ice salinity used in the thermodynamic calculation has differing values in the ice layers.

A fraction of the incoming shortwave $F_{sw\Downarrow}$ penetrates the snow and ice layers and passes into the ocean, as described in the *Thermodynamic surface forcing balance* section.

CHECK icepack_ocean.F90?

A thermodynamic slab ocean mixed-layer parameterization is available in **icepack_ocean.F90** and can be run in the full CICE configuration. The turbulent fluxes are computed above the water surface using the same parameterizations as for sea ice, but with parameters appropriate for the ocean. The surface flux balance takes into account the turbulent fluxes, oceanic heat fluxes from below the mixed layer, and shortwave and longwave radiation, including that passing through the sea ice into the ocean. If the resulting sea surface temperature falls below the salinity-dependent freezing point, then new ice (frazil) forms. Otherwise, heat is made available for melting the ice.

Variable exchange coefficients

In the default configuration, atmospheric and oceanic neutral drag coefficients (c_u and c_w) are assumed constant in time and space. These constants are chosen to reflect friction associated with an effective sea ice surface roughness at the ice–atmosphere and ice–ocean interfaces. Sea ice (in both Arctic and Antarctic) contains pressure ridges as well as floe and melt pond edges that act as discrete obstructions to the flow of air or water past the ice, and are a source of form drag. Following *[46]* and based on recent theoretical developments *[30][29]*, the neutral drag coefficients can now be estimated from properties of the ice cover such as ice concentration, vertical extent and area of the ridges, freeboard and floe draft, and size of floes and melt ponds. The new parameterization allows the drag coefficients to be coupled to the sea ice state and therefore to evolve spatially and temporally. This parameterization is contained in the subroutine *neutral_drag_coeffs* and is accessed by setting *formdrag* = true in the namelist.

Following *[46]*, consider the general case of fluid flow obstructed by N randomly oriented obstacles of height H and transverse length L_y, distributed on a domain surface area S_T. Under the assumption of a logarithmic fluid velocity profile, the general formulation of the form drag coefficient can be expressed as

$$C_d = \frac{N c S_c^2 \gamma L_y H}{2 S_T} \left[\frac{\ln(H/z_0)}{\ln(z_{ref}/z_0)} \right]^2, \tag{1.8}$$

where z_0 is a roughness length parameter at the top or bottom surface of the ice, γ is a geometric factor, c is the resistance coefficient of a single obstacle, and S_c is a sheltering function that takes into account the shielding effect of the obstacle,

$$S_c = \left(1 - \exp(-s_l D/H)\right)^{1/2}, \tag{1.9}$$

with D the distance between two obstacles and s_l an attenuation parameter.

As in the original drag formulation in CICE (*Atmosphere* and *Ocean* sections), c_u and c_w along with the transfer coefficients for sensible heat, c_θ, and latent heat, c_q, are initialized to a situation corresponding to neutral atmosphere–ice and ocean–ice boundary layers. The corresponding neutral exchange coefficients are then replaced by coefficients that explicitly account for form drag, expressed in terms of various contributions as

$$\texttt{Cdn\_atm} = \texttt{Cdn\_atm\_rdg} + \texttt{Cdn\_atm\_floe} + \texttt{Cdn\_atm\_skin} + \texttt{Cdn\_atm\_pond}, \tag{1.10}$$

$$\texttt{Cdn\_ocn} = \texttt{Cdn\_ocn\_rdg} + \texttt{Cdn\_ocn\_floe} + \texttt{Cdn\_ocn\_skin}. \tag{1.11}$$

The contributions to form drag from ridges (and keels underneath the ice), floe edges and melt pond edges can be expressed using the general formulation of equation (1.8) (see *[46]* for details). Individual terms in equation (1.11) are fully described in *[46]*. Following *[3]* the skin drag coefficient is parametrized as

$$\texttt{Cdn\_(atm/ocn)\_skin} = a_i \left(1 - m_{(s/k)} \frac{H_{(s/k)}}{D_{(s/k)}}\right) c_{s(s/k)}, \text{ if } \frac{H_{(s/k)}}{D_{(s/k)}} \ge \frac{1}{m_{(s/k)}}, \tag{1.12}$$

where m_s (m_k) is a sheltering parameter that depends on the average sail (keel) height, H_s (H_k), but is often assumed constant, D_s (D_k) is the average distance between sails (keels), and c_{ss} (c_{sk}) is the unobstructed atmospheric (oceanic) skin drag that would be attained in the absence of sails (keels) and with complete ice coverage, $a_{ice} = 1$.

Calculation of equations (1.8) – (1.12) requires that small-scale geometrical properties of the ice cover be related to average grid cell quantities already computed in the sea ice model. These intermediate quantities are briefly presented here and described in more detail in *[46]*. The sail height is given by

$$H_s = 2\frac{v_{rdg}}{a_{rdg}}\left(\frac{\alpha\tan\alpha_k R_d + \beta\tan\alpha_s R_h}{\phi_r\tan\alpha_k R_d + \phi_k\tan\alpha_s R_h^2}\right), \tag{1.13}$$

and the distance between sails

$$D_s = 2H_s\frac{a_i}{a_{rdg}}\left(\frac{\alpha}{\tan\alpha_s} + \frac{\beta}{\tan\alpha_k}\frac{R_h}{R_d}\right), \tag{1.14}$$

where $0 < \alpha < 1$ and $0 < \beta < 1$ are weight functions, α_s and α_k are the sail and keel slope, ϕ_s and ϕ_k are constant porosities for the sails and keels, and we assume constant ratios for the average keel depth and sail height ($H_k/H_s = R_h$) and for the average distances between keels and between sails ($D_k/D_s = R_d$). With the assumption of hydrostatic equilibrium, the effective ice plus snow freeboard is $H_f = \bar{h}_i(1 - \rho_i/\rho_w) + \bar{h}_s(1 - \rho_s/\rho_w)$, where ρ_i, ρ_w and ρ_s are respectively the densities of sea ice, water and snow, $\bar{h}_i$ is the mean ice thickness and $\bar{h}_s$ is the mean snow thickness (means taken over the ice covered regions). For the melt pond edge elevation we assume that the melt pond surface is at the same level as the ocean surface surrounding the floes *[12][13][14]* and use the simplification $H_p = H_f$. Finally to estimate the typical floe size L_A, distance between floes, D_F, and melt pond size, L_P we use the parameterizations of *[30]* to relate these quantities to the ice and pond concentrations. All of these intermediate quantities are available for output, along with `Cdn_atm`, `Cdn_ocn` and the ratio `Cdn_atm_ratio_n` between the total atmospheric drag and the atmospheric neutral drag coefficient.

We assume that the total neutral drag coefficients are thickness category independent, but through their dependance on the diagnostic variables described above, they vary both spatially and temporally. The total drag coefficients and heat transfer coefficients will also depend on the type of stratification of the atmosphere and the ocean, and we use the parameterization described in the *Atmosphere* section that accounts for both stable and unstable atmosphere–ice boundary layers. In contrast to the neutral drag coefficients the stability effect of the atmospheric boundary layer is calculated separately for each ice thickness category.

The transfer coefficient for oceanic heat flux to the bottom of the ice may be varied based on form drag considerations by setting the namelist variable `fbot_xfer_type` to `Cdn_ocn`; this is recommended when using the form drag parameterization. Its default value of the transfer coefficient is 0.006 (`fbot_xfer_type = 'constant'`).

1.2.2 Model components

The Arctic and Antarctic sea ice packs are mixtures of open water, thin first-year ice, thicker multiyear ice, and thick pressure ridges. The thermodynamic and dynamic properties of the ice pack depend on how much ice lies in each thickness range. Thus the basic problem in sea ice modeling is to describe the evolution of the ice thickness distribution (ITD) in time and space.

The fundamental equation solved by CICE is *[44]*:

$$\frac{\partial g}{\partial t} = -\nabla\cdot(g\mathbf{u}) - \frac{\partial}{\partial h}(fg) + \psi - L, \tag{1.15}$$

where $\mathbf{u}$ is the horizontal ice velocity, $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$, f is the rate of thermodynamic ice growth, ψ is a ridging redistribution function, L is the lateral melt rate and g is the ice thickness distribution function. We define $g(\mathbf{x}, h, t)\,dh$ as the fractional area covered by ice in the thickness range $(h, h+dh)$ at a given time and location. Icepack represents all of the terms in this equation except for the divergence (the first term on the right).

Equation (1.15) is solved by partitioning the ice pack in each grid cell into discrete thickness categories. The number of categories can be set by the user, with a default value $N_C = 5$. (Five categories, plus open water, are generally sufficient to simulate the annual cycles of ice thickness, ice strength, and surface fluxes *[5][27]*.) Each category n has lower thickness bound H_{n-1} and upper bound H_n. The lower bound of the thinnest ice category, H_0, is set to zero. The other boundaries are chosen with greater resolution for small h, since the properties of the ice pack are especially sensitive to the amount of thin ice *[31]*. The continuous function $g(h)$ is replaced by the discrete variable a_{in}, defined as the fractional area covered by ice in the open water by a_{i0}, giving $\sum_{n=0}^{N_C} a_{in} = 1$ by definition.

Category boundaries are computed in *init_itd* using one of several formulas, summarized in *Table 2*. Setting the namelist variable `kcatbound` equal to 0 or 1 gives lower thickness boundaries for any number of thickness categories N_C. *Table 2* shows the boundary values for N_C = 5 and linear remapping of the ice thickness distribution. A third option specifies the boundaries based on the World Meteorological Organization classification; the full WMO thickness distribution is used if N_C = 7; if N_C = 5 or 6, some of the thinner categories are combined. The original formula (`kcatbound` = 0) is the default. Category boundaries differ from those shown in *Table 2* for the delta-function ITD. Users may substitute their own preferred boundaries in *init_itd*.

Table 2 : *Lower boundary values for thickness categories, in meters, for the three distribution options (* `kcatbound` *) and linear remapping (kitd* = 1 *). In the WMO case, the distribution used depends on the number of categories used.*

Table 1.2: Table 2

distribution	original	round	WMO		
kcatbound	0	1	2		
N_C	5	5	5	6	7
categories	lower bound (m)				
1	0.00	0.00	0.00	0.00	0.00
2	0.64	0.60	0.30	0.15	0.10
3	1.39	1.40	0.70	0.30	0.15
4	2.47	2.40	1.20	0.70	0.30
5	4.57	3.60	2.00	1.20	0.70
6				2.00	1.20
7					2.00

Tracers

Numerous tracers are available with the column physics. Several of these are required (surface temperature and thickness, salinity and enthalpy of ice and snow layers), and many others are options. For instance, there are tracers to track the age of the ice; the area of first-year ice, fractions of ice area and volume that are level, from which the amount of deformed ice can be calculated; pond area, volume and ice-covered volume; aerosols and numerous other biogeochemical tracers.

Tracers that depend on other tracers

Tracers may be defined that depend on other tracers. Melt pond tracers provide an example (these equations pertain to cesm and topo tracers; level-ice tracers are similar with an extra factor of a_{lvl}, see Equations (1.67)–(1.70). Conservation equations for pond area fraction $a_{pnd}a_i$ and pond volume $h_{pnd}a_{pnd}a_i$, given the ice velocity $\mathbf{u}$, are

$$\frac{\partial}{\partial t}(a_{pnd}a_i) + \nabla\cdot(a_{pnd}a_i\mathbf{u}) = 0, \tag{1.16}$$

$$\frac{\partial}{\partial t}(h_{pnd}a_{pnd}a_i) + \nabla\cdot(h_{pnd}a_{pnd}a_i\mathbf{u}) = 0. \tag{1.17}$$

(These equations represent quantities within one thickness category; all melt pond calculations are performed for each category, separately.) Equation (1.17) expresses conservation of melt pond volume, but in this form highlights that the quantity tracked in the code is the pond depth tracer h_{pnd}, which depends on the pond area tracer a_{pnd}. Likewise, a_{pnd} is a tracer on ice area (Equation (1.16)), which is a state variable, not a tracer.

For a generic quantity q that represents a mean value over the ice fraction, qa_i is the average value over the grid cell. Thus for cesm or topo melt ponds, h_{pnd} can be considered the actual pond depth, $h_{pnd}a_{pnd}$ is the mean pond depth over the sea ice, and $h_{pnd}a_{pnd}a_i$ is the mean pond depth over the grid cell. These quantities are illustrated in *Figure 1*.

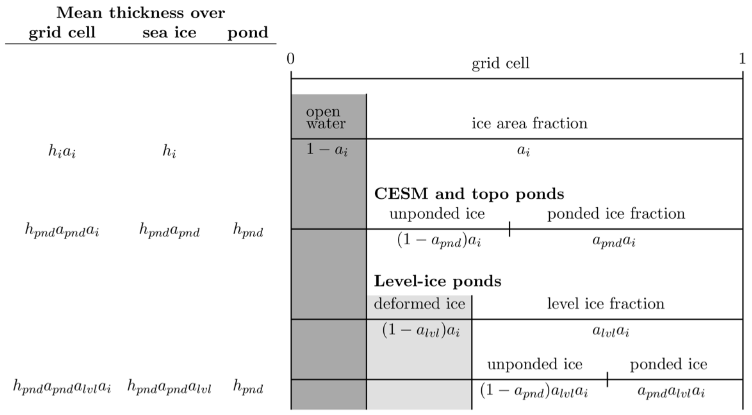

Fig. 1.1: Figure 1

Figure 1 : Melt pond tracer definitions. The graphic on the right illustrates the *grid cell* fraction of ponds or level ice as defined by the tracers. The chart on the left provides corresponding ice thickness and pond depth averages over the grid cell, sea ice and pond area fractions.

Tracers may need to be modified for physical reasons outside of the "core" module or subroutine describing their evolution. For example, when new ice forms in open water, the new ice does not yet have ponds on it. Likewise when sea ice deforms, we assume that pond water (and ice) on the portion of ice that ridges is lost to the ocean.

When new ice is added to a grid cell, the *grid cell* total area of melt ponds is preserved within each category gaining ice, $a_{pnd}^{t+\Delta t}a_i^{t+\Delta t} = a_{pnd}^t a_i^t$, or

$$a_{pnd}^{t+\Delta t} = \frac{a_{pnd}^t a_i^t}{a_i^{t+\Delta t}}. \tag{1.18}$$

Similar calculations are performed for all tracer types, for example tracer-on-tracer dependencies such as h_{pnd}, when needed:

$$h_{pnd}^{t+\Delta t} = \frac{h_{pnd}^t a_{pnd}^t a_i^t}{a_{pnd}^{t+\Delta t} a_i^{t+\Delta t}}. \tag{1.19}$$

In this case (adding new ice), h_{pnd} does not change because $a_{pnd}^{t+\Delta t}a_i^{t+\Delta t} = a_{pnd}^t a_i^t$.

When ice is transferred between two thickness categories, we conserve the total pond area summed over categories n,

$$\sum_n a_{pnd}^{t+\Delta t}(n)a_i^{t+\Delta t}(n) = \sum_n a_{pnd}^t(n)a_i^t(n). \tag{1.20}$$

Thus,

$$\begin{aligned} a_{pnd}^{t+\Delta t}(m) &= \frac{\sum_n a_{pnd}^t(n)a_i^t(n) - \sum_{n\neq m} a_{pnd}^{t+\Delta t}(n)a_i^{t+\Delta t}(n)}{a_i^{t+\Delta t}(m)} \\ &= \frac{a_{pnd}^t(m)a_i^t(m) + \sum_{n\neq m} \Delta\left(a_{pnd}a_i\right)^{t+\Delta t}}{a_i^{t+\Delta t}(m)} \end{aligned} \tag{1.21}$$

This is more complicated because of the Δ term on the right-hand side, which is handled in subroutine *icepack_compute_tracers*. Such tracer calculations are scattered throughout the code, wherever there are changes to the ice thickness distribution.

Note that if a quantity such as a_{pnd} becomes zero in a grid cell's thickness category, then all tracers that depend on it also become zero. If a tracer should be conserved (e.g., aerosols and the liquid water in topo ponds), additional code must be added to track changes in the conserved quantity.

More information about the melt pond schemes is in the *Melt ponds* section.

Ice age

The age of the ice, τ_{age}, is treated as an ice-volume tracer (*trcr_depend* = 1). It is initialized at 0 when ice forms as frazil, and the ice ages the length of the timestep during each timestep. Freezing directly onto the bottom of the ice does not affect the age, nor does melting. Mechanical redistribution processes and advection alter the age of ice in any given grid cell in a conservative manner following changes in ice area. The sea ice age tracer is validated in *[20]*.

Another age-related tracer, the area covered by first-year ice a_{FY}, is an area tracer (*trcr_depend* = 0) that corresponds more closely to satellite-derived ice age data for first-year ice than does τ_{age}. It is re-initialized each year on 15 September (`yday` = 259) in the northern hemisphere and 15 March (`yday` = 75) in the southern hemisphere, in non-leap years. This tracer is increased when new ice forms in open water, in subroutine *add_new_ice* in **icepack_therm_itd.F90**. The first-year area tracer is discussed in *[2]*.

Transport in thickness space

Next we solve the equation for ice transport in thickness space due to thermodynamic growth and melt,

$$\frac{\partial g}{\partial t} + \frac{\partial}{\partial h}(fg) = 0, \tag{1.22}$$

which is obtained from Equation (1.15) by neglecting the first and third terms on the right-hand side. We use the remapping method of *[27]*, in which thickness categories are represented as Lagrangian grid cells whose boundaries are projected forward in time. The thickness distribution function g is approximated as a linear function of h in each displaced category and is then remapped onto the original thickness categories. This method is numerically smooth and is not too diffusive. It can be viewed as a 1D simplification of the 2D incremental remapping scheme described above.

We first compute the displacement of category boundaries in thickness space. Assume that at time m the ice areas a_n^m and mean ice thicknesses h_n^m are known for each thickness category. (For now we omit the subscript i that distinguishes ice from snow.) We use a thermodynamic model (*Thermodynamics*) to compute the new mean thicknesses h_n^{m+1} at time $m+1$. The time step must be small enough that trajectories do not cross; i.e., $h_n^{m+1} < h_{n+1}^{m+1}$ for each pair of adjacent categories. The growth rate at $h = h_n$ is given by $f_n = (h_n^{m+1} - h_n^m)/\Delta t$. By linear interpolation we estimate the growth rate F_n at the upper category boundary H_n:

$$F_n = f_n + \frac{f_{n+1} - f_n}{h_{n+1} - h_n}(H_n - h_n). \tag{1.23}$$

If a_n or $a_{n+1} = 0$, F_n is set to the growth rate in the nonzero category, and if $a_n = a_{n+1} = 0$, we set $F_n = 0$. The temporary displaced boundaries are given by

$$H_n^* = H_n + F_n \, \Delta t, \; n = 1 \text{ to } N-1 \tag{1.24}$$

The boundaries must not be displaced by more than one category to the left or right; that is, we require $H_{n-1} < H_n^* < H_{n+1}$. Without this requirement we would need to do a general remapping rather than an incremental remapping, at the cost of added complexity.

Next we construct $g(h)$ in the displaced thickness categories. The ice areas in the displaced categories are $a_n^{m+1} = a_n^m$, since area is conserved following the motion in thickness space (i.e., during vertical ice growth or melting). The new ice volumes are $v_n^{m+1} = (a_n h_n)^{m+1} = a_n^m h_n^{m+1}$. For conciseness, define $H_L = H_{n-1}^*$ and $H_R = H_n^*$ and drop the time index $m+1$. We wish to construct a continuous function $g(h)$ within each category such that the total area and volume at time $m+1$ are a_n and v_n, respectively:

$$\int_{H_L}^{H_R} g\, dh = a_n, \tag{1.25}$$

$$\int_{H_L}^{H_R} h\, g\, dh = v_n. \tag{1.26}$$

The simplest polynomial that can satisfy both equations is a line. It is convenient to change coordinates, writing $g(\eta) = g_1\eta + g_0$, where $\eta = h - H_L$ and the coefficients g_0 and g_1 are to be determined. Then Equations (1.25) and (1.26) can be written as

$$g_1 \frac{\eta_R^2}{2} + g_0 \eta_R = a_n, \tag{1.27}$$

$$g_1 \frac{\eta_R^3}{3} + g_0 \frac{\eta_R^2}{2} = a_n \eta_n, \tag{1.28}$$

where $\eta_R = H_R - H_L$ and $\eta_n = h_n - H_L$. These equations have the solution

$$g_0 = \frac{6a_n}{\eta_R^2}\left(\frac{2\eta_R}{3} - \eta_n\right), \tag{1.29}$$

$$g_1 = \frac{12a_n}{\eta_R^3}\left(\eta_n - \frac{\eta_R}{2}\right). \tag{1.30}$$

Since g is linear, its maximum and minimum values lie at the boundaries, $\eta = 0$ and η_R:

$$g(0) = \frac{6a_n}{\eta_R^2}\left(\frac{2\eta_R}{3} - \eta_n\right) = g_0, \tag{1.31}$$

$$g(\eta_R) = \frac{6a_n}{\eta_R^2}\left(\eta_n - \frac{\eta_R}{3}\right). \tag{1.32}$$

Equation (1.31) implies that $g(0) < 0$ when $\eta_n > 2\eta_R/3$, i.e., when h_n lies in the right third of the thickness range (H_L, H_R). Similarly, Equation (1.32) implies that $g(\eta_R) < 0$ when $\eta_n < \eta_R/3$, i.e., when h_n is in the left third of the range. Since negative values of g are unphysical, a different solution is needed when h_n lies outside the central third of the thickness range. If h_n is in the left third of the range, we define a cutoff thickness, $H_C = 3h_n - 2H_L$, and set $g = 0$ between H_C and H_R. Equations (1.29) and (1.27) are then valid with η_R redefined as $H_C - H_L$. And if h_n is in the right third of the range, we define $H_C = 3h_n - 2H_R$ and set $g = 0$ between H_L and H_C. In this case, (1.29) and (1.27) apply with $\eta_R = H_R - H_C$ and $\eta_n = h_n - H_C$.

Figure 4 illustrates the linear reconstruction of g for the simple cases $H_L = 0$, $H_R = 1$, $a_n = 1$, and $h_n = 0.2$, 0.4, 0.6, and 0.8. Note that g slopes downward ($g_1 < 0$) when h_n is less than the midpoint thickness, $(H_L + H_R)/2 = 1/2$, and upward when h_n exceeds the midpoint thickness. For $h_n = 0.2$ and 0.8, $g = 0$ over part of the range.

Figure 4 : Linear approximation of the thickness distribution function $g(h)$ for an ice category with left boundary $H_L = 0$, right boundary $H_R = 1$, fractional area $a_n = 1$, and mean ice thickness $h_n = 0.2, 0.4, 0.6,$ and 0.8.

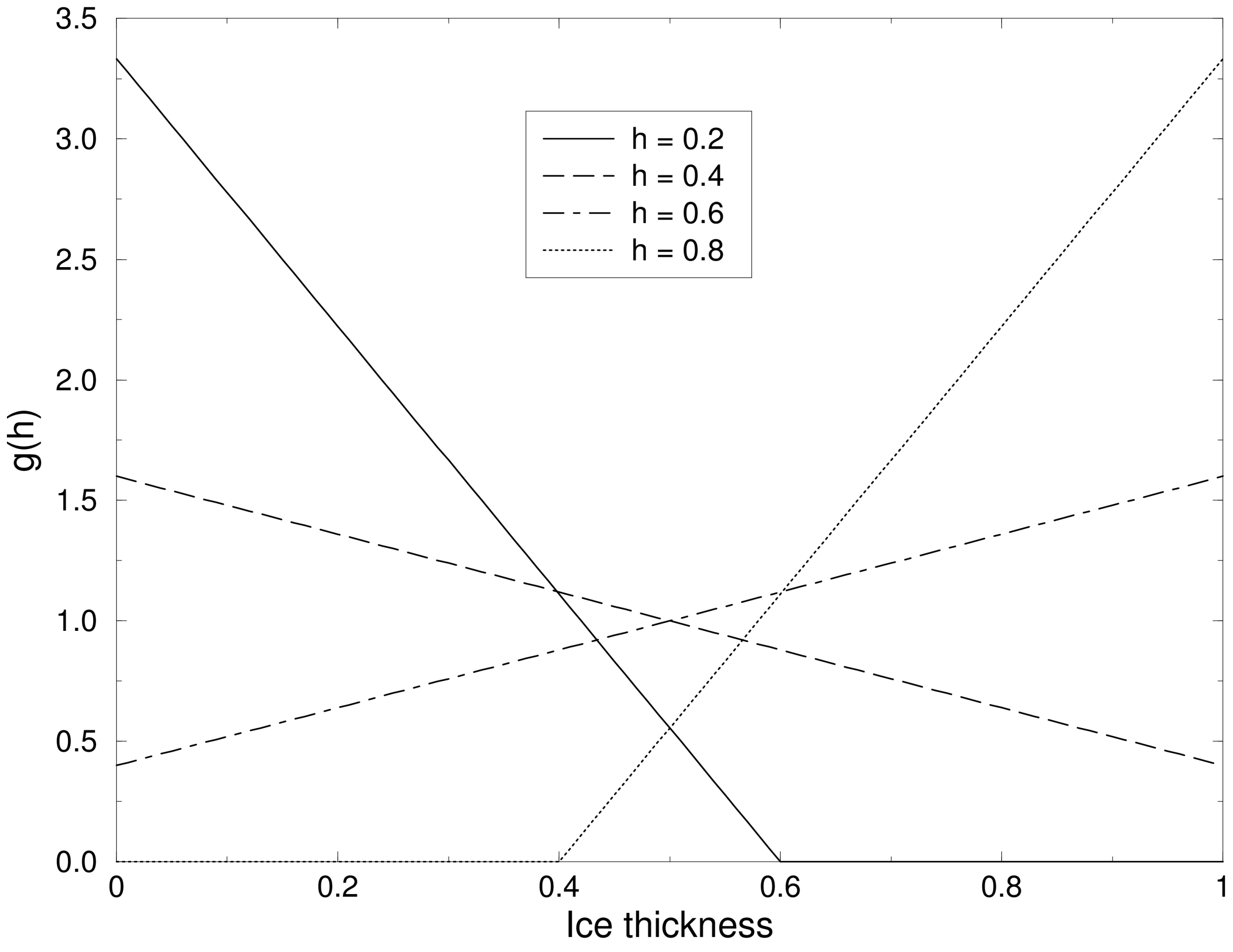

Fig. 1.2: Figure 4

Finally, we remap the thickness distribution to the original boundaries by transferring area and volume between categories. We compute the ice area Δa_n and volume Δv_n between each original boundary H_n and displaced boundary H_n^*. If $H_n^* > H_n$, ice moves from category n to $n+1$. The area and volume transferred are

$$\Delta a_n = \int_{H_n}^{H_n^*} g\, dh, \tag{1.33}$$

$$\Delta v_n = \int_{H_n}^{H_n^*} h\, g\, dh. \tag{1.34}$$

If $H_n^* < H_N$, ice area and volume are transferred from category $n+1$ to n using Equations (1.33) and (1.34) with the limits of integration reversed. To evaluate the integrals we change coordinates from h to $\eta = h - H_L$, where H_L is the left limit of the range over which $g > 0$, and write $g(\eta)$ using Equations (1.29) and (1.27). In this way we obtain the new areas a_n and volumes v_n between the original boundaries H_{n-1} and H_n in each category. The new thicknesses, $h_n = v_n/a_n$, are guaranteed to lie in the range (H_{n-1}, H_n). If $g = 0$ in the part of a category that is remapped to a neighboring category, no ice is transferred.

Other conserved quantities are transferred in proportion to the ice volume Δv_{in}. For example, the transferred ice energy in layer k is $\Delta e_{ink} = e_{ink}(\Delta v_{in}/v_{in})$.

The left and right boundaries of the domain require special treatment. If ice is growing in open water at a rate F_0, the left boundary H_0 is shifted to the right by $F_0 \Delta t$ before g is constructed in category 1, then reset to zero after the remapping is complete. New ice is then added to the grid cell, conserving area, volume, and energy. If ice cannot grow in open water (because the ocean is too warm or the net surface energy flux is downward), H_0 is fixed at zero, and the growth rate at the left boundary is estimated as $F_0 = f_1$. If $F_0 < 0$, all ice thinner than $\Delta h_0 = -F_0 \Delta t$ is assumed to have melted, and the ice area in category 1 is reduced accordingly. The area of new open water is

$$\Delta a_0 = \int_0^{\Delta h_0} g\, dh. \tag{1.35}$$

The right boundary H_N is not fixed but varies with h_N, the mean ice thickness in the thickest category. Given h_N, we set $H_N = 3h_N - 2H_{N-1}$, which ensures that $g(h) > 0$ for $H_{N-1} < h < H_N$ and $g(h) = 0$ for $h \geq H_N$. No ice crosses the right boundary. If the ice growth or melt rates in a given grid cell are too large, the thickness remapping scheme will not work. Instead, the thickness categories in that grid cell are treated as delta functions following *[5]*, and categories outside their prescribed boundaries are merged with neighboring categories as needed. For time steps of less than a day and category thickness ranges of 10 cm or more, this simplification is needed rarely, if ever.

The linear remapping algorithm for thickness is not monotonic for tracers, although significant errors rarely occur. Usually they appear as snow temperatures (enthalpy) outside the physical range of values in very small snow volumes. In this case we transfer the snow and its heat and tracer contents to the ocean.

Mechanical redistribution

The last term on the right-hand side of Equation (1.15) is ψ, which describes the redistribution of ice in thickness space due to ridging and other mechanical processes. The mechanical redistribution scheme in Icepack is based on *[44]*, *[38]*, *[18]*, *[11]*, and *[28]*. This scheme converts thinner ice to thicker ice and is applied after horizontal transport. When the ice is converging, enough ice ridges to ensure that the ice area does not exceed the grid cell area.

First we specify the participation function: the thickness distribution $a_P(h) = b(h)\, g(h)$ of the ice participating in ridging. (We use "ridging" as shorthand for all forms of mechanical redistribution, including rafting.) The weighting function $b(h)$ favors ridging of thin ice and closing of open water in preference to ridging of thicker ice. There are two options for the form of $b(h)$. If `krdg_partic` = 0 in the namelist, we follow *[44]* and set

$$b(h) = \begin{cases} \frac{2}{G^*}\left(1 - \frac{G(h)}{G^*}\right) & \text{if } G(h) < G^* \\ 0 & \text{otherwise} \end{cases} \tag{1.36}$$

where $G(h)$ is the fractional area covered by ice thinner than h, and G^* is an empirical constant. Integrating $a_P(h)$ between category boundaries H_{n-1} and H_n, we obtain the mean value of a_P in category n:

$$a_{Pn} = \frac{2}{G^*}(G_n - G_{n-1})\left(1 - \frac{G_{n-1} + G_n}{2G^*}\right), \tag{1.37}$$

where a_{Pn} is the ratio of the ice area ridging (or open water area closing) in category n to the total area ridging and closing, and G_n is the total fractional ice area in categories 0 to n. Equation (1.37) applies to categories with $G_n < G^*$. If $G_{n-1} < G^* < G_n$, then Equation (1.37) is valid with G^* replacing G_n, and if $G_{n-1} > G^*$, then $a_{Pn} = 0$. If the open water fraction $a_0 > G^*$, no ice can ridge, because "ridging" simply reduces the area of open water. As in *[44]* we set $G^* = 0.15$.

If the spatial resolution is too fine for a given time step Δt, the weighting function Equation (1.36) can promote numerical instability. For $\Delta t = 1$ hour, resolutions finer than $\Delta x \sim 10$ km are typically unstable. The instability results from feedback between the ridging scheme and the dynamics via the ice strength. If the strength changes significantly on time scales less than Δt, the viscous-plastic solution of the momentum equation is inaccurate and sometimes oscillatory. As a result, the fields of ice area, thickness, velocity, strength, divergence, and shear can become noisy and unphysical.

A more stable weighting function was suggested by *[28]*:

$$b(h) = \frac{\exp[-G(h)/a^*]}{a^*[1 - \exp(-1/a^*)]} \tag{1.38}$$

When integrated between category boundaries, Equation (1.38) implies

$$a_{Pn} = \frac{\exp(-G_{n-1}/a^*) - \exp(-G_n/a^*)}{1 - \exp(-1/a^*)} \tag{1.39}$$

This weighting function is used if `krdg_partic` = 1 in the namelist. From Equation (1.38), the mean value of G for ice participating in ridging is a^*, as compared to $G^*/3$ for Equation (1.36). For typical ice thickness distributions, setting $a^* = 0.05$ with `krdg_partic` = 1 gives participation fractions similar to those given by $G^* = 0.15$ with `krdg_partic` = 0. See *[28]* for a detailed comparison of these two participation functions.

Thin ice is converted to thick, ridged ice in a way that reduces the total ice area while conserving ice volume and internal energy. There are two namelist options for redistributing ice among thickness categories. If `krdg_redist` = 0, ridging ice of thickness h_n forms ridges whose area is distributed uniformly between $H_{\min} = 2h_n$ and $H_{\max} = 2\sqrt{H^* h_n}$, as in *[18]*. The default value of H^* is 25 m, as in earlier versions of CICE. Observations suggest that $H^* = 50$ m gives a better fit to first-year ridges *[1]*, although the lower value may be appropriate for multiyear ridges *[11]*. The ratio of the mean ridge thickness to the thickness of ridging ice is $k_n = (H_{\min} + H_{\max})/(2h_n)$. If the area of category n is reduced by ridging at the rate r_n, the area of thicker categories grows simultaneously at the rate r_n/k_n. Thus the *net* rate of area loss due to ridging of ice in category n is $r_n(1 - 1/k_n)$.

The ridged ice area and volume are apportioned among categories in the thickness range $(H_{\min}, H_{\max})$. The fraction of the new ridge area in category m is

$$f_m^{\mathrm{area}} = \frac{H_R - H_L}{H_{\max} - H_{\min}}, \tag{1.40}$$

where $H_L = \max(H_{m-1}, H_{\min})$ and $H_R = \min(H_m, H_{\max})$. The fraction of the ridge volume going to category m is

$$f_m^{\mathrm{vol}} = \frac{(H_R)^2 - (H_L)^2}{(H_{\max})^2 - (H_{\min})^2}. \tag{1.41}$$

This uniform redistribution function tends to produce too little ice in the 3–5 m range and too much ice thicker than 10 m *[1]*. Observations show that the ITD of ridges is better approximated by a negative exponential. Setting `krdg_redist` = 1 gives ridges with an exponential ITD *[28]*:

$$g_R(h) \propto \exp[-(h - H_{\min})/\lambda] \tag{1.42}$$

for $h \geq H_{\min}$, with $g_R(h) = 0$ for $h < H_{\min}$. Here, λ is an empirical e-folding scale and $H_{\min} = 2h_n$ (where h_n is the thickness of ridging ice). We assume that $\lambda = \mu h_n^{1/2}$, where μ (mu_rdg) is a tunable parameter with units . Thus the mean ridge thickness increases in proportion to $h_n^{1/2}$, as in *[18]*. The value $\mu = 4.0$ gives λ in the range 1–4 m for most ridged ice. Ice strengths with $\mu = 4.0$ and `krdg_redist` = 1 are roughly comparable to the strengths with $H^* = 50$ m and `krdg_redist` = 0.

From Equation (1.42) it can be shown that the fractional area going to category m as a result of ridging is

$$f_m^{\rm area} = \exp[-(H_{m-1} - H_{\min})/\lambda] - \exp[-(H_m - H_{\min})/\lambda]. \tag{1.43}$$

The fractional volume going to category m is

$$f_m^{\rm vol} = \frac{(H_{m-1} + \lambda)\exp[-(H_{m-1} - H_{\min})/\lambda] - (H_m + \lambda)\exp[-(H_m - H_{\min})/\lambda]}{H_{min} + \lambda}. \tag{1.44}$$

Equations (1.43) and (1.44) replace Equations (1.40) and (1.41) when `krdg_redist` = 1.

Internal ice energy is transferred between categories in proportion to ice volume. Snow volume and internal energy are transferred in the same way, except that a fraction of the snow may be deposited in the ocean instead of added to the new ridge.

The net area removed by ridging and closing is a function of the strain rates. Let $R_{\rm net}$ be the net rate of area loss for the ice pack (i.e., the rate of open water area closing, plus the net rate of ice area loss due to ridging). Following *[11]*, $R_{\rm net}$ is given by

$$R_{\rm net} = \frac{C_s}{2}(\Delta - |D_D|) - \min(D_D, 0), \tag{1.45}$$

where C_s is the fraction of shear dissipation energy that contributes to ridge-building, D_D is the divergence, and Δ is a function of the divergence and shear. These strain rates are computed by the dynamics scheme. The default value of C_s is 0.25.

Next, define $R_{\rm tot} = \sum_{n=0}^{N} r_n$. This rate is related to $R_{\rm net}$ by

$$R_{\rm net} = \left[a_{P0} + \sum_{n=1}^{N} a_{Pn}\left(1 - \frac{1}{k_n}\right)\right] R_{\rm tot}. \tag{1.46}$$

Given $R_{\rm net}$ from Equation (1.45), we use Equation (1.46) to compute $R_{\rm tot}$. Then the area ridged in category n is given by $a_{rn} = r_n \Delta t$, where $r_n = a_{Pn} R_{\rm tot}$. The area of new ridges is a_{rn}/k_n, and the volume of new ridges is $a_{rn} h_n$ (since volume is conserved during ridging). We remove the ridging ice from category n and use Equations (1.40) and (1.41): (or (1.43) and (1.44)) to redistribute the ice among thicker categories.

Occasionally the ridging rate in thickness category n may be large enough to ridge the entire area a_n during a time interval less than Δt. In this case $R_{\rm tot}$ is reduced to the value that exactly ridges an area a_n during Δt. After each ridging iteration, the total fractional ice area a_i is computed. If $a_i > 1$, the ridging is repeated with a value of $R_{\rm net}$ sufficient to yield $a_i = 1$.

Two tracers for tracking the ridged ice area and volume are available. The actual tracers are for level (undeformed) ice area (*alvl*) and volume (*vlvl*), which are easier to implement for a couple of reasons: (1) ice ridged in a given thickness category is spread out among the rest of the categories, making it more difficult (and expensive) to track than the level ice remaining behind in the original category; (2) previously ridged ice may ridge again, so that simply adding a volume of freshly ridged ice to the volume of previously ridged ice in a grid cell may be inappropriate. Although the code currently only tracks level ice internally, both level ice and ridged ice are available for output. They are simply related:

$$\begin{aligned} a_{lvl} + a_{rdg} &= a_i, \\ v_{lvl} + v_{rdg} &= v_i. \end{aligned} \tag{1.47}$$

Level ice area fraction and volume increase with new ice formation and decrease steadily via ridging processes. Without the formation of new ice, level ice asymptotes to zero because we assume that both level ice and ridged ice ridge, in proportion to their fractional areas in a grid cell (in the spirit of the ridging calculation itself which does not prefer level ice over previously ridged ice).

The ice strength P may be computed in either of two ways. If the namelist parameter kstrength = 0, we use the strength formula from *[17]*:

$$P = P^* h \exp[-C(1 - a_i)], \tag{1.48}$$

where $P^* = 27,500\,\mathrm{N/m}$ and $C = 20$ are empirical constants, and h is the mean ice thickness. Alternatively, setting kstrength = 1 gives an ice strength closely related to the ridging scheme. Following *[38]*, the strength is assumed proportional to the change in ice potential energy ΔE_P per unit area of compressive deformation. Given uniform ridge ITDs (`krdg_redist` = 0), we have

$$P = C_f\, C_p\, \beta \sum_{n=1}^{N_C} \left[-a_{Pn}\, h_n^2 + \frac{a_{Pn}}{k_n} \left(\frac{(H_n^{\max})^3 - (H_n^{\min})^3}{3(H_n^{\max} - H_n^{\min})} \right) \right], \tag{1.49}$$

where $C_P = (g/2)(\rho_i/\rho_w)(\rho_w - \rho_i)$, $\beta = R_{\mathrm{tot}}/R_{\mathrm{net}} > 1$ from Equation (1.46), and C_f is an empirical parameter that accounts for frictional energy dissipation. Following *[11]*, we set $C_f = 17$. The first term in the summation is the potential energy of ridging ice, and the second, larger term is the potential energy of the resulting ridges. The factor of β is included because a_{Pn} is normalized with respect to the total area of ice ridging, not the net area removed. Recall that more than one unit area of ice must be ridged to reduce the net ice area by one unit. For exponential ridge ITDs (`krdg_redist` = 1), the ridge potential energy is modified:

$$P = C_f\, C_p\, \beta \sum_{n=1}^{N_C} \left[-a_{Pn}\, h_n^2 + \frac{a_{Pn}}{k_n} \left(H_{\min}^2 + 2H_{\min}\lambda + 2\lambda^2 \right) \right] \tag{1.50}$$

The energy-based ice strength given by Equations (1.49) or (1.50) is more physically realistic than the strength given by Equation (1.48). However, use of Equation (1.48) is less likely to allow numerical instability at a given resolution and time step. See *[28]* for more details.

Thermodynamics

The current Icepack version includes three thermodynamics options, the "zero-layer" thermodynamics of *[40]* (`ktherm` = 0), the Bitz and Lipscomb model *[6]* (`ktherm` = 1) that assumes a fixed salinity profile, and a new "mushy" formulation (`ktherm` = 2) in which salinity evolves *[47]*. For each thickness category, Icepack computes changes in the ice and snow thickness and vertical temperature profile resulting from radiative, turbulent, and conductive heat fluxes. The ice has a temperature-dependent specific heam to simulate the effect of brine pocket melting and freezing, for `ktherm` = 1 and 2.

Each thickness category n in each grid cell is treated as a horizontally uniform column with ice thickness $h_{in} = v_{in}/a_{in}$ and snow thickness $h_{sn} = v_{sn}/a_{in}$. (Henceforth we omit the category index n.) Each column is divided into N_i ice layers of thickness $\Delta h_i = h_i/N_i$ and N_s snow layers of thickness $\Delta h_s = h_s/N_s$. The surface temperature (i.e., the temperature of ice or snow at the interface with the atmosphere) is T_{sf}, which cannot exceed . The temperature at the midpoint of the snow layer is T_s, and the midpoint ice layer temperatures are T_{ik}, where k ranges from 1 to N_i. The temperature at the bottom of the ice is held at T_f, the freezing temperature of the ocean mixed layer. All temperatures are in degrees Celsius unless stated otherwise.

Each ice layer has an enthalpy q_{ik}, defined as the negative of the energy required to melt a unit volume of ice and raise its temperature to . Because of internal melting and freezing in brine pockets, the ice enthalpy depends on the brine pocket volume and is a function of temperature and salinity. We can also define a snow enthalpy q_s, which depends on temperature alone.

Given surface forcing at the atmosphere–ice and ice–ocean interfaces along with the ice and snow thicknesses and temperatures/enthalpies at time m, the thermodynamic model advances these quantities to time $m + 1$ (`ktherm` = 2 also advances salinity). The calculation proceeds in two steps. First we solve a set of equations for the new temperatures, as discussed in the *New temperatures* section. Then we compute the melting, if any, of ice or snow at the top surface, and the growth or melting of ice at the bottom surface, as described in the *Growth and melting* section. We begin by describing the surface forcing parameterizations, which are closely related to the ice and snow surface temperatures.

Melt ponds

Three explicit melt pond parameterizations are available in Icepack, and all must use the delta-Eddington radiation scheme, described below. The default (ccsm3) shortwave parameterization incorporates melt ponds implicitly by adjusting the albedo based on surface conditions.

For each of the three explicit parameterizations, a volume ΔV_{melt} of melt water produced on a given category may be added to the melt pond liquid volume:

$$\Delta V_{melt} = \frac{r}{\rho_w} \left(\rho_i \Delta h_i + \rho_s \Delta h_s + F_{rain} \Delta t\right) a_i, \tag{1.51}$$

where

$$r = r_{min} + \left(r_{max} - r_{min}\right) a_i \tag{1.52}$$

is the fraction of the total melt water available that is added to the ponds, ρ_i and ρ_s are ice and snow densities, Δh_i and Δh_s are the thicknesses of ice and snow that melted, and F_{rain} is the rainfall rate. Namelist parameters are set for the level-ice (`tr_pond_lvl`) parameterization; in the cesm and topo pond schemes the standard values of r_{max} and r_{min} are 0.7 and 0.15, respectively.

Radiatively, the surface of an ice category is divided into fractions of snow, pond and bare ice. In these melt pond schemes, the actual pond area and depth are maintained throughout the simulation according to the physical processes acting on it. However, snow on the sea ice and pond ice may shield the pond and ice below from solar radiation. These processes do not alter the actual pond volume; instead they are used to define an "effective pond fraction" (and likewise, effective pond depth, snow fraction and snow depth) used only for the shortwave radiation calculation.

In addition to the physical processes discussed below, tracer equations and definitions for melt ponds are also described in the *Tracers* and *Figure 1* sections.

CESM formulation (`tr_pond_cesm` = true)

Melt pond area and thickness tracers are carried on each ice thickness category as in the *Tracers* section. Defined simply as the product of pond area, a_p, and depth, h_p, the melt pond volume, V_p, grows through addition of ice or snow melt water or rain water, and shrinks when the ice surface temperature becomes cold,

$$\begin{aligned} \text{pond growth}: \ V_p' &= V_p(t) + \Delta V_{melt}, \\ \text{pond contraction}: \ V_p(t+\Delta t) &= V_p' \exp\left[r_2\left(\frac{\max\left(T_p - T_{sfc}, 0\right)}{T_p}\right)\right], \end{aligned} \tag{1.53}$$

where dh_i and dh_s represent ice and snow melt at the top surface of each thickness category and $r_2 = 0.01$. Here, T_p is a reference temperature equal to -2 °C. Pond depth is assumed to be a linear function of the pond fraction ($h_p = \delta_p a_p$) and is limited by the category ice thickness ($h_p \le 0.9 h_i$). The pond shape (`pndaspect`) $\delta_p = 0.8$ in the standard CESM pond configuration. The area and thickness are computed according to the assumed pond shape, and the pond area is then reduced in the presence of snow for the radiation calculation. Ponds are allowed only on ice at least 1 cm thick. This formulation differs slightly from that documented in *[19]*.

Topographic formulation (`tr_pond_topo` = true)

The principle concept of this scheme is that melt water runs downhill under the influence of gravity and collects on sea ice with increasing surface height starting at the lowest height *[12][13][14]*. Thus, the topography of the ice cover plays a crucial role in determining the melt pond cover. However, Icepack does not explicitly represent the topography of sea ice. Therefore, we split the existing ice thickness distribution function into a surface height and basal depth distribution assuming that each sea ice thickness category is in hydrostatic equilibrium at the beginning of the melt season. We then calculate the position of sea level assuming that the ice in the whole grid cell is rigid and in hydrostatic equilibrium.

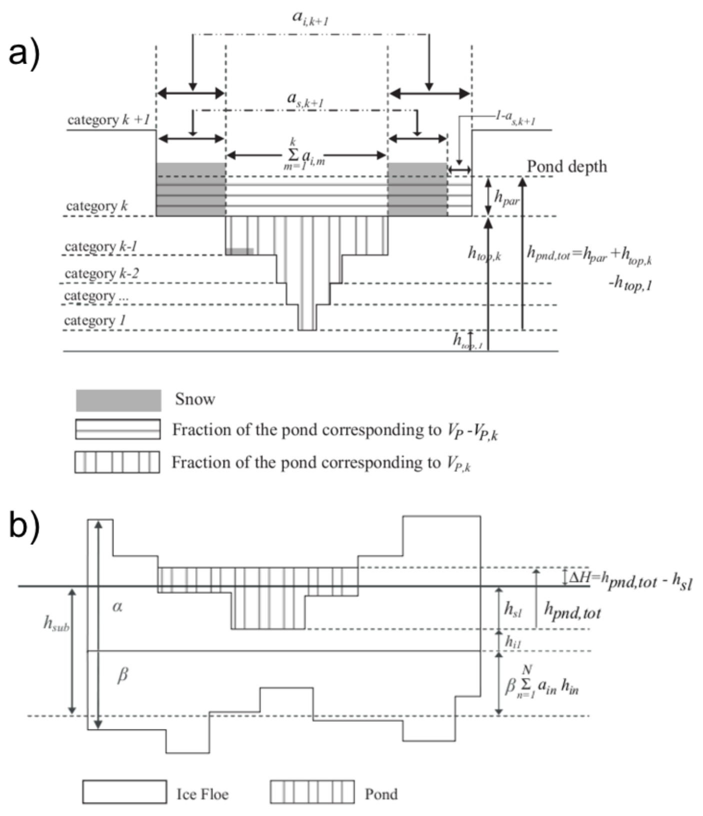

Fig. 1.3: Figure 6

Figure 6 : (a) Schematic illustration of the relationship between the height of the pond surface $h_{pnd,tot}$, the volume of water V_{Pk} required to completely fill up to category k, the volume of water $V_P - V_{Pk}$, and the depth to which this fills up category $k+1$. Ice and snow areas a_i and a_s are also depicted. The volume calculation takes account of the presence of snow, which may be partially or completely saturated. (b) Schematic illustration indicating pond surface height $h_{pnd,tot}$ and sea level h_{sl} measured with respect to the thinnest surface height category h_{i1}, the submerged portion of the floe h_{sub}, and hydraulic head ΔH . A positive hydraulic head (pond surface above sea level) will flush melt water through the sea ice into the ocean; a negative hydraulic head can drive percolation of sea water up onto the ice surface. Here, $\alpha = 0.6$ and $\beta = 0.4$ are the surface height and basal depth distribution fractions. The height of the steps is the height of the ice above the reference level, and the width of the steps is the area of ice of that height. The illustration does not imply a particular assumed topography, rather it is assumed that all thickness categories are present at the sub-grid scale so that water will always flow to the lowest surface height class.

Once a volume of water is produced from ice and snow melting, we calculate the number of ice categories covered by water. At each time step, we construct a list of volumes of water $\{V_{P1}, V_{P2}, ...V_{P,k-1}, V_{Pk}, V_{P,k+1}, ...\}$, where V_{Pk} is the volume of water required to completely cover the ice and snow in the surface height categories from $i = 1$ up to $i = k$. The volume V_{Pk} is defined so that if the volume of water V_P is such that $V_{Pk} < V_P < V_{P,k+1}$ then the snow and ice in categories $n = 1$ up to $n = k+1$ are covered in melt water. *Figure 6* (a) depicts the areas covered in melt water and saturated snow on the surface height (rather than thickness) categories $h_{top,k}$. Note in the code, we assume that $h_{top,n}/h_{in} = 0.6$ (an arbitrary choice). The fractional area of the nth category covered in snow is a_{sn}. The volume V_{P1}, which is the region with vertical hatching, is the volume of water required to completely fill up the first thickness category, so that any extra melt water must occupy the second thickness category, and it is given by the expression

$$V_{P1} = a_{i1}(h_{top,2} - h_{top,1}) - a_{s1}a_{i1}h_{s1}(1 - V_{sw}), \tag{1.54}$$

where V_{sw} is the fraction of the snow volume that can be occupied by water, and h_{s1} is the snow depth on ice height class 1. In a similar way, the volume required to fill up the first and second surface categories, V_{P2}, is given by

$$V_{P2} = a_{i1}(h_{top,3} - h_{top,2}) + a_{i2}(h_{top,3} - h_{top,2}) - a_{s2}a_{i2}h_{s2}(1 - V_{sw}) + V_{P1}. \tag{1.55}$$

The general expression for volume V_{Pk} is given by

$$V_{Pk} = \sum_{m=0}^{k} a_{im}(h_{top,k+1} - h_{top,k}) - a_{sk}a_{ik}h_{sk}(1 - V_{sw}) + \sum_{m=0}^{k-1} V_{Pm}. \tag{1.56}$$

(Note that we have implicitly assumed that $h_{si} < h_{top,k+1} - h_{top,k}$ for all k.) No melt water can be stored on the thickest ice thickness category. If the melt water volume exceeds the volume calculated above, the remaining melt water is released to the ocean.

At each time step, the pond height above the level of the thinnest surface height class, that is, the maximum pond depth, is diagnosed from the list of volumes V_{Pk}. In particular, if the total volume of melt water V_P is such that $V_{Pk} < V_P < V_{P,k+1}$ then the pond height $h_{pnd,tot}$ is

$$h_{pnd,tot} = h_{par} + h_{top,k} - h_{top,1}, \tag{1.57}$$

where h_{par} is the height of the pond above the level of the ice in class k and partially fills the volume between $V_{P,k}$ and $V_{P,k+1}$. From *Figure 6* (a) we see that $h_{top,k} - h_{top,1}$ is the height of the melt water, which has volume V_{Pk}, which completely fills the surface categories up to category k. The remaining volume, $V_P - V_{Pk}$, partially fills category $k + 1$ up to the height h_{par} and there are two cases to consider: either the snow cover on category $k + 1$, with height $h_{s,k+1}$, is completely covered in melt water (i.e., $h_{par} > h_{s,k+1}$), or it is not (i.e., $h_{par} \leq h_{s,k+1}$). From conservation of volume, we see from *Figure 6* (a) that for an incompletely to completely saturated snow cover on surface ice class $k + 1$,

$$V_P - V_{Pk} = \quad h_{par}\left(\sum_{m=1}^{k} a_{ik} + a_{i,k+1}(1 - a_{s,k+1}) + a_{i,k+1}a_{s,k+1}V_{sw}\right) \quad \text{for} \quad h_{par} \leq h_{s,k+1}, \tag{1.58}$$

and for a saturated snow cover with water on top of the snow on surface ice class $k + 1$,

$$\begin{aligned} V_P - V_{Pk} = \quad & h_{par}\left(\sum_{m=1}^{k} a_{ik} + a_{i,k+1}(1 - a_{s,k+1})\right) + a_{i,k+1}a_{s,k+1}V_{sw}h_{s,k+1} \\ + \quad & a_{i,k+1}a_{s,k+1}(h_{par} - h_{s,k+1}) \quad \text{for} \quad h_{par} > h_{s,k+1}. \end{aligned} \tag{1.59}$$

As the melting season progresses, not only does melt water accumulate upon the upper surface of the sea ice, but the sea ice beneath the melt water becomes more porous owing to a reduction in solid fraction [9]. The hydraulic head of melt water on sea ice (i.e., its height above sea level) drives flushing of melt water through the porous sea ice and into the underlying ocean. The mushy thermodynamics scheme (*ktherm* = 2) handles flushing. For *ktherm* $\neq 2$ we model the vertical flushing rate using Darcy's law for flow through a porous medium

$$w = -\frac{\Pi_v}{\mu}\rho_o g \frac{\Delta H}{h_i}, \tag{1.60}$$

where w is the vertical mass flux per unit perpendicular cross-sectional area (i.e., the vertical component of the Darcy velocity), Π_v is the vertical component of the permeability tensor (assumed to be isotropic in the horizontal), μ is the viscosity of water, ρ_o is the ocean density, g is gravitational acceleration, ΔH is the the hydraulic head, and h_i is the thickness of the ice through which the pond flushes. As proposed by [16] the vertical permeability of sea ice can be calculated from the liquid fraction ϕ:

$$\Pi_v = 3 \times 10^{-8}\phi^3 \mathrm{m}^2. \tag{1.61}$$

Since the solid fraction varies throughout the depth of the sea ice, so does the permeability. The rate of vertical drainage is determined by the lowest (least permeable) layer, corresponding to the highest solid fraction. From the equations describing sea ice as a mushy layer *[10]*, the solid fraction is determined by:

$$\phi = \frac{c_i - S}{c_i - S_{br}(T)}, \tag{1.62}$$

where S is the bulk salinity of the ice, $S_{br}(T)$ is the concentration of salt in the brine at temperature T and c_i is the concentration of salt in the ice crystals (set to zero).

The hydraulic head is given by the difference in height between the upper surface of the melt pond $h_{pnd,tot}$ and the sea level h_{sl}. The value of the sea level h_{sl} is calculated from

$$h_{sl} = h_{sub} - 0.4 \sum_{n=1}^{N} a_{in} h_{in} - \beta h_{i1}, \tag{1.63}$$

where $0.4 \sum_{n=1}^{N} a_{in} h_{i,n}$ is the mean thickness of the basal depth classes, and h_{sub} is the depth of the submerged portion of the floe. *Figure 6* (b) depicts the relationship between the hydraulic head and the depths and heights that appear in Equation (1.63). The depth of the submerged portion of the floe is determined from hydrostatic equilibrium to be

$$h_{sub} = \frac{\rho_m}{\rho_w} V_P + \frac{\rho_s}{\rho_w} V_s + \frac{\rho_i}{\rho_w} V_i, \tag{1.64}$$

where ρ_m is the density of melt water, V_P is the total pond volume, V_s is the total snow volume, and V_i is the total ice volume.

When the surface energy balance is negative, a layer of ice is formed at the upper surface of the ponds. The rate of growth of the ice lid is given by the Stefan energy budget at the lid-pond interface

$$\rho_i L_0 \frac{dh_{ipnd}}{dt} = k_i \frac{\partial T_i}{\partial z} - k_p \frac{\partial T_p}{\partial z}, \tag{1.65}$$

where L_0 is the latent heat of fusion of pure ice per unit volume, T_i and T_p are the ice surface and pond temperatures, and k_i and k_p are the thermal conductivity of the ice lid and pond respectively. The second term on the right hand-side is close to zero since the pond is almost uniformly at the freezing temperature *[43]*. Approximating the temperature gradient in the ice lid as linear, the Stefan condition yields the classic Stefan solution for ice lid depth

$$h_{ipnd} = \sqrt{\frac{2k_i}{\rho_s L} \Delta T_i t}, \tag{1.66}$$

where ΔT is the temperature difference between the top and the bottom of the lid. Depending on the surface flux conditions the ice lid can grow, partially melt, or melt completely. Provided that the ice lid is thinner than a critical lid depth (1 cm is suggested) then the pond is regarded as effective, i.e. the pond affects the optical properties of the ice cover. Effective pond area and pond depth for each thickness category are passed to the radiation scheme for calculating albedo. Note that once the ice lid has exceeded the critical thickness, snow may accumulate on the lid causing a substantial increase in albedo. In the current CICE model, melt ponds only affect the thermodynamics of the ice through the albedo. To conserve energy, the ice lid is dismissed once the pond is completely refrozen.

As the sea ice area shrinks due to melting and ridging, the pond volume over the lost area is released to the ocean immediately. In *[13]*, the pond volume was carried as an ice area tracer, but in *[14]* and here, pond area and thickness are carried as separate tracers, as in the *Tracers* section.

Unlike the cesm and level-ice melt pond schemes, the liquid pond water in the topo parameterization is not necessarily virtual; it can be withheld from being passed to the ocean model until the ponds drain by setting the namelist variable `l_mpond_fresh` = .true. The refrozen pond lids are still virtual. Extra code needed to track and enforce conservation of water has been added to **icepack_itd.F90** (subroutine *zap_small_areas*), **icepack_mechred.F90** (subroutine *ridge_shift*), **icepack_therm_itd.F90** (subroutines *linear_itd* and *lateral_melt*), and **icepack_therm_vertical.F90** (subroutine *thermo_vertical*), along with global diagnostics in **icepack_diagnostics.F90**.

Level-ice formulation (`tr_pond_lvl` = true)

This meltpond parameterization represents a combination of ideas from the empirical CESM melt pond scheme and the topo approach, and is documented in *[21]*. The ponds evolve according to physically based process descriptions, assuming a thickness-area ratio for changes in pond volume. A novel aspect of the new scheme is that the ponds are carried as tracers on the level (undeformed) ice area of each thickness category, thus limiting their spatial extent based on the simulated sea ice topography. This limiting is meant to approximate the horizontal drainage of melt water into depressions in ice floes. (The primary difference between the level-ice and topo meltpond parameterizations lies in how sea ice topography is taken into account when determining the areal coverage of ponds.) Infiltration of the snow by melt water postpones the appearance of ponds and the subsequent acceleration of melting through albedo feedback, while snow on top of refrozen pond ice also reduces the ponds' effect on the radiation budget.

Melt pond processes, described in more detail below, include addition of liquid water from rain, melting snow and melting surface ice, drainage of pond water when its weight pushes the ice surface below sea level or when the ice interior becomes permeable, and refreezing of the pond water. If snow falls after a layer of ice has formed on the ponds, the snow may block sunlight from reaching the ponds below. When melt water forms with snow still on the ice, the water is assumed to infiltrate the snow. If there is enough water to fill the air spaces within the snowpack, then the pond becomes visible above the snow, thus decreasing the albedo and ultimately causing the snow to melt faster. The albedo also decreases as snow depth decreases, and thus a thin layer of snow remaining above a pond-saturated layer of snow will have a lower albedo than if the melt water were not present.

The level-ice formulation assumes a thickness-area ratio for *changes* in pond volume, while the CESM scheme assumes this ratio for the total pond volume. Pond volume changes are distributed as changes to the area and to the depth of the ponds using an assumed aspect ratio, or shape, given by the parameter δ_p (`pndaspect`), $\delta_p = \Delta h_p/\Delta a_p$ and $\Delta V = \Delta h_p \Delta a_p = \delta_p \Delta a_p^2 = \Delta h_p^2/\delta_p$. Here, $a_p = a_{pnd}a_{lvl}$, the mean pond area over the ice.

Given the ice velocity $\mathbf{u}$, conservation equations for level ice fraction $a_{lvl}a_i$, pond area fraction $a_{pnd}a_{lvl}a_i$, pond volume $h_{pnd}a_{pnd}a_{lvl}a_i$ and pond ice volume $h_{ipnd}a_{pnd}a_{lvl}a_i$ are

$$\frac{\partial}{\partial t}(a_{lvl}a_i) + \nabla \cdot (a_{lvl}a_i\mathbf{u}) = 0, \tag{1.67}$$

$$\frac{\partial}{\partial t}(a_{pnd}a_{lvl}a_i) + \nabla \cdot (a_{pnd}a_{lvl}a_i\mathbf{u}) = 0, \tag{1.68}$$

$$\frac{\partial}{\partial t}(h_{pnd}a_{pnd}a_{lvl}a_i) + \nabla \cdot (h_{pnd}a_{pnd}a_{lvl}a_i\mathbf{u}) = 0, \tag{1.69}$$

$$\frac{\partial}{\partial t}(h_{ipnd}a_{pnd}a_{lvl}a_i) + \nabla \cdot (h_{ipnd}a_{pnd}a_{lvl}a_i\mathbf{u}) = 0. \tag{1.70}$$

(We have dropped the category subscript here, for clarity.) Equations (1.69) and (1.70) express conservation of melt pond volume and pond ice volume, but in this form highlight that the quantities tracked in the code are the tracers h_{pnd} and h_{ipnd}, pond depth and pond ice thickness. Likewise, the level ice fraction a_{lvl} is a tracer on ice area fraction (Equation (1.67)), and pond fraction a_{pnd} is a tracer on level ice (Equation (1.68)).

Pond ice. The ponds are assumed to be well mixed fresh water, and therefore their temperature is 0°C. If the air temperature is cold enough, a layer of clear ice may form on top of the ponds. There are currently three options in the code for refreezing the pond ice. Only option A tracks the thickness of the lid ice using the tracer h_{ipnd} and includes the radiative effect of snow on top of the lid.

A. The `frzpnd` = 'hlid' option uses a Stefan approximation for growth of fresh ice and is invoked only when $\Delta V_{melt} = 0$.

The basic thermodynamic equation governing ice growth is

$$\rho_i L \frac{\partial h_i}{\partial t} = k_i \frac{\partial T_i}{\partial z} \sim k_i \frac{\Delta T}{h_i} \tag{1.71}$$

assuming a linear temperature profile through the ice thickness h_i. In discrete form, the solution is

$$\Delta h_i = \begin{cases} \sqrt{\beta \Delta t}/2 & \text{if } h_i = 0 \\ \beta \Delta t / 2 h_i & \text{if } h_i > 0, \end{cases} \tag{1.72}$$

where

$$\beta = \frac{2 k_i \Delta T}{\rho_i L}. \tag{1.73}$$

When $\Delta V_{melt} > 0$, any existing pond ice may also melt. In this case,

$$\Delta h_i = -\min\left(\frac{\max(F_\circ, 0)\Delta t}{\rho_i L}, h_i\right), \tag{1.74}$$

where $F_\circ$ is the net downward surface flux.

In either case, the change in pond volume associated with growth or melt of pond ice is

$$\Delta V_{frz} = -\Delta h_i a_{pnd} a_{lvl} a_i \rho_i / \rho_0, \tag{1.75}$$

where ρ_0 is the density of fresh water.

B. The `frzpnd` = 'cesm' option uses the same empirical function as in the CESM melt pond parameterization.

Radiative effects. Freshwater ice that has formed on top of a melt pond is assumed to be perfectly clear. Snow may accumulate on top of the pond ice, however, shading the pond and ice below. The depth of the snow on the pond ice is initialized as $h^0_{ps} = F_{snow}\Delta t$ at the first snowfall after the pond ice forms. From that time until either the pond ice or the pond snow disappears, the pond snow depth tracks the depth of snow on sea ice (h_s) using a constant difference Δ. As h_s melts, $h_{ps} = h_s - \Delta$ will be reduced to zero eventually, at which time the pond ice is fully uncovered and shortwave radiation passes through.

To prevent a sudden change in the shortwave reaching the sea ice (which can prevent the thermodynamics from converging), thin layers of snow on pond ice are assumed to be patchy, thus allowing the shortwave flux to increase gradually as the layer thins. This is done using the same parameterization for patchy snow as is used elsewhere in Icepack, but with its own parameter h_{s1}:

$$a^{eff}_{pnd} = \left(1 - \min\left(h_{ps}/h_{s1}, 1\right)\right) a_{pnd} a_{lvl}. \tag{1.76}$$

If any of the pond ice melts, the radiative flux allowed to pass through the ice is reduced by the (roughly) equivalent flux required to melt that ice. This is accomplished (approximately) with $a^{eff}_{pnd} = (1 - f_{frac}) a_{pnd} a_{lvl}$, where (see Equation (1.74))

$$f_{frac} = \min\left(-\frac{\rho_i L \Delta h_i}{F_\circ \Delta t}, 1\right). \tag{1.77}$$

Snow infiltration by pond water. If there is snow on top of the sea ice, melt water may infiltrate the snow. It is a "virtual process" that affects the model's thermodynamics through the input parameters of the radiation scheme; it does not melt the snow or affect the snow heat content.

A snow pack is considered saturated when its percentage of liquid water content is greater or equal to 15% (Sturm and others, 2009). We assume that if the volume fraction of retained melt water to total liquid content

$$r_p = \frac{V_p}{V_p + V_s \rho_s / \rho_0} < 0.15, \tag{1.78}$$

then effectively there are no meltponds present, that is, $a^{eff}_{pnd} = h^{eff}_{pnd} = 0$. Otherwise, we assume that the snowpack is saturated with liquid water.

We assume that all of the liquid water accumulates at the base of the snow pack and would eventually melt the surrounding snow. Two configurations are therefore possible, (1) the top of the liquid lies below the snow surface and (2) the liquid water volume overtops the snow, and all of the snow is assumed to have melted into the pond. The volume of void space within the snow that can be filled with liquid melt water is

$$V_{mx} = h_{mx} a_p = \left(\frac{\rho_0 - \rho_s}{\rho_0}\right) h_s a_p, \tag{1.79}$$

and we compare V_p with V_{mx}.

Case 1: For $V_p < V_{mx}$, we define V_p^{eff} to be the volume of void space filled by the volume V_p of melt water: $\rho_0 V_p = (\rho_0 - \rho_s) V_p^{eff}$, or in terms of depths,

$$h_p^{eff} = \left(\frac{\rho_0}{\rho_0 - \rho_s}\right) h_{pnd}. \tag{1.80}$$

The liquid water under the snow layer is not visible and therefore the ponds themselves have no direct impact on the radiation ($a_{pnd}^{eff} = h_{pnd}^{eff} = 0$), but the effective snow thickness used for the radiation scheme is reduced to

$$h_s^{eff} = h_s - h_p^{eff} a_p = h_s - \frac{\rho_0}{\rho_0 - \rho_s} h_{pnd} a_p. \tag{1.81}$$

Here, the factor $a_p = a_{pnd} a_{lvl}$ averages the reduced snow depth over the ponds with the full snow depth over the remainder of the ice; that is, $h_s^{eff} = h_s(1 - a_p) + (h_s - h_p^{eff}) a_p$.

Case 2: Similarly, for $V_p \geq V_{mx}$, the total mass in the liquid is $\rho_0 V_p + \rho_s V_s = \rho_0 V_p^{eff}$, or

$$h_p^{eff} = \frac{\rho_0 h_{pnd} + \rho_s h_s}{\rho_0}. \tag{1.82}$$

Thus the effective depth of the pond is the depth of the whole slush layer h_p^{eff}. In this case, $a_{pnd}^{eff} = a_{pnd} a_{lvl}$.

Drainage. A portion $1 - r$ of the available melt water drains immediately into the ocean. Once the volume changes described above have been applied and the resulting pond area and depth calculated, the pond depth may be further reduced if the top surface of the ice would be below sea level or if the sea ice becomes permeable.

We require that the sea ice surface remain at or above sea level. If the weight of the pond water would push the mean ice–snow interface of a thickness category below sea level, some or all of the pond water is removed to bring the interface back to sea level via Archimedes' Principle written in terms of the draft d,

$$\rho_i h_i + \rho_s h_s + \rho_0 h_p = \rho_w d \leq \rho_w h_i. \tag{1.83}$$

There is a separate freeboard calculation in the thermodynamics which considers only the ice and snow and converts flooded snow to sea ice. Because the current melt ponds are "virtual" in the sense that they only have a radiative influence, we do not allow the pond mass to change the sea ice and snow masses at this time, although this issue may need to be reconsidered in the future, especially for the Antarctic.

The mushy thermodynamics scheme (*ktherm* = 2) handles flushing. For *ktherm* $\neq 2$, the permeability of the sea ice is calculated using the internal ice temperatures T_i (computed from the enthalpies as in the sea ice thermodynamics). The brine salinity and liquid fraction are given by [35] [eq 3.6] $S_{br} = 1/(10^{-3} - 0.054/T_i)$ and $\phi = S/S_{br}$, where S is the bulk salinity of the combined ice and brine. The ice is considered permeable if $\phi \geq 0.05$ with a permeability of $p = 3 \times 10^{-8} \min(\phi^3)$ (the minimum being taken over all of the ice layers). A hydraulic pressure head is computed as $P = g\rho_w \Delta h$ where Δh is the height of the pond and sea ice above sea level. Then the volume of water drained is given by

$$\Delta V_{perm} = -a_{pnd} \min\left(h_{pnd}, \frac{pPd_p \Delta t}{\mu h_i}\right), \tag{1.84}$$

where d_p is a scaling factor (dpscale), and $\mu = 1.79 \times 10^{-3}$ kg m $^{-1}$ s $^{-1}$ is the dynamic viscosity.

Conservation elsewhere. When ice ridges and when new ice forms in open water, the level ice area changes and ponds must be handled appropriately. For example, when sea ice deforms, some of the level ice is transformed into ridged ice. We assume that pond water (and ice) on the portion of level ice that ridges is lost to the ocean. All of the tracer volumes are altered at this point in the code, even though h_{pnd} and h_{ipnd} should not change; compensating factors in the tracer volumes cancel out (subroutine *ridge_shift* in **icepack_mechred.F90**).

When new ice forms in open water, level ice is added to the existing sea ice, but the new level ice does not yet have ponds on top of it. Therefore the fractional coverage of ponds on level ice decreases (thicknesses are unchanged). This is accomplished in **icepack_therm_itd.F90** (subroutine *add_new_ice*) by maintaining the same mean pond area in a grid cell after the addition of new ice,

$$a'_{pnd}(a_{lvl} + \Delta a_{lvl})(a_i + \Delta a_i) = a_{pnd} a_{lvl} a_i, \tag{1.85}$$

and solving for the new pond area tracer a'_{pnd} given the newly formed ice area $\Delta a_i = \Delta a_{lvl}$.

Thermodynamic surface forcing balance

The net surface energy flux from the atmosphere to the ice (with all fluxes defined as positive downward) is

$$F_0 = F_s + F_l + F_{L\downarrow} + F_{L\uparrow} + (1-\alpha)(1-i_0)F_{sw}, \tag{1.86}$$

where F_s is the sensible heat flux, F_l is the latent heat flux, $F_{L\downarrow}$ is the incoming longwave flux, $F_{L\uparrow}$ is the outgoing longwave flux, F_{sw} is the incoming shortwave flux, α is the shortwave albedo, and i_0 is the fraction of absorbed shortwave flux that penetrates into the ice. The albedo may be altered by the presence of melt ponds. Each of the explicit melt pond parameterizations (CESM, topo and level-ice ponds) should be used in conjunction with the Delta-Eddington shortwave scheme, described below.

Shortwave radiation: Delta-Eddington

Two methods for computing albedo and shortwave fluxes are available, the “ccsm3” method, described below, and a multiple scattering radiative transfer scheme that uses a Delta-Eddington approach. “Inherent” optical properties (IOPs) for snow and sea ice, such as extinction coefficient and single scattering albedo, are prescribed based on physical measurements; reflected, absorbed and transmitted shortwave radiation (“apparent” optical properties) are then computed for each snow and ice layer in a self-consistent manner. Absorptive effects of inclusions in the ice/snow matrix such as dust and algae can also be included, along with radiative treatment of melt ponds and other changes in physical properties, for example granularization associated with snow aging. The Delta-Eddington formulation is described in detail in *[7]*. Since publication of this technical paper, a number of improvements have been made to the Delta-Eddington scheme, including a surface scattering layer and internal shortwave absorption for snow, generalization for multiple snow layers and more than four layers of ice, and updated IOP values.

The namelist parameters `R_ice` and `R_pnd` adjust the albedo of bare or ponded ice by the product of the namelist value and one standard deviation. For example, if `R_ice` = 0.1, the albedo increases by 0.1σ. Similarly, setting `R_snw` = 0.1 decreases the snow grain radius by 0.1σ (thus increasing the albedo). Two additional tuning parameters are available for this scheme, `dT_mlt` and `rsnw_mlt`. `dT_mlt` is the temperature change needed for a change in snow grain radius from non-melting to melting, and `rsnw_mlt` is the maximum snow grain radius when melting. An absorption coefficient for algae (`kalg`) may also be set. See *[7]* for details; the CESM melt pond and Delta-Eddington parameterizations are further explained and validated in *[19]*.

Shortwave radiation: CCSM3

In the parameterization used in the previous version of the Community Climate System Model (CCSM3), the albedo depends on the temperature and thickness of ice and snow and on the spectral distribution of the incoming solar radiation. Albedo parameters have been chosen to fit observations from the SHEBA field experiment. For $T_{sf} < -1^\circ C$ and $h_i >$ “‘$ahmax$‘, the bare ice albedo is 0.78 for visible wavelengths ($<$ 700nm) and 0.36 for near IR wavelengths ($>$ 700nm). As h_i decreases from ahmax to zero, the ice albedo decreases smoothly (using an arctangent function) to the ocean albedo, 0.06. The ice albedo in both spectral bands decreases by 0.075 as T_{sf} rises from $-1^\circ C$ to . The albedo of cold snow ($T_{sf} < -1^\circ C$) is 0.98 for visible wavelengths and 0.70 for near IR wavelengths. The visible snow albedo decreases by 0.10 and the near IR albedo by 0.15 as T_{sf} increases from $-1°C$ to $0°C$. The total albedo is an area-weighted average of the ice and snow albedos, where the fractional snow-covered area is

$$f_{snow} = \frac{h_s}{h_s + h_{snowpatch}}, \tag{1.87}$$

and $h_{snowpatch} = 0.02$ m. The envelope of albedo values is shown in *Figure 7*. This albedo formulation incorporates the effects of melt ponds implicitly; the explicit melt pond parameterization is not used in this case.

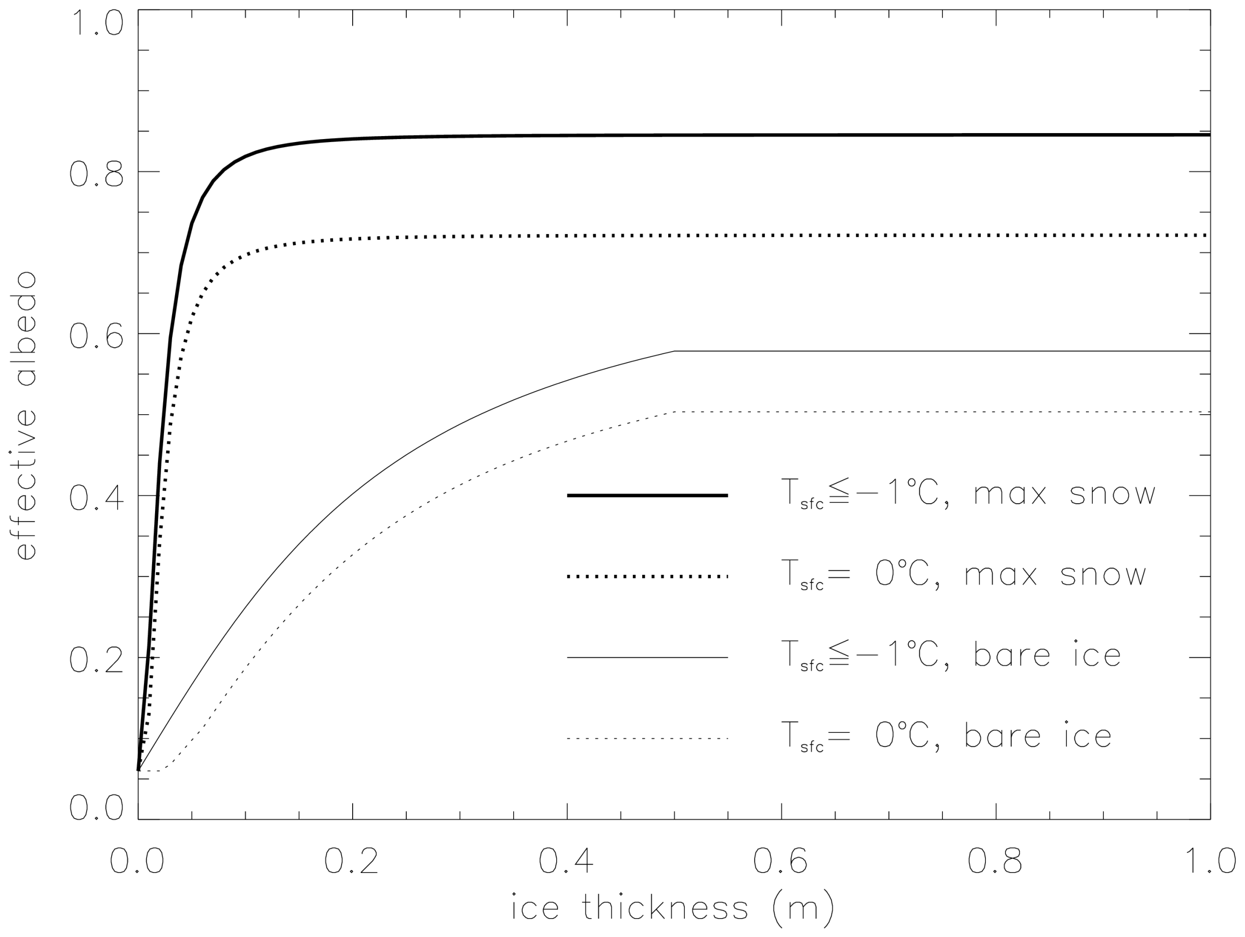

Fig. 1.4: Figure 7

Figure 7 : Albedo as a function of ice thickness and temperature, for the two extrema in snow depth, for the default (CCSM3) shortwave option. Maximum snow depth is computed based on Archimedes' Principle for the given ice thickness. These curves represent the envelope of possible albedo values.

The net absorbed shortwave flux is $F_{swabs} = \sum(1 - \alpha_j)F_{sw\downarrow}$, where the summation is over four radiative categories (direct and diffuse visible, direct and diffuse near infrared). The flux penetrating into the ice is $I_0 = i_0 \, F_{swabs}$, where $i_0 = 0.70 \, (1 - f_{snow})$ for visible radiation and $i_0 = 0$ for near IR. Radiation penetrating into the ice is attenuated according to Beer's Law:

$$I(z) = I_0 \exp(-\kappa_i z), \tag{1.88}$$

where $I(z)$ is the shortwave flux that reaches depth z beneath the surface without being absorbed, and κ_i is the bulk extinction coefficient for solar radiation in ice, set to $1.4 \ \mathrm{m}^{-1}$ for visible wavelengths [8]. A fraction $\exp(-\kappa_i h_i)$ of the penetrating solar radiation passes through the ice to the ocean ($F_{sw\Downarrow}$).

Longwave radiation, turbulent fluxes

While incoming shortwave and longwave radiation are obtained from the atmosphere, outgoing longwave radiation and the turbulent heat fluxes are derived quantities. Outgoing longwave takes the standard blackbody form, $F_{L\uparrow} = \epsilon\sigma\left(T^K_{sf}\right)^4$, where $\epsilon = 0.95$ is the emissivity of snow or ice, σ is the Stefan-Boltzmann constant and T^K_{sf} is the surface temperature in Kelvin. (The longwave fluxes are partitioned such that $\epsilon F_{L\downarrow}$ is absorbed at the surface, the remaining $(1-\epsilon)\,F_{L\downarrow}$ being returned to the atmosphere via $F_{L\uparrow}$.) The sensible heat flux is proportional to the difference between air potential temperature and the surface temperature of the snow or snow-free ice,

$$F_s = C_s\left(\Theta_a - T^K_{sf}\right). \tag{1.89}$$

C_s and C_l (below) are nonlinear turbulent heat transfer coefficients described in the *Atmosphere* section. Similarly, the latent heat flux is proportional to the difference between Q_a and the surface saturation specific humidity Q_{sf}:

$$\begin{aligned} F_l =& \quad C_l\left(Q_a - Q_{sf}\right), \\ Q_{sf} =& \quad (q_1/\rho_a)\exp(-q_2/T^K_{sf}), \end{aligned}$$

where $q_1 = 1.16378 \times 10^7\,\mathrm{kg/m^3}$, $q_2 = 5897.8\,\mathrm{K}$, T^K_{sf} is the surface temperature in Kelvin, and ρ_a is the surface air density.

The net downward heat flux from the ice to the ocean is given by *[32]*:

$$F_{bot} = -\rho_w c_w c_h u_*(T_w - T_f), \tag{1.90}$$

where ρ_w is the density of seawater, c_w is the specific heat of seawater, $c_h = 0.006$ is a heat transfer coefficient, $u_* = \sqrt{\left|\vec{\tau}_w\right|/\rho_w}$ is the friction velocity, and T_w is the sea surface temperature. A minimum value of u_* is available; we recommend $u_{*\,\min} = 5 \times 10^{-4}$ m/s, but the optimal value may depend on the ocean forcing used and can be as low as 0.

F_{bot} is limited by the total amount of heat available from the ocean, F_{frzmlt}. Additional heat, F_{side}, is used to melt the ice laterally following *[33]* and *[42]*. F_{bot} and the fraction of ice melting laterally are scaled so that $F_{bot} + F_{side} \geq F_{frzmlt}$ in the case that $F_{frzmlt} < 0$ (melting; see *Growth and melting*).

New temperatures

Zero-layer thermodynamics (`ktherm` = 0) An option for zero-layer thermodynamics *[40]* is available in this version of Icepack by setting the namelist parameter `ktherm` to 0 and changing the number of ice layers, nilyr, in **icepack_domain_size.F90** to 1. In the zero-layer case, the ice is fresh and the thermodynamic calculations are much simpler than in the other configurations, which we describe here.

Bitz and Lipscomb thermodynamics (`ktherm` = 1)

The "BL99" thermodynamic sea ice model is based on *[34]* and *[6]*, and is described more fully in *[26]*. The vertical salinity profile is prescribed and is unchanging in time. The snow is assumed to be fresh, and the midpoint salinity S_{ik} in each ice layer is given by

$$S_{ik} = \frac{1}{2}S_{\max}[1 - \cos(\pi z^{(\frac{a}{z+b})})], \tag{1.91}$$

where $z \equiv (k - 1/2)/N_i$, $S_{\max} = 3.2$ ppt, and $a = 0.407$ and $b = 0.573$ are determined from a least-squares fit to the salinity profile observed in multiyear sea ice by *[39]*. This profile varies from $S = 0$ at the top surface ($z = 0$) to $S = S_{\max}$ at the bottom surface ($z = 1$) and is similar to that used by *[34]*. Equation (1.91) is fairly accurate for ice that has drained at the top surface due to summer melting. It is not a good approximation for cold first-year ice, which has a more vertically uniform salinity because it has not yet drained. However, the effects of salinity on heat capacity are small for temperatures well below freezing, so the salinity error does not lead to significant temperature errors.

Temperature updates.

Given the temperatures T_{sf}^m, T_s^m, and T_{ik}^m at time m, we solve a set of finite-difference equations to obtain the new temperatures at time $m+1$. Each temperature is coupled to the temperatures of the layers immediately above and below by heat conduction terms that are treated implicitly. For example, the rate of change of T_{ik} depends on the new temperatures in layers $k-1$, k, and $k+1$. Thus we have a set of equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1.92}$$

where $\mathbf{A}$ is a tridiagonal matrix, $\mathbf{x}$ is a column vector whose components are the unknown new temperatures, and $\mathbf{b}$ is another column vector. Given $\mathbf{A}$ and $\mathbf{b}$, we can compute $\mathbf{x}$ with a standard tridiagonal solver.

There are four general cases: (1) $T_{sf} < 0^\circ C$, snow present; (2) $T_{sf} = 0^\circ C$, snow present; (3) $T_{sf} < 0^\circ C$, snow absent; and (4) $T_{sf} = 0^\circ C$, snow absent. For case 1 we have one equation (the top row of the matrix) for the new surface temperature, N_s equations for the new snow temperatures, and N_i equations for the new ice temperatures. For cases 2 and 4 we omit the equation for the surface temperature, which is held at , and for cases 3 and 4 we omit the snow temperature equations. Snow is considered absent if the snow depth is less than a user-specified minimum value, `hs_min`. (Very thin snow layers are still transported conservatively by the transport modules; they are simply ignored by the thermodynamics.)

The rate of temperature change in the ice interior is given by *[34]*:

$$\rho_i c_i \frac{\partial T_i}{\partial t} = \frac{\partial}{\partial z}\left(K_i \frac{\partial T_i}{\partial z}\right) - \frac{\partial}{\partial z}[I_{pen}(z)], \tag{1.93}$$

where $\rho_i = 917$ kg/m^3 is the sea ice density (assumed to be uniform), $c_i(T,S)$ is the specific heat of sea ice, $K_i(T,S)$ is the thermal conductivity of sea ice, I_{pen} is the flux of penetrating solar radiation at depth z, and z is the vertical coordinate, defined to be positive downward with $z = 0$ at the top surface. If `shortwave` = 'default', the penetrating radiation is given by Beer's Law:

$$I_{pen}(z) = I_0 \exp(-\kappa_i z),$$

where I_0 is the penetrating solar flux at the top ice surface and κ_i is an extinction coefficient. If `shortwave` = 'dEdd', then solar absorption is computed by the Delta-Eddington scheme.

The specific heat of sea ice is given to an excellent approximation by *[36]*

$$c_i(T,S) = c_0 + \frac{L_0 \mu S}{T^2}, \tag{1.94}$$

where $c_0 = 2106$ J/kg/deg is the specific heat of fresh ice at , $L_0 = 3.34 \times 10^5$ J/kg is the latent heat of fusion of fresh ice at , and $\mu = 0.054$ deg/ppt is the (liquidus) ratio between the freezing temperature and salinity of brine.

Following *[48]* and *[34]*, the standard thermal conductivity (`conduct` = 'MU71') is given by

$$K_i(T,S) = K_0 + \frac{\beta S}{T}, \tag{1.95}$$

where $K_0 = 2.03$ W/m/deg is the conductivity of fresh ice and $\beta = 0.13$ W/m/ppt is an empirical constant. Experimental results *[45]* suggest that Equation (1.95) may not be a good description of the thermal conductivity of sea ice. In particular, the measured conductivity does not markedly decrease as T approaches $0^\circ C$, but does decrease near the top surface (regardless of temperature).

An alternative parameterization based on the "bubbly brine" model of *[37]* for conductivity is available (`conduct` = 'bubbly'):

$$K_i = \frac{\rho_i}{\rho_0}\left(2.11 - 0.011T + 0.09S/T\right), \tag{1.96}$$

where ρ_i and $\rho_0 = 917$ kg/m ^{3} are densities of sea ice and pure ice. Whereas the parameterization in Equation (1.95) asymptotes to a constant conductivity of 2.03 W m^{-1} K $^{-1}$ with decreasing T, K_i in Equation (1.96) continues to increase with colder temperatures.

The equation for temperature changes in snow is analogous to Equation (1.93), with $\rho_s = 330$ kg/m ^{3}, $c_s = c_0$, and $K_s = 0.30$ W/m/deg replacing the corresponding ice values. If shortwave = 'default', then the penetrating solar radiation is equal to zero for snow-covered ice, since most of the incoming sunlight is absorbed near the top surface. If shortwave = 'dEdd', however, then I_{pen} is nonzero in snow layers.

It is possible that more shortwave penetrates into an ice layer than is needed to completely melt the layer, or else it causes the computed temperature to be greater than the melting temperature, which until now has caused the vertical thermodynamics code to abort. A parameter `frac` = 0.9 sets the fraction of the ice layer than can be melted through. A minimum temperature difference for absorption of radiation is also set, currently `dTemp` = 0.02 (K). The limiting occurs in **icepack_therm_vertical.F90**, for both the default and delta Eddington radiation schemes. If the available energy would melt through a layer, then penetrating shortwave is first reduced, possibly to zero, and if that is insufficient then the local conductivity is also reduced to bring the layer temperature just to the melting point.

We now convert Equation (1.93) to finite-difference form. The resulting equations are second-order accurate in space, except possibly at material boundaries, and first-order accurate in time. Before writing the equations in full we give finite-difference expressions for some of the terms.

First consider the terms on the left-hand side of Equation (1.93). We write the time derivatives as

$$\frac{\partial T}{\partial t} = \frac{T^{m+1} - T^m}{\Delta t},$$

where T is the temperature of either ice or snow and m is a time index. The specific heat of ice layer k is approximated as

$$c_{ik} = c_0 + \frac{L_0 \mu S_{ik}}{T_{ik}^m T_{ik}^{m+1}}, \qquad (1.97)$$

which ensures that energy is conserved during a change in temperature. This can be shown by using Equation (1.94) to integrate $c_i\, dT$ from T_{ik}^m to T_{ik}^{m+1}; the result is $c_{ik}(T_{ik}^{m+1} - T_{ik}^m)$, where c_{ik} is given by Equation (1.97). The specific heat is a nonlinear function of T_{ik}^{m+1}, the unknown new temperature. We can retain a set of linear equations, however, by initially guessing $T_{ik}^{m+1} = T_{ik}^m$ and then iterating the solution, updating T_{ik}^{m+1} in Equation (1.97) with each iteration until the solution converges.

Next consider the first term on the right-hand side of Equation (1.93). The first term describes heat diffusion and is discretized for a given ice or snow layer k as

$$\frac{\partial}{\partial z}\left(K \frac{\partial T}{\partial z}\right) = \frac{1}{\Delta h}\left[K_k^*(T_{k-1}^{m+1} - T_k^{m+1}) - K_{k+1}^*(T_k^{m+1} - T_{k+1}^{m+1})\right], \qquad (1.98)$$

where Δh is the layer thickness and K_k is the effective conductivity at the upper boundary of layer k. This discretization is centered and second-order accurate in space, except at the boundaries. The flux terms on the right-hand side (RHS) are treated implicitly; i.e., they depend on the temperatures at the new time $m + 1$. The resulting scheme is first-order accurate in time and unconditionally stable. The effective conductivity K^* at the interface of layers $k - 1$ and k is defined as

$$K_k^* = \frac{2K_{k-1}K_k}{K_{k-1}h_k + K_k h_{k-1}},$$

which reduces to the appropriate values in the limits $K_k \gg K_{k-1}$ (or vice versa) and $h_k \gg h_{k-1}$ (or vice versa). The effective conductivity at the top (bottom) interface of the ice-snow column is given by $K^* = 2K/\Delta h$, where K and Δh are the thermal conductivity and thickness of the top (bottom) layer. The second term on the RHS of Equation (1.93) is discretized as

$$\frac{\partial}{\partial z}\left[I_{pen}(z)\right] = I_0 \frac{\tau_{k-1} - \tau_k}{\Delta h} = \frac{I_k}{\Delta h}$$

where τ_k is the fraction of the penetrating solar radiation I_0 that is transmitted through layer k without being absorbed.

We now construct a system of equations for the new temperatures. For $T_{sf} < 0°C$ we require

$$F_0 = F_{ct}, \qquad (1.99)$$

where F_{ct} is the conductive flux from the top surface to the ice interior, and both fluxes are evaluated at time $m+1$. Although F_0 is a nonlinear function of T_{sf}, we can make the linear approximation

$$F_0^{m+1} = F_0^* + \left(\frac{dF_0}{dT_{sf}}\right)^* (T_{sf}^{m+1} - T_{sf}^*),$$

where T_{sf}^* is the surface temperature from the most recent iteration, and F_0^* and $(dF_0/dT_{sf})^*$ are functions of T_{sf}^*. We initialize $T_{sf}^* = T_{sf}^m$ and update it with each iteration. Thus we can rewrite Equation (1.99) as

$$F_0^* + \left(\frac{dF_0}{dT_{sf}}\right)^* (T_{sf}^{m+1} - T_{sf}^*) = K_1^*(T_{sf}^{m+1} - T_1^{m+1}),$$

Rearranging terms, we obtain

$$\left[\left(\frac{dF_0}{dT_{sf}}\right)^* - K_1^*\right] T_{sf}^{m+1} + K_1^* T_1^{m+1} = \left(\frac{dF_0}{dT_{sf}}\right)^* T_{sf}^* - F_0^*, \tag{1.100}$$

the first equation in the set of equations (1.92). The temperature change in ice/snow layer k is

$$\rho_k c_k \frac{(T_k^{m+1} - T_k^m)}{\Delta t} = \frac{1}{\Delta h_k}[K_k^*(T_{k-1}^{m+1} - T_k^{m+1}) - K_{k+1}(T_k^{m+1} - T_{k+1}^{m+1})], \tag{1.101}$$

where $T_0 = T_{sf}$ in the equation for layer 1. In tridiagonal matrix form, Equation (1.101) becomes

$$-\eta_k K_k T_{k-1}^{m+1} + [1 + \eta_k(K_k + K_{k+1})] T_k^{m+1} - \eta_k K_{k+1} T_{k+1}^{m+1} = T_k^m + \eta_k I_k, \tag{1.102}$$

where $\eta_k = \Delta t/(\rho_k c_k \Delta h_k)$. In the equation for the bottom ice layer, the temperature at the ice–ocean interface is held fixed at T_f, the freezing temperature of the mixed layer; thus the last term on the LHS is known and is moved to the RHS. If $T_{sf} = 0°C$, then there is no surface flux equation. In this case the first equation in Equation (1.92) is similar to Equation (1.102), but with the first term on the LHS moved to the RHS.

These equations are modified if T_{sf} and F_{ct} are computed within the atmospheric model and passed to the host sea ice model (calc_Tsfc = false; see *Atmosphere*). In this case there is no surface flux equation. The top layer temperature is computed by an equation similar to Equation (1.102) but with the first term on the LHS replaced by $\eta_1 F_{ct}$ and moved to the RHS. The main drawback of treating the surface temperature and fluxes explicitly is that the solution scheme is no longer unconditionally stable. Instead, the effective conductivity in the top layer must satisfy a diffusive CFL condition:

$$K^* \le \frac{\rho c h}{\Delta t}.$$

For thin layers and typical coupling intervals ($\sim$ 1 hr), K^* may need to be limited before being passed to the atmosphere via the coupler. Otherwise, the fluxes that are returned to the host sea ice model may result in oscillating, highly inaccurate temperatures. The effect of limiting is to treat the ice as a poor heat conductor. As a result, winter growth rates are reduced, and the ice is likely to be too thin (other things being equal). The values of `hs_min` and Δt must therefore be chosen with care. If `hs_min` is too small, frequent limiting is required, but if `hs_min` is too large, snow will be ignored when its thermodynamic effects are significant. Likewise, infrequent coupling requires more limiting, whereas frequent coupling is computationally expensive.

This completes the specification of the matrix equations for the four cases. We compute the new temperatures using a tridiagonal solver. After each iteration we check to see whether the following conditions hold:

1. $T_{sf} \le 0°C.$
2. The change in T_{sf} since the previous iteration is less than a prescribed limit, $\Delta T_{\max}$.
3. $F_0 \ge F_{ct}$. (If $F_0 < F_{ct}$, ice would be growing at the top surface, which is not allowed.)
4. The rate at which energy is added to the ice by the external fluxes equals the rate at which the internal ice energy is changing, to within a prescribed limit $\Delta F_{\max}$.

We also check the convergence rate of T_{sf}. If T_{sf} is oscillating and failing to converge, we average temperatures from successive iterations to improve convergence. When all these conditions are satisfied—usually within two to four iterations for $\Delta T_{\max} \approx 0.01^\circ C$ and $\Delta F_{max} \approx 0.01$ W/m^2—the calculation is complete.

To compute growth and melt rates (*Growth and melting*, we derive expressions for the enthalpy q. The enthalpy of snow (or fresh ice) is given by

$$q_s(T) = -\rho_s(-c_0 T + L_0).$$

Sea ice enthalpy is more complex, because of brine pockets whose salinity varies inversely with temperature. Since the salinity is prescribed, there is a one-to-one relationship between temperature and enthalpy. The specific heat of sea ice, given by Equation (1.94), includes not only the energy needed to warm or cool ice, but also the energy used to freeze or melt ice adjacent to brine pockets. Equation (1.94) can be integrated to give the energy δ_e required to raise the temperature of a unit mass of sea ice of salinity S from T to T':

$$\delta_e(T, T') = c_0(T' - T) + L_0 \mu S \left(\frac{1}{T} - \frac{1}{T'}\right).$$

If we let $T' = T_m \equiv -\mu S$, the temperature at which the ice is completely melted, we have

$$\delta_e(T, T_m) = c_0(T_m - T) + L_0 \left(1 - \frac{T_m}{T}\right).$$

Multiplying by ρ_i to change the units from J/kg to J/m^3 and adding a term for the energy needed to raise the meltwater temperature to , we obtain the sea ice enthalpy:

$$q_i(T, S) = -\rho_i \left[c_0(T_m - T) + L_0 \left(1 - \frac{T_m}{T}\right) - c_w T_m.\right] \tag{1.103}$$

Note that Equation (1.103) is a quadratic equation in T. Given the layer enthalpies we can compute the temperatures using the quadratic formula:

$$T = \frac{-b - \sqrt{b^2 - 4ac}}{2a},$$

where

$$\begin{aligned} a = & \quad c_0, \\ b = & \quad (c_w - c_0)\, T_m - \frac{q_i}{\rho_i} - L_0, \\ c = & \quad L_0 T_m. \end{aligned}$$

The other root is unphysical.

Mushy thermodynamics (`ktherm` = 2)

The "mushy" thermodynamics option treats the sea ice as a mushy layer *[10]* in which the ice is assumed to be composed of microscopic brine inclusions surrounded by a matrix of pure water ice. Both enthalpy and salinity are prognostic variables. The size of the brine inclusions is assumed to be much smaller than the size of the ice layers, allowing a continuum approximation: a bulk sea-ice quantity is taken to be the liquid-fraction-weighted average of that quantity in the ice and in the brine.

Enthalpy and mushy physics.

The mush enthalpy, q, is related to the temperature, T, and the brine volume, ϕ, by

$$q = \phi q_{br} \quad + (1 - \phi) q_i = \phi \rho_w c_w T \quad + (1 - \phi)(\rho_i c_i T - \rho_i L_0) \tag{1.104}$$

where q_{br} is the brine enthalpy, q_i is the pure ice enthalpy, ρ_i and c_i are density and heat capacity of the ice, ρ_w and c_w are density and heat capacity of the brine and L_0 is the latent heat of melting of pure ice. We assume that the specific heats of the ice and brine are fixed at the values of cp_ice and cp_ocn, respectively. The enthalpy is the energy required to raise the temperature of the sea ice to , including both sensible and latent heat changes. Since the sea ice contains salt, it usually will be fully melted at a temperature below $0°C$. Equations (1.103) and (1.104) are equivalent except for the density used in the term representing the energy required to bring the melt water temperature to (ρ_i and ρ_w in equations (1.103) and (1.104), respectively).

The liquid fraction, ϕ, of sea ice is given by

$$\phi = \frac{S}{S_{br}}$$

where the brine salinity, S_{br}, is given by the liquidus relation using the ice temperature.

Within the parameterizations of brine drainage the brine density is a function of brine salinity *[35]*:

$$\rho(S_{br}) = 1000.3 + 0.78237 S_{br} + 2.8008 \times 10^{-4} S_{br}^2.$$

Outside the parameterizations of brine drainage the densities of brine and ice are fixed at the values of ρ_w and ρ_i, respectively.

The permeability of ice is computed from the liquid fraction as in *[16]*:

$$\Pi(\phi) = 3 \times 10^{-8} (\phi - \phi_\Pi)^3$$

where ϕ_Π is 0.05.

The liquidus relation used in the mushy layer module is based on observations of *[4]*. A piecewise linear relation can be fitted to observations of Z (the ratio of mass of salt (in g) to mass of pure water (in kg) in brine) to the melting temperature: $Z = aT + b$. Salinity is the mass of salt (in g) per mass of brine (in kg) so is related to Z by

$$\frac{1}{S} = \frac{1}{1000} + \frac{1}{Z}.$$

The data is well fitted with two linear regions,

$$S_{br} = \frac{(T + J_1)}{(T/1000 + L_1)} l_0 + \frac{(T + J_2)}{(T/1000 + L_2)} (1 - l_0)$$

where

$$l_0 = \begin{cases} 1 & \text{if} \quad T \geq T_0 \\ 0 & \text{if} \quad T < T_0 \end{cases},$$

$$J_{1,2} = \frac{b_{1,2}}{a_{1,2}},$$

$$L_{1,2} = \frac{(1 + b_{1,2}/1000)}{a_{1,2}}.$$

T_0 is the temperature at which the two linear regions meet. Fitting to the data, $T_0 = -7.636$°C, $a_1 = -18.48$ g kg^{-1} K^{-1}, $a_2 = -10.3085$ g kg^{-1} K^{-1}, $b_1 = 0$ and $b_2 = 62.4$ g kg^{-1}.

Two stage outer iteration.

As for the BL99 thermodynamics *[6]* there are two qualitatively different situations that must be considered when solving for the vertical thermodynamics: the surface can be melting and at the melting temperature, or the surface can be colder than the melting temperature and not melting. In the BL99 thermodynamics these two situations were treated within the same iterative loop, but here they are dealt with separately. If at the beginning of the time step the ice surface is cold and not melting, we solve the ice temperatures assuming that this is also true at the end of the time step. Once we have solved for the new temperatures we test to see if the answer is consistent with this assumption. If the surface temperature is below the melting temperature then we have found the appropriate consistent solution. If the surface is above the melting temperature at the end of the initial solution attempt, we recalculate the new temperatures assuming the surface temperature is fixed at the melting temperature. Alternatively if the surface is at the melting temperature at the start of a time step, we assume initially that this is also the case at the end of the time step, solve for the new temperatures and then check that the surface conductive heat flux is less than the surface atmospheric heat flux as is required for a melting surface. If this is not the case, the temperatures are recalculated assuming the surface is colder than melting. We have found that solutions of the temperature equations that only treat one of the two qualitatively different solutions at a time are more numerically robust than if both are solved together. The surface state rarely changes qualitatively during the solution so the method is also numerically efficient.

Temperature updates.

During the calculation of the new temperatures and salinities, the liquid fraction is held fixed at the value from the previous time step. Updating the liquid fraction during the Picard iteration described below was found to be numerically unstable. Keeping the liquid fraction fixed drastically improves the numerical stability of the method without significantly changing the solution.

Temperatures are calculated in a similar way to BL99 with an outer Picard iteration of an inner tridiagonal matrix solve. The conservation equation for the internal ice temperatures is

$$\frac{\partial q}{\partial t} = \frac{\partial}{\partial z}\left(K\frac{\partial T}{\partial z}\right) + w\frac{\partial q_{br}}{\partial z} + F$$

where q is the sea ice enthalpy, K is the bulk thermal conductivity of the ice, w is the vertical Darcy velocity of the brine, q_{br} is the brine enthalpy and F is the internally absorbed shortwave radiation. The first term on the right represents heat conduction and the second term represents the vertical advection of heat by gravity drainage and flushing.

The conductivity of the mush is given by

$$K = \phi K_{br} + (1-\phi)K_i$$

where $K_i = 2.3$ Wm:math:*^{-1}‘K:math:‘^{-1}* is the conductivity of pure ice and $K_{br} = 0.5375$ Wm:math:*^{-1}‘K:math:‘^{-1}* is the conductivity of the brine. The thermal conductivity of brine is a function of temperature and salinity, but here we take it as a constant value for the middle of the temperature range experienced by sea ice, $-10°$C *[41]*, assuming the brine liquidus salinity at $-10°$C.

We discretize the terms that include temperature in the heat conservation equation as

$$\frac{q_k^t - q_k^{t_0}}{\Delta t} = \frac{\frac{K_{k+1}^*}{\Delta z_{k+1}'}(T_{k+1}^t - T_k^t) - \frac{K_k^*}{\Delta z_k'}(T_k^t - T_{k-1}^t)}{\Delta h} \tag{1.105}$$

where the superscript signifies whether the quantity is evaluated at the start (t_0) or the end (t) of the time step and the subscript indicates the vertical layer. Writing out the temperature dependence of the enthalpy term we have

$$\frac{\left(\phi(c_w\rho_w - c_i\rho_i) + c_i\rho_i\right)T_k^t - (1-\phi)\rho_i L - q_k^{t_0}}{\Delta t} = \frac{\frac{K_{k+1}^*}{\Delta z_{k+1}'}(T_{k+1}^t - T_k^t) - \frac{K_k^*}{\Delta z_k'}(T_k^t - T_{k-1}^t)}{\Delta h}.$$

The mush thermal conductivities are fixed at the start of the timestep. For the lowest ice layer T_{k+1} is replaced with T_{bot}, the temperature of the ice base. Δh is the layer thickness and z_k' is the distance between the $k-1$ and k layer centers.

Similarly, for the snow layer temperatures we have the following discretized equation:

$$\frac{c_i\rho_s T_k^t - \rho_s L_0 - q_k^{t_0}}{\Delta t} = \frac{\frac{K_{k+1}^*}{\Delta z_{k+1}'}(T_{k+1}^t - T_k^t) - \frac{K_k^*}{\Delta z_k'}(T_k^t - T_{k-1}^t)}{\Delta h}.$$

For the upper-most layer (either ice layer or snow layer if it present) T_{k-1} is replaced with T_{sf}, the temperature of the surface.

If the surface is colder than the melting temperature then we also have to solve for the surface temperature, T_{sf}. Here we follow the methodology of BL99 described above.

These discretized temperature equations form a tridiagional matrix for the new temperatures and are solved with a standard tridiagonal solver. A Picard iteration is used to incorporate nonlinearity in the equations. The surface heat flux is a function of surface temperature and with each iteration, the surface heat flux is calculated with the new surface temperature until convergence is achieved. Convergence normally occurs after a few iterations once the temperature changes during an iteration fall below 5×10^{-4} °C and the energy conservation error falls below 0.9‘‘ferrmax‘‘.

Salinity updates.

Several physical processes alter the sea ice bulk salinity. New ice forms with the salinity of the sea water from which it formed. Gravity drainage reduces the bulk salinity of newly formed sea ice, while flushing of melt water through the ice also alters the salinity profile.

The salinity equation takes the form

$$\frac{\partial S}{\partial t} = w\frac{\partial S_{br}}{\partial z} + G$$

where w is a vertical Darcy velocity and G is a source term. The right-hand side depends indirectly on the bulk salinity through the liquid fraction ($S = \phi S_{br}$). Since ϕ is fixed for the time step, we solve the salinity equation explicitly after the temperature equation is solved.

A. Gravity drainage. Sea ice initially retains all the salt present in the sea water from which it formed. Cold temperatures near the top surface of forming sea ice result in higher brine salinities there, because the brine is always at its melting temperature. This colder, saltier brine is denser than the underlying sea water and the brine undergoes convective overturning with the ocean. As the dense, cold brine drains out of the ice, it is replaced by fresher seawater, lowering the bulk salinity of the ice. Following [47], gravity drainage is assumed to occur as two simultaneously operating modes: a rapid mode operating principally near the ice base and a slow mode occurring everywhere.

Rapid drainage takes the form of a vertically varying upward Darcy flow. The contribution to the bulk salinity equation for the rapid mode is

$$\left.\frac{\partial S}{\partial t}\right|_{rapid} = w(z)\frac{\partial S_{br}}{\partial z}$$

where S is the bulk salinity and B_{br} is the brine salinity, specified by the liquidus relation with ice temperature. This equation is discretized using an upwind advection scheme,

$$\frac{S_k^t - S_k^{t_0}}{\Delta t} = w_k \frac{S_{brk+1} - S_{brk}}{\Delta z}.$$

The upward advective flow also carries heat, contributing a term to the heat conservation Equation (1.105),

$$\left.\frac{\partial q}{\partial t}\right|_{rapid} = w(z)\frac{\partial q_{br}}{\partial z}$$

where q_{br} is the brine enthalpy. This term is discretized as

$$\left.\frac{q_k^t - q_k^{t_0}}{\Delta t}\right|_{rapid} = w_k \frac{q_{br\,k+1} - q_{br\,k}}{\Delta z}.$$

$$w_k = \max_{j=k,n}\left(\tilde{w}_j\right)$$

where the maximum is taken over all the ice layers between layer k and the ice base. $\tilde{w}_j$ is given by

$$\tilde{w}(z) = w\left(\frac{Ra(z) - Ra_c}{Ra(z)}\right). \tag{1.106}$$

where Ra_c is a critical Rayleigh number and $Ra(z)$ is the local Rayleigh number at a particular level,

$$Ra(z) = \frac{g\Delta\rho\Pi(h-z)}{\kappa\eta}$$

where $\Delta\rho$ is the difference in density between the brine at z and the ocean, Π is the minimum permeability between z and the ocean, h is the ice thickness, κ is the brine thermal diffusivity and η is the brine dynamic viscosity. Equation ([eq:mushyvel]) reduces the flow rate for Rayleigh numbers below the critical Rayleigh number.

The unmodified flow rate, w, is determined from a hydraulic pressure balance argument for upward flow through the mush and returning downward flow through ice free channels:

$$w(z)\Delta x^2 = A_m\left(-\frac{\Delta P}{l} + B_m\right)$$

where

$$\begin{aligned}
\frac{\Delta P}{l} &= \frac{A_pB_p + A_mB_m}{A_m + A_p}, \\
A_m &= \frac{\Delta x^2}{\eta}\frac{n}{\sum_{k=1}^{n}\frac{1}{\Pi(k)}}, \\
B_m &= -\frac{g}{n}\sum_{k=1}^{n}\rho(k), \\
A_p &= \frac{\pi a^4}{8\eta}, \\
B_p &= -\rho_p g.
\end{aligned}$$

There are three tunable parameters in the above parameterization, a, the diameter of the channel, Δx, the horizontal size of the mush draining through each channel, and Ra_c, the critical Rayleigh number. ρ_p is the density of brine in the channel which we take to be the density of brine in the mush at the level that the brine is draining from. l is the thickness of mush from the ice base to the top of the layer in question. We assume that Δx is proportional to l so that $\Delta x = 2\beta l$. a (`a_rapid_mode`), β (`aspect_rapid_mode`) and Ra_c (`Ra_c_rapid_mode`) are all namelist parameters with default values of 0.5 mm, 1 and 10, respectively. The value $\beta = 1$ gives a square aspect ratio for the convective flow in the mush.

The *slow drainage* mode takes the form of a simple relaxation of bulk salinity:

$$\left.\frac{\partial S(z)}{\partial t}\right|_{slow} = -\lambda(S(z) - S_c).$$

The decay constant, λ, is modeled as

$$\lambda = S^* \max\left(\frac{T_{bot} - T_{sf}}{h}, 0\right)$$

where S^* is a tuning parameter for the drainage strength, T_{bot} is the basal ice temperature, T_{sf} is the upper surface temperature and h is the ice thickness. The bulk salinity relaxes to a value, $S_c(z)$, given by

$$S_c(z) = \phi_c S_{br}(z)$$

where $S_{br}(z)$ is the brine salinity at depth z and ϕ_c is a critical liquid fraction. Both S^* and ϕ_c are namelist parameters, `dSdt_slow_mode` $= 1.5 \times 10^{-7}$ m s^{-1} K^{-1} and `phi_c_slow_mode` $= 0.05$.

B. Downwards flushing. Melt pond water drains through sea ice and flushes out brine, reducing the bulk salinity of the sea ice. This is modeled with the mushy physics option as a vertical Darcy flow through the ice that affects both the enthalpy and bulk salinity of the sea ice:

$$\left.\frac{\partial q}{\partial t}\right|_{flush} = w_f\frac{\partial q_{br}}{\partial z}$$

$$\left.\frac{\partial S}{\partial t}\right|_{flush} = w_f \frac{\partial S_{br}}{\partial z}$$

These equations are discretized with an upwind advection scheme. The flushing Darcy flow, w_f, is given by

$$w_f = \frac{\overline{\Pi}\rho_w g \Delta h}{h\eta},$$

where $\overline{\Pi}$ is the harmonic mean of the ice layer permeabilities and Δh is the hydraulic head driving melt water through the sea ice. It is the difference in height between the top of the melt pond and sea level.

Basal boundary condition.

In traditional Stefan problems the ice growth rate is calculated by determining the difference in heat flux on either side of the ice/ocean interface and equating this energy difference to the latent heat of new ice formed. Thus,

$$(1-\phi_i)L_0\rho_i\frac{\partial h}{\partial t} = K\left.\frac{\partial T}{\partial z}\right|_i - K_w\left.\frac{\partial T}{\partial z}\right|_w \tag{1.107}$$

where $(1-\phi_i)$ is the solid fraction of new ice formed and the right hand is the difference in heat flux at the ice–ocean interface between the ice side and the ocean side of the interface. However, with mushy layers there is usually no discontinuity in solid fraction across the interface, so $\phi_i = 1$ and Equation (1.107) cannot be used explicitly. To circumvent this problem we set the interface solid fraction to be 0.15, a value that reproduces observations. ϕ_i is a namelist parameter (`phi_i_mushy` = 0.85). The basal ice temperature is set to the liquidus temperature T_f of the ocean surface salinity.

Tracer consistency.

In order to ensure conservation of energy and salt content, the advection routines will occasionally limit changes to either enthalpy or bulk salinity. The mushy thermodynamics routine determines temperature from both enthalpy and bulk salinity. Since the limiting changes performed in the advection routine are not applied consistently (from a mushy physics point of view) to both enthalpy and bulk salinity, the resulting temperature may be changed to be greater than the limit allowed in the thermodynamics routines. If this situation is detected, the code corrects the enthalpy so the temperature is below the limiting value. Conservation of energy is ensured by placing the excess energy in the ocean, and the code writes a warning that this has occurred to the diagnostics file. This situation only occurs with the mushy thermodynamics, and it should only occur very infrequently and have a minimal effect on results. The addition of the heat to the ocean may reduce ice formation by a small amount afterwards.

Growth and melting

Melting at the top surface is given by

$$q\,\delta h = \begin{cases} (F_0 - F_{ct})\,\Delta t & \text{if } F_0 > F_{ct} \\ 0 & \text{otherwise} \end{cases} \tag{1.108}$$

where q is the enthalpy of the surface ice or snow layer[1] (recall that $q < 0$) and δh is the change in thickness. If the layer melts completely, the remaining flux is used to melt the layers beneath. Any energy left over when the ice and snow are gone is added to the ocean mixed layer. Ice cannot grow at the top surface due to conductive fluxes; however, snow–ice can form. New snowfall is added at the end of the thermodynamic time step.

Growth and melting at the bottom ice surface are governed by

$$q\,\delta h = (F_{cb} - F_{bot})\,\Delta t, \tag{1.109}$$

where F_{bot} is given by Equation (1.90) and F_{cb} is the conductive heat flux at the bottom surface:

$$F_{cb} = \frac{K_{i,N+1}}{\Delta h_i}(T_{iN} - T_f).$$

[1] The mushy thermodynamics option does not include the enthalpy associated with raising the meltwater temperature to in these calculations, unlike BL99, which does include it. This extra heat is returned to the ocean (or the atmosphere, in the case of evaporation) with the melt water.

If ice is melting at the bottom surface, q in Equation (1.109) is the enthalpy of the bottom ice layer. If ice is growing, q is the enthalpy of new ice with temperature T_f and salinity S_{max} (`ktherm` = 1) or ocean surface salinity (`ktherm` = 2). This ice is added to the bottom layer.

In general, frazil ice formed in the ocean is added to the thinnest ice category. The new ice is grown in the open water area of the grid cell to a specified minimum thickness; if the open water area is nearly zero or if there is more new ice than will fit into the thinnest ice category, then the new ice is spread over the entire cell.

If the latent heat flux is negative (i.e., latent heat is transferred from the ice to the atmosphere), snow or snow-free ice sublimates at the top surface. If the latent heat flux is positive, vapor from the atmosphere is deposited at the surface as snow or ice. The thickness change of the surface layer is given by

$$(\rho L_v - q)\delta h = F_l \Delta t, \tag{1.110}$$

where ρ is the density of the surface material (snow or ice), and $L_v = 2.501\times 10^6$ J/kg is the latent heat of vaporization of liquid water at . Note that ρL_v is nearly an order of magnitude larger than typical values of q. For positive latent heat fluxes, the deposited snow or ice is assumed to have the same enthalpy as the existing surface layer.

After growth and melting, the various ice layers no longer have equal thicknesses. We therefore adjust the layer interfaces, conserving energy, so as to restore layers of equal thickness $\Delta h_i = h_i/N_i$. This is done by computing the overlap η_{km} of each new layer k with each old layer m:

$$\eta_{km} = \min(z_m, z_k) - \max(z_{m-1}, z_{k-1}),$$

where z_m and z_k are the vertical coordinates of the old and new layers, respectively. The enthalpies of the new layers are

$$q_k = \frac{1}{\Delta h_i} \sum_{m=1}^{N_i} \eta_{km} q_m.$$

Lateral melting is accomplished by multiplying the state variables by $1 - r_{side}$, where r_{side} is the fraction of ice melted laterally *[33][42]*, and adjusting the ice energy and fluxes as appropriate. We assume a floe diameter of 300 m.

Snow ice formation.

At the end of the time step we check whether the snow is deep enough to lie partially below the surface of the ocean (freeboard). From Archimedes' principle, the base of the snow is at sea level when

$$\rho_i h_i + \rho_s h_s = \rho_w h_i.$$

Thus the snow base lies below sea level when

$$h^* \equiv h_s - \frac{(\rho_w - \rho_i) h_i}{\rho_s} > 0.$$

In this case, for `ktherm` = 1 (BL99) we raise the snow base to sea level by converting some snow to ice:

$$\begin{aligned} \delta h_s &= \frac{-\rho_i h^*}{\rho_w}, \\ \delta h_i &= \frac{\rho_s h^*}{\rho_w}. \end{aligned}$$

In rare cases this process can increase the ice thickness substantially. For this reason snow–ice conversions are postponed until after the remapping in thickness space (*Transport in thickness space*), which assumes that ice growth during a single time step is fairly small.

For `ktherm` = 2 (mushy), we model the snow–ice formation process as follows: If the ice surface is below sea level then we replace some snow with the same thickness of sea ice. The thickness change chosen is that which brings the ice surface to sea level. The new ice has a porosity of the snow, which is calculated as

$$\phi = 1 - \frac{\rho_s}{\rho_i}$$

where ρ_s is the density of snow and ρ_i is the density of fresh ice. The salinity of the brine occupying the above porosity within the new ice is taken as the sea surface salinity. Once the new ice is formed, the vertical ice and snow layers are regridded into equal thicknesses while conserving energy and salt.

Biogeochemistry

From: Nicole Jeffery, Scott Elliott, Elizabeth C. Hunke, William H. Lipscomb, and Adrian K. Turner default aerosols from: David Baily, Marika Holland… others?

Aerosols

Default Aerosols

Aerosols may be deposited on the ice and gradually work their way through it until the ice melts and they are passed into the ocean. They are defined as ice and snow volume tracers (Eq. 15 and 16 in CICE.v5 documentation), with the snow and ice each having two tracers for each aerosol species, one in the surface scattering layer (delta-Eddington SSL) and one in the snow or ice interior below the SSL.

Rather than updating aerosols for each change to ice/snow thickness due to evaporation, melting, snow-ice formation, etc., during the thermodynamics calculation, these changes are deduced from the diagnostic variables (melts, meltb, snoice, etc) in **icepack_aerosol.F90**. Three processes change the volume of ice or snow but do not change the total amount of aerosol, thus causing the aerosol concentration (the value of the tracer itself) to increase: evaporation, snow deposition and basal ice growth. Basal and lateral melting remove all aerosols in the melted portion. Surface ice and snow melt leave a significant fraction of the aerosols behind, but they do "scavenge" a fraction of them given by the parameter kscav = [0.03, 0.2, 0.02, 0.02, 0.01, 0.01] (only the first 3 are used in CESM, for their 3 aerosol species). Scavenging also applies to snow-ice formation. When sea ice ridges, a fraction of the snow on the ridging ice is thrown into the ocean, and any aerosols in that fraction are also lost to the ocean.

As upper SSL or interior layers disappear from the snow or ice, aerosols are transferred to the next lower layer, or into the ocean when no ice remains. The atmospheric flux faero_atm contains the rates of aerosol deposition for each species, while faero_ocn has the rate at which the aerosols are transferred to the ocean.

The aerosol tracer flag tr_aero must be set to true in **icepack_in**, and the number of aerosol species is set in **icepack.settings**; CESM uses 3. Global diagnostics are available when print_global is true, and history variables include the mass density for each layer (snow and ice SSL and interior), and atmospheric and oceanic fluxes, for each species.

Z-Aerosols

zbgc_colpkg offers an alternate scheme for aerosols in sea ice using the brine motion based transport scheme of the biogeochemical tracers. All vertically resolved biogeochemical tracers (z-tracers), including zaerosols, have the potential to be atmospherically deposited onto the snow or ice, scavenged during snow melt, and passed into the brine. The mobile fraction (discussed in *Mobile and Stationary Phases in code*) is then transported via brine drainage processes (Eq. (1.125) in section *Transport along the interface bio grid*) while a stationary fraction (discussed in *Mobile and Stationary Phases in code*) adheres to the ice crystals. Snow deposition and the process of scavenging aerosols during snow melt is consistent with the default aerosol scheme, though parameters have been generalized to accomadate potential atmospheric deposition for all z-tracers. For an example, see the scavenging parameter kscavz for z-tracers defined in **icepack_zbgc_shared**.

Within the snow, z-tracers are defined as concentrations in the snow surface layer (h_{ssl}) and the snow interior ($h_s - h_{ssl}$). The total snow content of z-tracers per ice area per grid cell area, C_{snow} is

$$C_{snow} = C_{ssl}h_{ssl} + C_{sint}(h_s - h_{ssl})$$

One major difference in how the two schemes model snow aerosol transport is that the fraction scavenged from snow melt in the z-tracer scheme is not immediately fluxed into the ocean, but rather, enters the ice as a source of low salinity but potentially tracer rich brine. The snow melt source is included as a surface flux condition in **icepack_algae.F90**.

All the z-aerosols are nonreactive with the exception of the dust aerosols. We assume that a small fraction of the dust flux into the ice has soluble iron (dustFe_sol in **icepack_in**) and so is passed to the dissolved iron tracer. The remaining dust passes through the ice without reactions.

To use z-aerosols, tr_zaero must be set to true in **icepack_in**, and the number of z-aerosol species is set in **icepack.settings**, TRZAERO. Note, the default tracers tr_aero must be false and NTRAERO in **icepack.settings** should be 0. In addition, z-tracers and the brine height tracer must also be active. These are set in **icepack_in** with tr_brine and z_tracer equal to true. In addition, to turn on the radiative coupling between the aerosols and the Delta-Eddington radiative scheme, shortwave must equal 'dEdd' and dEdd_algae must be true in **icepack_in**.

Brine height

The brine height, h_b, is the distance from the ice-ocean interface to the brine surface. When tr_brine is set true in **icepack_in** and TRBRI is set equal to 1 in **icepack.settings**, the brine surface can move relative to the ice surface. Physically, this occurs when the ice is permeable and there is a nonzero pressure head: the difference between the brine height and the equilibrium sea surface. Brine height motion is computed in **icepack_brine.F90** from thermodynamic variables and the ice microstructural state deduced from internal bulk salinities and temperature. This tracer is required for the transport of vertically resolved biogeochemical tracers and is closely coupled to the z-salinity prognostic salinity model.

Vertical transport processes are, generally, a result of the brine motion. Therefore the vertical transport equations for biogeochemical tracers will be defined only where brine is present. This region, from the ice-ocean interface to the brine height, defines the domain of the vertical bio-grid. The resolution of the bio-grid is specified in **icepack.settings** by setting the variable NBGCLYR. A detailed description of the bio-grid is given in section *Bio grid*. The ice microstructural state, determined in **icepack_brine.F90**, is computed from sea ice salinities and temperatures linearly interpolated to the bio-grid. When $h_b > h_i$, the upper surface brine is assumed to have the same temperature as the ice surface.

Brine height is transported horizontally as the fraction $f_{bri} = h_b/h_i$, a volume conserved tracer. Note that unlike the sea ice porosity, brine height fraction may be greater than 1 when $h_b > h_i$.

Changes to h_b occur from ice and snow melt, ice bottom boundary changes, and from pressure adjustments. The computation of h_b at $t + \Delta t$ is a two step process. First, h_b is updated from changes in ice and snow thickness, ie.

$$h'_b = h_b(t) + \Delta h_b|_{h_i,h_s}. \tag{1.111}$$

Second, pressure driven adjustments arising from meltwater flushing and snow loading are applied to h'_b. Brine flow due to pressure forces are governed by Darcy's equation

$$w = -\frac{\Pi^* \bar{\rho} g}{\mu}\frac{h_p}{h_i}. \tag{1.112}$$

The vertical component of the net permeability tensor Π^* is computed as

$$\Pi^* = \left(\frac{1}{h}\sum_{i=1}^{N}\frac{\Delta z_i}{\Pi_i}\right)^{-1} \tag{1.113}$$

where the sea ice is composed of N vertical layers with ith layer thickness Δz_i and permeability Π_i. The average sea ice density is $\bar{\rho}$ specified in **icepack_zbgc_shared.F90**. The hydraulic head is $h_p = h_b - h_{sl}$ where h_{sl} is the sea level given by

$$h_{sl} = \frac{\bar{\rho}}{\rho_w}h_i + \frac{\rho_s}{\rho_w}h_s. \tag{1.114}$$

Assuming constant h_i and h_s during Darcy flow, the rate of change of h_b is

$$\frac{\partial h_b}{\partial t} = -w h_p \tag{1.115}$$

where $w_o = \Pi^* \bar{\rho} g / (h_i \mu \phi_{top})$ and ϕ_{top} is the upper surface porosity. When the Darcy flow is downward into the ice ($w_o < 0$), then ϕ_{top} equals the sea ice porosity in the uppermost layer. However, when the flow is upwards into the snow, then ϕ_{top} equals the snow porosity phi_snow specified in **icepack_in**. If a negative number is specified for phi_snow, then the default value is used: phi_snow $= 1 - \rho_s/\rho_w$.

Since h_{sl} remains relatively unchanged during Darcy flow, (1.115) has the approximate solution

$$h_b(t + \Delta t) \approx h_{sl}(t + \Delta t) + \left[h_b' - h_{sl}(t + \Delta t)\right] \exp\left\{-w \Delta t\right\}. \tag{1.116}$$

The contribution $\Delta h_b|_{h_i,h_s}$ arises from snow and ice melt and bottom ice changes. Since the ice and brine bottom boundaries coincide, changes in the ice bottom from growth or melt, $(\Delta h_i)_{bot}$, equal the bottom brine boundary changes. The surface contribution from ice and snow melt, however, is opposite in sign. The ice contribution is as follows. If $h_i > h_b$ and the ice surface is melting, ie. $(\Delta h_i)_{top} < 0$), then meltwater increases the brine height:

$$(\Delta h_b)_{top} = \frac{\rho_i}{\rho_o} \cdot \begin{cases} -(\Delta h_i)_{top} & \text{if } |(\Delta h_i)_{top}| < h_i - h_b \\ h_i - h_b & \text{otherwise.} \end{cases} \quad : label : delta - hb$$

For snow melt ($\Delta h_s < 0$), it is assumed that all snow meltwater contributes a source of surface brine. The total change from snow melt and ice thickness changes is

$$\Delta h_b|_{h_i,h_s} = (\Delta h_b)_{top} - (\Delta h_i)_{bot} - \frac{\rho_s}{\rho_o} \Delta h_s. \tag{1.117}$$

The above brine height calculation is used only when h_i and h_b exceed a minimum thickness, thinS, specified in **icepack_zbgc_shared**. Otherwise

$$h_b(t + \Delta t) = h_b(t) + \Delta h_i \tag{1.118}$$

provided that $|h_{sl} - h_b| \leq 0.001$. This formulation ensures small Darcy velocities when h_b first exceeds thinS.

Both the volume fraction f_{bri} and the area-weighted brine height h_b are available for output.

$$\frac{\sum f_{bri} v_i}{\sum v_i}, \tag{1.119}$$

while `hbri` is comparable to hi (h_i)

$$\frac{\sum f_{bri} h_i a_i}{\sum a_i}, \tag{1.120}$$

where the sums are taken over thickness categories.

Sea Ice Biogeochemistry

There are two options for modeling biogeochemistry in sea ice: 1) a skeletal layer or bottom layer model (skl-model) that assumes biology and biological molecules are restricted to a single layer at the base of the sea ice; and 2) a vertically resolved model (zbgc) that allows for biogeochemical processes throughout the ice column. The two models may be run with the same suite of biogeochemical tracers and use the same module **algal_dyn** in **icepack_algae.F90** to determine the biochemical reaction terms for the tracers at each vertical grid level. In the case of the skl-model this is a single layer, while for zbgc there are NBGCLYR+1 vertical layers. The primary difference between the two schemes is in the vertical transport assumptions for each biogeochemical tracer. This includes the parameterizations of fluxes between ocean and ice.

In order to run with the skl-model, the code must be built with the following options in **icepack.settings**:

```
setenv TRBGCS 1   # set to 1 for skeletal layer tracers
setenv TRBGCZ 0   # set to 1 for zbgc tracers
```

For zbgc with 8 vertical layers:

```
setenv TRBRI  1   # set to 1 for brine height tracer
setenv TRBGCS 0   # set to 1 for skeletal layer tracers
setenv TRBGCZ 1   # set to 1 for zbgc tracers
setenv NBGCLYR 7  # number of zbgc layers
```

There are also environmental variables in **icepack.settings** that, in part, specify the complexity of the ecosystem and are used for both zbgc and the skl-model. These are 1) TRALG, the number of algal species; 2) TRDOC, the number of dissolved organic carbon groups, 3) TRDIC, the number of dissolved inorganic carbon groups (this is currently not yet implemented and should be set to 0); 4) TRDON, the number of dissolved organic nitrogen groups, 5) TRFEP , the number of particulate iron groups; and 6) TRFED, the number of dissolved iron groups. The current version of **algal_dyn** biochemistry has parameters for up to 3 algal species (diatoms, small phytoplankton and *Phaeocystis* sp, respectively), 2 DOC tracers (polysaccharids and lipids, respectively), 0 DIC tracers, 1 DON tracer (proteins/amino acids), 1 particulate iron tracer and 1 dissolved iron tracer. Note, for tracers with multiple species/groups, the order is important. For example, specifying TRALG = 1 will compute reaction terms using parameters specific to ice diatoms. However, many of these parameters can be modified in **icepack_in**.

The complexity of the algal ecosystem must be specified in both **icepack.settings** during the build and in the namelist, **icepack_in**. The procedure is equivalent for both the skl-model and zbgc. The namelist specification is described in detail in section *Vertical "Z" BGC*

Biogeochemical upper ocean concentrations are initialized in the subroutine **icepack_init_ocean_conc** in **icepack_zbgc.F90** unless coupled to the ocean biogeochemistry. Silicate and nitrate may be read from a file. This option is specified in the namelist by setting the variables sil_data_type and nit_data_type to 'ISPOL' or 'NICE'. nit_data_type also has an option 'sss' which equates the upper ocean nitrate concentration with sea surface salinity. fe_data_type currently only has the 'default' option. The location of forcing files is specified in bgc_data_dir and the filename is hardcoded in **icedrv_forcing** (NJ - needs to be updated).

Skeletal Layer BGC

In the skeletal layer model, biogeochemical processing is modelled as a single layer of reactive tracers attached to the sea ice bottom. Optional settings are available via the *zbgc_nml* namelist in **icepack_in**. In particular, skl_bgc must be true and z_tracers and solve_zbgc must both be false.

History fields are controlled in the *icefields_bgc_nml* namelist and will be discussed in section *Biogeochemistry History Fields*. As with other CICE history fields, the suffix _ai indicates that the field is multiplied by ice area and is therefore a grid cell average.

Skeletal tracers T_b are ice area conserved and follow the horizontal transport Equation (1.22). For each horizontal grid point, local biogeochemical tracer equations are solved in **icepack_algae.F90**. There are two types of ice-ocean tracer flux formulations: 1) 'Jin2006' modeled after the growth rate dependent piston velocity and 2) 'constant' modeled after a constant piston velocity. The formulation is specified in **icepack_in** by setting bgc_flux_type equal to 'Jin2006' or 'constant'.

In addition to horizontal advection and transport among thickness categories, biogeochemical tracers (T_b where $b = 1, \ldots, N_b$) satisfy a set of local coupled equations of the form

$$\frac{dT_b}{dt} = w_b \frac{\Delta T_b}{\Delta z} + R_b(T_j : j = 1, \ldots, N_b) \tag{1.121}$$

where R_b represents the nonlinear biochemical reaction terms (described in section *Reaction Terms*) and Δz is a length scale representing the molecular sublayer of the ice-ocean interface. Its value is absorbed in the piston velocity parameters. The piston velocity w_b depends on the particular tracer and the flux formulation.

For 'Jin2006', the piston velocity is a function of ice growth and melt rates. All tracers (algae included) flux with the same piston velocity during ice growth, $dh/dt > 0$:

$$w_b = \quad -p_g \left| m_1 + m_2 \frac{dh}{dt} - m_3 \left(\frac{dh}{dt} \right)^2 \right| \tag{1.122}$$

with parameters m_1, m_2, m_3 and p_g defined in **skl_biogeochemistry** in **icepack_algae.F90**. For ice melt, $dh/dt < 0$, all tracers with the exception of ice algae flux with

$$w_b = \quad p_m \left| m_2 \frac{dh}{dt} - m_3 \left(\frac{dh}{dt} \right)^2 \right| \tag{1.123}$$

with p_m defined in **skl_biogeochemistry**. The 'Jin2006' formulation also requires that for both expressions, $|w_b| \leq 0.9 h_{sk}/\Delta t$. The concentration difference at the ice-ocean boundary for each tracer, ΔT_b, depends on the sign of w_b. For growing ice, $w_b < 0$, $\Delta T_b = T_b/h_{sk} - T_{io}$, where T_{io} is the ocean concentration of tracer i. For melting ice, $w_b > 0$, $\Delta T_b = T_b/h_{sk}$.

In 'Jin2006', the algal tracer (N_a) responds to ice melt in the same manner as the other tracers (1.123). However, this is not the case for ice growth. Unlike dissolved nutrients, algae are able to cling to the ice matrix and resist expulsion during desalination. For this reason, algal tracers do not flux between ice and ocean during ice growth unless the ice algal brine concentration is less than the ocean algal concentration (N_o). Then the ocean seeds the sea ice concentration according to

$$w_b \frac{\Delta N_a}{\Delta z} = \frac{N_o h_{sk}/\phi_{sk} - N_a}{\Delta t} \tag{1.124}$$

The 'constant' formulation uses a fixed piston velocity (PVc) for positive ice growth rates for all tracers except N_a. As in 'Jin2006', congelation ice growth seeds the sea ice algal population according to (1.124) when $N_a < N_o h_{sk}/\phi_{sk}$. For bottom ice melt, all tracers follow the prescription

$$w_b \frac{\Delta T_b}{\Delta z} = \begin{cases} T_b |dh_i/dt|/h_{sk} & \text{if } |dh_i/dt| \Delta t/h_{sk} < 1 \\ T_b/\Delta t & \text{otherwise.} \end{cases} \quad : label : constant_melt$$

A detailed description of the biogeochemistry reaction terms is given in section *Reaction Terms*.

Vertical "Z" BGC

In order to solve for the vertically resolved biogeochemistry, several flags in **icepack_in** must be true: a) tr_brine, b) z_tracers, and c) solve_zbgc.

- a) tr_brine true, turns on the dynamic brine height tracer, h_b, which defines the vertical domain of the biogeochemical tracers. z-Tracer horizontal transport is conserved on ice volume$\times$brine height fraction.
- b) z_tracers true, indicates use of vertically resolved biogeochemical and z-aerosol tracers. This flag alone turns on the vertical transport scheme but not the biochemistry.
- c) solve_zbgc true, turns on the biochemistry for the vertically resolved tracers and automatically turns on the algal nitrogen tracer flag tr_bgc_N. If false, tr_bgc_N is set false and any other biogeochemical tracers in use are transported as passive tracers. This is appropriate for the black carbon and dust aerosols specified by tr_zaero true.

In addition, a halodynamics scheme must also be used. The default thermo-halodynamics is mushy layer ktherm set to 2. An alternative uses the Bitz and Lipscomb thermodynamics ktherm set to 1 and solve_zsal true (referred to as "zsalinity").

With the above flags true, the default biochemistry is a simple algal-nitrate system: tr_bgc_N and tr_bgc_Nit equal true. Options exist in icepack_in to use a more complicated ecosystem which includes up to three algal classes, two

DOC groups, one DON pool, limitation by nitrate, silicate and dissolved iron, sulfur chemistry plus refractory humic material.

The **icepack_in** namelist options are described below.

```
&zbgc_nml
    tr_brine        = .true.     ! turns on the brine height tracer
                                 ! (needs TRBRI 1 in comp_ice)
  , restart_hbrine  = .false.    ! restart the brine height tracer
                                 ! (will be automatically switched on
                                 ! if restart = .true.)
  , tr_zaero        = .false.    ! turns on black carbon and
                                 ! dust aerosols
  , modal_aero      = .false.    ! turns on a modal aerosol option
                                 ! (not well tested)
  , skl_bgc         = .false.    ! turns on a single bottom layer
                                 ! biogeochemistry.  z_tracers and
                                 ! solve_zbgc must be false
                                 ! (needs TRBGCS 1 in comp_ice)
  , z_tracers       = .true.     ! turns on vertically resolved transport
                                 ! (needs TRBGCZ 1 in comp_ice)
  , dEdd_algae      = .false.    ! Include radiative impact of algae
                                 ! and aerosols in the delta-Eddington
                                 ! shortwave scheme.  Requires
                                 ! shortwave = 'dEdd'
                                 ! (Should not be used when solve_zbgc
                                 !  of skl_bgc are true*)
  , solve_zbgc      = .true.     ! turns on the biochemistry using z_tracers
                                 ! (specify algal numbers in comp_ice TRALG)
  , bgc_flux_type   = 'Jin2006'  ! ice-ocean flux type for bottom
                                 ! layer tracers only (skl_bgc = .true.)
  , restore_bgc     = .false.    ! restores upper ocean concentration
                                 ! fields to data values (nitrate and
                                 ! silicate)
  , restart_bgc     = .false.    ! restarts biogeochemical tracers
                                 ! (will be automatically switched on
                                 ! if restart = .true.)
  , scale_bgc       = .false.    ! Initializes biogeochemical profiles
                                 ! to scale with prognosed salinity profile
  , solve_zsal      = .false.    ! prognostic salinity tracer used with
                                 ! ktherm = 1 (zsalinity)
                                 ! (needs TRZS 1 in comp_ice)
  , restart_zsal    = .false.    ! restarts zsalinity
  , bgc_data_dir    = '/nitrate_and_silicate/forcing_directory/'
  , sil_data_type   = 'default'  ! fixed, spatially homogenous
                                 ! value. 'clim' data file
                                 ! (see ice_forcing_bgc.F90)
  , nit_data_type   = 'default'  ! fixed, spatially homogenous
                                 ! value. 'clim' data file
                                 ! (see ice_forcing_bgc.F90)
  , fe_data_type    = 'default'  ! fixed, spatially homogenous
  , tr_bgc_Nit      = .true.     ! nitrate tracer
  , tr_bgc_C        = .true.     ! Dissolved organic carbon tracers
                                 ! (numbers specified in comp_ice as
                                 ! TRDOC) and dissolved inorganic
                                 ! carbon tracers (not yet implemented,
                                 ! TRDIC 0 in comp_ice)
  , tr_bgc_chl      = .false.    ! dummy variable for now.  Chl is
                                 ! simply a fixed ratio of algal Nitrogen
```

```
, tr_bgc_Am       = .true.     ! Ammonium
, tr_bgc_Sil      = .true.     ! Silicate
, tr_bgc_DMS      = .true.     ! Three tracers: DMS dimethyl sulfide, DMSPp
                               ! (assumed to be a fixed ratio of
                               ! sulfur to algal Nitrogen) and
                               ! DMSPd
, tr_bgc_PON      = .false.    ! passive purely mobile ice tracer with
                               ! ocean concentration equivalent to nitrate
, tr_bgc_hum      = .true.     ! refractory DOC or DON (units depend
                               ! on the ocean source)
, tr_bgc_DON      = .true.     ! dissolved organic nitrogen (proteins)
, tr_bgc_Fe       = .true.     ! Dissolved iron (number in comp_ice TRFED)
                               ! particulate iron (number in comp_ice TRFEP)
, grid_o          = 0.006      ! ice-ocean surface layer thickness
                               ! (bgc transport scheme)
, grid_o_t        = 0.006      ! ice-atm surface layer thickeness
                               ! (bgc transport scheme)
, l_sk            = 0.024      ! length scale in gravity drainage
                               !  parameterization
                               ! (bgc transport scheme)
, grid_oS         = 0.0        ! ice-ocean surface layer thickness
                               ! (zsalinity transport scheme)
, l_skS           = 0.028      ! ice-atm surface layer thickeness
                               ! (zsalinity transport scheme)
, phi_snow        = -0.3       ! snow porosity at the ice-snow interface
                               ! if < 0 then phi_snow is computed
                               ! from snow density
, initbio_frac    = 0.8        ! For each bgc tracer, specifies the
                               ! fraction of the ocean
                               ! concentration that is retained in
                               ! the ice during initial new ice formation.
, frazil_scav     = 0.8        ! For each bgc tracer, specifies the
                               ! fraction or multiple of the ocean
                               ! concentration that is retained in
                               ! the ice from frazil formation.
                               !---------------------------------------
                               !  Notation used below:
                               !  _diatoms  == diatoms
                               !  _sp       == small phytoplankton
                               !  _phaeo    == phaeocystis
                               !  _s        == saccharids
                               !       (unless otherwise indicated)
                               !  _l        == lipdids
                               !       (unless otherwise indicated)
                               !---------------------------------------
, ratio_Si2N_diatoms = 1.8_dbl_kind    ! algal Si to N (mol/mol)
, ratio_Si2N_sp      = c0
, ratio_Si2N_phaeo   = c0
, ratio_S2N_diatoms  = 0.03_dbl_kind   ! algal S  to N (mol/mol)
, ratio_S2N_sp       = 0.03_dbl_kind
, ratio_S2N_phaeo    = 0.03_dbl_kind
, ratio_Fe2C_diatoms = 0.0033_dbl_kind ! algal Fe to C  (umol/mol)
, ratio_Fe2C_sp      = 0.0033_dbl_kind
, ratio_Fe2C_phaeo   = p1
, ratio_Fe2N_diatoms = 0.023_dbl_kind  ! algal Fe to N  (umol/mol)
, ratio_Fe2N_sp      = 0.023_dbl_kind
, ratio_Fe2N_phaeo   = 0.7_dbl_kind
, ratio_Fe2DON       = 0.023_dbl_kind  ! Fe to N of DON (nmol/umol)
, ratio_Fe2DOC_s     = p1               ! Fe to C of DOC (nmol/umol)
, ratio_Fe2DOC_l     = 0.033_dbl_kind   ! Fe to C of DOC (nmol/umol)
, fr_resp            = 0.05_dbl_kind    ! frac of algal growth lost
                                        ! due to respiration
, tau_min            = 5200.0_dbl_kind ! rapid mobile to stationary
                                        ! exchanges (s)
, tau_max            = 1.73e5_dbl_kind ! long time mobile to
                                        ! stationary exchanges (s)
, algal_vel          = 1.11e-8_dbl_kind! 0.5 cm/d(m/s)
, R_dFe2dust         = 0.035_dbl_kind   ! g/g (3.5% content)
, dustFe_sol         = 0.005_dbl_kind   ! solubility fraction
, chlabs_diatoms     = 0.03_dbl_kind    ! chl absorption (1/m/(mg/m^3))
, chlabs_sp          = 0.01_dbl_kind
, chlabs_phaeo       = 0.05_dbl_kind
, alpha2max_low_diatoms = 0.8_dbl_kind ! light limitation (1/(W/m^2))
, alpha2max_low_sp      = 0.67_dbl_kind
, alpha2max_low_phaeo   = 0.67_dbl_kind
, beta2max_diatoms   = 0.018_dbl_kind   ! light inhibition (1/(W/m^2))
, beta2max_sp        = 0.0025_dbl_kind
, beta2max_phaeo     = 0.01_dbl_kind
, mu_max_diatoms     = 1.2_dbl_kind     ! maximum growth rate (1/day)
, mu_max_sp          = 0.851_dbl_kind
, mu_max_phaeo       = 0.851_dbl_kind
, grow_Tdep_diatoms  = 0.06_dbl_kind ! Temperature dependence of
                                      ! growth (1/C)
, grow_Tdep_sp       = 0.06_dbl_kind
, grow_Tdep_phaeo    = 0.06_dbl_kind
, fr_graze_diatoms   = 0.01_dbl_kind ! Fraction grazed
, fr_graze_sp        = p1
, fr_graze_phaeo     = p1
, mort_pre_diatoms   = 0.007_dbl_kind! Mortality (1/day)
, mort_pre_sp        = 0.007_dbl_kind
, mort_pre_phaeo     = 0.007_dbl_kind
, mort_Tdep_diatoms  = 0.03_dbl_kind ! T dependence of mortality (1/C)
, mort_Tdep_sp       = 0.03_dbl_kind
, mort_Tdep_phaeo    = 0.03_dbl_kind
, k_exude_diatoms    = c0            ! algal exudation (1/d)
, k_exude_sp         = c0
, k_exude_phaeo      = c0
, K_Nit_diatoms      = c1            ! nitrate half saturation
                                      ! (mmol/m^3)
, K_Nit_sp           = c1
, K_Nit_phaeo        = c1
, K_Am_diatoms       = 0.3_dbl_kind  ! ammonium half saturation
                                      ! (mmol/m^3)
, K_Am_sp            = 0.3_dbl_kind
, K_Am_phaeo         = 0.3_dbl_kind
, K_Sil_diatoms      = 4.0_dbl_kind  ! silicate half saturation
                                      ! (mmol/m^3)
, K_Sil_sp           = c0
, K_Sil_phaeo        = c0
, K_Fe_diatoms       = c1            ! iron half saturation (nM)
, K_Fe_sp            = 0.2_dbl_kind
, K_Fe_phaeo         = p1
, f_don_protein      = 0.6_dbl_kind  ! fraction of spilled grazing
                                      ! to proteins
, kn_bac_protein     = 0.03_dbl_kind ! Bacterial degredation of DON (1/d)
, f_don_Am_protein   = 0.25_dbl_kind ! fraction of remineralized
```

```
                                     ! DON to ammonium
  , f_doc_s            = 0.4_dbl_kind  ! fraction of mortality to DOC
  , f_doc_l            = 0.4_dbl_kind  !
  , f_exude_s          = c1            ! fraction of exudation to DOC
  , f_exude_l          = c1            !
  , k_bac_s            = 0.03_dbl_kind ! Bacterial degredation of DOC (1/d)
  , k_bac_l            = 0.03_dbl_kind !
  , T_max              = c0            ! maximum temperature (C)
  , fsal               = c1            ! Salinity limitation (ppt)
  , op_dep_min         = p1            ! Light attenuates for optical
                                       ! depths exceeding min
  , fr_graze_s         = p5            ! fraction of grazing spilled
                                       ! or slopped
  , fr_graze_e         = p5            ! fraction of assimilation excreted
  , fr_mort2min        = p5            ! fractionation of mortality to Am
  , fr_dFe             = 0.3_dbl_kind  ! fraction of remineralized nitrogen
  ,                                    ! (in units of algal iron)
  , k_nitrif           = c0            ! nitrification rate (1/day)
  , t_iron_conv        = 3065.0_dbl_kind ! desorption loss pFe to dFe (day)
  , max_loss           = 0.9_dbl_kind  ! restrict uptake to % of remaining value
  , max_dfe_doc1       = 0.2_dbl_kind  ! max ratio of dFe to
                                       ! saccharides in the ice
                                       !(nM Fe/muM C)
  , fr_resp_s          = 0.75_dbl_kind ! DMSPd fraction of respiration
                                       ! loss as DMSPd
  , y_sk_DMS           = p5            ! fraction conversion given high yield
  , t_sk_conv          = 3.0_dbl_kind  ! Stefels conversion time (d)
  , t_sk_ox            = 10.0_dbl_kind ! DMS oxidation time (d)
  , algaltype_diatoms  = c0            ! ------------------
  , algaltype_sp       = p5            !
  , algaltype_phaeo    = p5            !
  , nitratetype        = -c1           ! mobility type between
  , ammoniumtype       = c1            ! stationary <-->  mobile
  , silicatetype       = -c1           !
  , dmspptype          = p5            !
  , dmspdtype          = -c1           !
  , humtype            = c1            !
  , doctype_s          = p5            !
  , doctype_l          = p5            !
  , dontype_protein    = p5            !
  , fedtype_1          = p5            !
  , feptype_1          = p5            !
  , zaerotype_bc1      = c1            !
  , zaerotype_bc2      = c1            !
  , zaerotype_dust1    = c1            !
  , zaerotype_dust2    = c1            !
  , zaerotype_dust3    = c1            !
  , zaerotype_dust4    = c1            !--------------------
  , ratio_C2N_diatoms  = 7.0_dbl_kind  ! algal C to N ratio (mol/mol)
  , ratio_C2N_sp       = 7.0_dbl_kind
  , ratio_C2N_phaeo    = 7.0_dbl_kind
  , ratio_chl2N_diatoms= 2.1_dbl_kind  ! algal chlorophyll to N ratio (mg/mmol)
  , ratio_chl2N_sp     = 1.1_dbl_kind
  , ratio_chl2N_phaeo  = 0.84_dbl_kind
  , F_abs_chl_diatoms  = 2.0_dbl_kind  ! scales absorbed radiation for dEdd
  , F_abs_chl_sp       = 4.0_dbl_kind
  , F_abs_chl_phaeo    = 5.0
  , ratio_C2N_proteins = 7.0_dbl_kind  ! ratio of C to N in proteins (mol/mol)
```

```
/
```

Vertically resolved z-tracers are brine volume conserved and thus depend on both the ice volume and the brine height fraction tracer ($v_{in} f_b$). These tracers follow the conservation equations for multiply dependent tracers (see, for example Equation (1.68) where a_{pnd} is a tracer on $a_{lvl} a_i$)

The following sections describe the vertical biological grid, the vertical transport equation for mobile tracers, the partitioning of tracers into mobile and stationary fractions and the biochemical reaction equations.

Bio grid

The bio grid is a vertical grid used for solving the brine height variable h_b and descretizing the vertical transport equations of biogeochemical tracers. The bio grid is a non-dimensional vertical grid which takes the value zero at h_b and one at the ice-ocean interface. The number of grid levels is specified during compilation in **icepack.settings** by setting the variable NBGCLYR equal to an integer (n_b) .

Ice tracers and microstructural properties defined on the bio grid are referenced in two ways: as 1) $n_b + 2$ bgrid points and 2) $n_b + 1$ igrid points. For both bgrid and igrid, the first and last points reference h_b and the ice-ocean interface and so take the values 0 and 1, respectively. For bgrid, the interior points $[2, n_b + 1]$ are spaced at $1/n_b$ intervals beginning with bgrid(2) = $1/(2n_b)$. The igrid interior points $[2, n_b]$ are also equidistant with the same spacing, but physically coincide with points midway between those of the bgrid.

Transport along the interface bio grid

Purely mobile tracers are tracers which move with the brine and thus, in the absence of biochemical reactions, evolve like salinity. For vertical tracer transport of purely mobile tracers, the flux conserved quantity is the bulk tracer concentration multiplied by the ice thickness, i.e. $C = h\phi[c]$, where h is the ice thickness, ϕ is the porosity, and $[c]$ is the tracer concentration in the brine. ϕ, $[c]$ and C are defined on the interface bio grid (igrid):

$$\text{igrid}(k) = \Delta(k-1) \quad \text{for } k = 1 : n_b + 1$$

and $\Delta = 1/n_b$

The biogeochemical module solves the following equation:

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left\{\left(\frac{v}{h} + \frac{w_f}{h\phi} - \frac{\tilde{D}}{h^2\phi^2}\frac{\partial \phi}{\partial x}\right)C + \frac{\tilde{D}}{h^2\phi}\frac{\partial C}{\partial x}\right\} + h\phi R([c]) \tag{1.125}$$

where $D_{in} = \tilde{D}/h^2 = (D + \phi D_m)/h^2$ and $R([c])$ is the nonlinear biogeochemical interaction term (see *[22]*).

The solution to (1.125) is flux-corrected and positive definite. This is accomplished using a finite element Gelerkin discretization. Details are in *Flux-Corrected, Positive Definite Transport Scheme* .

Splitting tracers: zbgc_type

In addition to purely mobile tracers, some tracers may also adsorb or otherwise adhere to the ice crystals. These tracers exist in both the mobile and stationary phases. In this case, their total brine concentration is a sum $c_m + c_s$ where c_m is the moble fraction transported by equation (1.125) and c_s is fixed vertically in the ice matrix. The algae are an exception, however. We assume that algae in the stationary phase resist brine motion, but rather than being fixed vertically, these tracers maintain their relative position in the ice. Algae that adhere to the ice interior (bottom, surface), remain in the ice interior (bottom, surface) until release to the mobile phase.

In order to model the transfer between these fractions, we assume that tracers adhere (are retained) to the crystals with a time-constant of τ_{ret}, and release with a time constant τ_{rel}, i.e.

$$\begin{aligned}\frac{\partial c_m}{\partial t} &= -\frac{c_m}{\tau_{ret}} + \frac{c_s}{\tau_{rel}} \\ \frac{\partial c_s}{\partial t} &= \frac{c_m}{\tau_{ret}} - \frac{c_s}{\tau_{rel}}\end{aligned}$$

We use the exponential form of these equations:

$$\begin{aligned}c_m^{t+dt} &= c_m^t \exp\left(\left\{-\frac{dt}{\tau_{ret}}\right)\right\} + c_s^t\left(1 - \exp\left[-\left\{\frac{dt}{\tau_{rel}}\right\}\right]\right) \\ c_s^{t+dt} &= c_s^t \exp\left(\left\{-\frac{dt}{\tau_{rel}}\right)\right\} + c_m^t\left(1 - \exp\left[-\left\{\frac{dt}{\tau_{ret}}\right\}\right]\right)\end{aligned}$$

The time constants are functions of the ice growth and melt rates (dh/dt). All tracers except algal nitrogen diatoms follow the simple case: when $dh/dt \geq 0$, then $\tau_{rel} \to \infty$ and τ_{ret} is finite. For $dh/dt < 0$, then $\tau_{ret} \to \infty$ and τ_{rel} is finite. In other words, ice growth promotes transitions to the stationary phase and ice melt enables transitions to the mobile phase.

The exception is the diatom pool. We assume that diatoms, the first algal nitrogen group, can actively maintain their relative position within the ice, i.e. bottom (interior, upper) algae remain in the bottom (interior, upper) ice, unless melt rates exceed a threshold. The namelist parameter algal_vel sets this threshold.

Mobile and Stationary Phases in code

The variable bgc_tracer_type determines the mobile to stationary transition timescales for each z-tracer. It is multi-dimensional with a value for each z-tracer. For bgc_tracer_type(k) equal to -1, the kth tracer remains solely in the mobile phase. For bgc_tracer_type(k) equal to 1, the tracer has maximal rates in the retention phase and minimal in the release. For bgc_tracer_type(k) equal to 0, the tracer has maximal rates in the release phase and minimal in the retention. Finally for bgc_tracer_type(k) equal to 0.5, minimum timescales are used for both transitions. *Table 3* summarizes the transition types. The tracer types are: algaltype_diatoms, algaltype_sp (small plankton), algaltype_phaeo (*phaeocystis*), nitratetype, ammoniumtype, silicatetype, dmspptype, dmspdtype, humtype, doctype_s (saccharids), doctype_l (lipids), dontype_protein, fedtype_1, feptype_1, zaerotype_bc1 (black carbon class 1), zaerotype_bc2 (black carbon class 2), and four dust classes, zaerotype_dustj, where J takes values 1 to 4. These may be modified to increase or decrease retention. Another option is to alter the minimum tau_min and maximum tau_max timescales which would impact all the z-tracers.

Table 3: *Types of Mobile and Stationary Transitions*

Table 1.3: Table 3

bgc_tracer_type	τ_{ret}	τ_{rel}	Description
-1.0	∞	0	entirely in the mobile phase
0.0	min	max	retention dominated
1.0	max	min	release dominated
0.5	min	min	equal but rapid exchange
2.0	max	max	equal but slow exchange

The fraction of a given tracer in the mobile phase is independent of ice depth and stored in the tracer variable zbgc_frac. The horizontal transport of this tracer is conserved on brine volume and so is dependent on two tracers: brine height fraction (f_b) and ice volume (v_{in}). The conservation equations are given by Eq. 18 in the CICE.v5 documentation with $a_{pnd}a_i$ replaced by $f_b v_{in}$.

The tracer, zbgc_frac, is initialized to 1 during new ice formation, because all z-tracers are initially in the purely mobile phase. Similarly, as the ice melts, z-tracers return to the mobile phase. Very large release timescales will prevent this transition and could result in an unphysically large accumulation during the melt season.

Reaction Terms

The biogeochemical reaction terms for each biogeochemical tracer (see table *Table 4* for tracer definitions) are defined in **icepack_algae.F90** in the subroutine algal_dyn. The same set of equations is used for the bottom layer model (when sk_bgc is true) and the multi-layer biogeochemical model (when z_tracers and solve_zbgc are true).

Table 4: *Biogeochemical Tracers*

Table 1.4: Table 4

Text Variable	Variable in code	flag	Description	units
N (1)	Nin(1)	*tr_bgc_N*	diatom	$mmol\ N/m^3$
N (2)	Nin(2)	*tr_bgc_N*	small phytoplankton	$mmol\ N/m^3$
N (3)	Nin(3)	*tr_bgc_N*	*Phaeocystis sp*	$mmol\ N/m^3$
DOC (1)	DOCin(1)	*tr_bgc_DOC*	polysaccharids	$mmol\ C/m^3$
DOC (2)	DOCin(2)	*tr_bgc_DOC*	lipids	$mmol\ C/m^3$
DON	DONin(1)	*tr_bgc_DON*	proteins	$mmol\ C/m^3$
fed	Fedin(1)	*tr_bgc_Fe*	dissolved iron	$\mu\ Fe/m^3$
fep	Fepin(1)	*tr_bgc_Fe*	particulate iron	$\mu\ Fe/m^3$
NO_3	Nitin	*tr_bgc_Nit*	NO_3	$mmol\ N/m^3$
NH_4	Amin	*tr_bgc_Am*	NH_4	$mmol\ N/m^3$
SiO_3	Silin	*tr_bgc_Sil*	SiO_2	$mmol\ Si/m^3$
DMSPp	DMSPpin	*tr_bgc_DMS*	particulate DMSP	$mmol\ S/m^3$
DMSPd	DMSPdin	*tr_bgc_DMS*	dissolved DMSP	$mmol\ S/m^3$
DMS	DMSin	*tr_bgc_DMS*	DMS	$mmol\ S/m^3$
PON	PON [a]	*tr_bgc_PON*	passive mobile tracer	$mmol\ N/m^3$
hum	hum [ab]	*tr_bgc_hum*	passive sticky tracer	$mmol\ /m^3$
BC (1)	zaero(1) [a]	*tr_zaero*	black carbon species 1	$kg\ /m^3$
BC (2)	zaero(2) [a]	*tr_zaero*	black carbon species 2	$kg\ /m^3$
dust (1)	zaero(3) [a]	*tr_zaero*	dust species 1	$kg\ /m^3$
dust (2)	zaero(4) [a]	*tr_zaero*	dust species 2	$kg\ /m^3$
dust (3)	zaero(5) [a]	*tr_zaero*	dust species 3	$kg\ /m^3$
dust (4)	zaero(6) [a]	*tr_zaero*	dust species 4	$kg\ /m^3$

[a] not modified in `algal_dyn`

[b] may be in C or N units depending on the ocean concentration

The Reaction Equations

The biochemical reaction term for each algal species has the form:

$$\Delta \mathrm{N}/dt = R_{\mathrm{N}} = \mu(1 - f_{graze} - f_{res}) - M_{ort}$$

where μ is the algal growth rate, M_{ort} is a mortality loss, f_{graze} is the fraction of algal growth that is lost to predatory grazing, and f_{res} is the fraction of algal growth lost to respiration. Algal mortality is temperture dependent and limited by a maximum loss rate fraction (l_{max}):

$$M_{ort} = \min(l_{max}[\mathrm{N}], m_{pre}\exp\{m_T(T - T_{max})\}[\mathrm{N}])$$

Note, $[\cdots]$ denotes brine concentration.

Nitrate and ammonium reaction terms are given by

$$\begin{aligned}
\Delta \text{NO}_3/dt = & \quad R_{\text{NO}_3} = [\text{NH}_4]k_{nitr} - U^{tot}_{\text{NO}_3} \\
\Delta \text{NH}_4/dt = & \quad R_{\text{NH}_4} = -[\text{NH}_4]k_{nitr} - U^{tot}_{\text{NH}_4} + (f_{ng} f_{graze}(1 - f_{gs}) + f_{res})\mu^{tot} \\
+ & \quad f_{nm} M_{ort} \\
= & \quad -[\text{NH}_4]k_{nitr} - U^{tot}_{\text{NH}_4} + N_{remin}
\end{aligned}$$

where the uptake U^{tot} and algal growth μ^{tot} are accumulated totals for all algal species. k_{nitr} is the nitrification rate and f_{ng} and f_{nm} are the fractions of grazing and algal mortality that are remineralized to ammonium and f_{gs} is the fraction of grazing spilled or lost. Algal growth and nutrient uptake terms are described in more detail in *Algal Growth and Nutrient Uptake*.

Dissolved organic nitrogen satisfies the equation

$$\Delta \text{DON}/dt = \quad R_{\text{DON}} = f_{dg} f_{gs} f_{graze} \mu^{tot} - [\text{DON}]k_{nb}$$

With a loss from bacterial degration (rate k_{nb}) and a gain from spilled grazing that does not enter the NH_4 pool.

A term Z_{oo} closes the nitrogen cycle by summing all the excess nitrogen removed as zooplankton/bacteria in a timestep. This term is not a true tracer, i.e. not advected horizontally with the ice motion, but provides a diagnostic comparison of the amount of N removed biogeochemically from the ice N-NO_3-NH_4-DON cycle at each timestep.

$$\text{Z}_{oo} = \quad [(1 - f_{ng}(1 - f_{gs}) - f_{dg} f_{gs}] f_{graze} \mu^{tot} dt + (1 - f_{nm}) M_{ort} dt + [\text{DON}]k_{nb} dt$$

Dissolved organic carbon may be divided into polysaccharids and lipids. Parameters are two dimensional (indicated by superscript i) with index 1 corresponding to polysaccharids and index 2 appropriate for lipids. The DOC^i equation is:

$$\Delta \text{DOC}^i/dt = \quad R_{\text{DOC}} = f^i_{cg} f_{ng} \mu^{tot} + R^i_{c:n} M_{ort} - [\text{DOC}]k^i_{cb}$$

Silicate has no biochemical source terms within the ice and is lost only through algal uptake:

$$\Delta \text{SiO}_3/dt = \quad R_{\text{SiO}_3} = -U^{tot}_{\text{SiO}_3}$$

Dissolved iron has algal uptake and remineralization pathways. In addition, fed may be converted to or released from the particulate iron pool depending on the dissolve iron (fed) to polysaccharid (DOC(1)) concentration ratio. If this ratio exceeds a maximum value $r^{max}_{fed:doc}$ then the change in concentration for dissolved and particulate iron is

$$\begin{aligned}
\Delta_{fe}\text{fed}/dt = & \quad -[\text{fed}]/\tau_{fe} \\
\Delta_{fe}\text{fep}/dt = & \quad [\text{fed}]/\tau_{fe}
\end{aligned}$$

For values less than $r^{max}_{fed:doc}$

$$\begin{aligned}
\Delta_{fe}\text{fed}/dt = & \quad [\text{fep}]/\tau_{fe} \\
\Delta_{fe}\text{fep}/dt = & \quad -[\text{fep}]/\tau_{fe}
\end{aligned}$$

Very long timescales τ_{fe} will remove this source/sink term. The default value is currently set at 3065 days to turn off this dependency. 61-65 days is a more realistic option (Parekh et al., 2004).

The full equation for fed including uptake and remineralization is

$$\Delta \text{fed}/dt = \quad R_{\text{fed}} = -U^{tot}_{\text{fed}} + f_{fa} R_{fe:n} N_{remin} + \Delta_{fe}\text{fed}/dt$$

Particulate iron also includes a source term from algal mortality and grazing that is not immediately bioavailable. The full equation for fep is

$$\Delta \text{fep}/dt = \quad R_{\text{fep}} = R_{fe:n}[\text{Z}_{oo}/dt + (1 - f_{fa})]N_{remin} + \Delta_{fe}\text{fep}/dt$$

The sulfur cycle includes DMS and dissolved DMSP (DMSPd). Particulate DMSP is assumed to be proportional to the algal concentration, i.e. DMSPp $= R^i_{s:n}\mathrm{N}^i$ for algal species i. For DMSP and DMS,

$$\begin{aligned}\Delta\text{DMSPd}/dt = &\quad R_{\text{DMSPd}} = R_{s:n}[f_{sr}f_{res}\mu^{tot} + f_{nm}M_{ort}] - [\text{DMSPd}]/\tau_{dmsp}\\ \Delta\text{DMS}/dt = &\quad R_{\text{DMS}} = y_{dms}[\text{DMSPd}]/\tau_{dmsp} - [\text{DMS}]/\tau_{dms}\end{aligned}$$

See *BGC Tuning Parameters* and Table *Table 5* for a more complete list and description of biogeochemical parameters.

Algal Growth and Nutrient Uptake

Nutrient limitation terms are defined, in the simplest ecosystem, for NO_3. If the appropriate tracer flags are true, then limitation terms may also be found for NH_4, SiO_3, and fed

$$\begin{aligned}\text{NO}_{3lim} = &\quad \frac{[\text{NO}_3]}{[\text{NO}_3] + K_{\text{NO}_3}}\\ \text{NH}_{4lim} = &\quad \frac{[\text{NH}_4]}{[\text{NH}_4] + K_{\text{NH}_4}}\\ N_{lim} = &\quad \min(1, \text{NO}_{3lim} + \text{NH}_{4lim})\\ \text{SiO}_{3lim} = &\quad \frac{[\text{SiO}_3]}{[\text{SiO}_3] + K_{\text{SiO}_3}}\\ \text{fed}_{lim} = &\quad \frac{[\text{fed}]}{[\text{fed}] + K_{\text{fed}}}\end{aligned}$$

Light limitation L_{lim} is defined in the following way: $I_{sw}(z)$ (in W/m^2) is the shortwave radiation at the ice level and the optical depth is proportional to the chlorophyll concentration, $op_{dep} = chlabs[\text{Chl}a]$. If ($op_{dep} > op_{min}$) then

$$I_{avg} = I_{sw}(1 - \exp(-op_{dep}))/op_{dep}$$

otherwise $I_{avg} = I_{sw}$.

$$L_{lim} = (1 - \exp(-\alpha \cdot I_{avg}))\exp(-\beta \cdot I_{avg})$$

The maximal algal growth rate before limitation is

$$\begin{aligned}\mu_o = &\quad \mu_{max}\exp(\mu_T\Delta T)f_{sal}[\text{N}]\\ \mu' = &\quad min(L_{lim}, N_{lim}, \text{SiO}_{3lim}, \text{fed}_{lim})\mu_o\end{aligned}$$

where μ' is the initial estimate of algal growth rate for a given algal species and ΔT is the difference between the local tempurature and the maximum (in this case $\text{T}_{max} = 0^o\text{C}$).

The initial estimate of the uptake rate for silicate and iron is

$$\begin{aligned}\tilde{U}_{\text{SiO}_3} = &\quad R_{si:n}\mu'\\ \tilde{U}_{\text{fed}} = &\quad R_{fe:n}\mu'\end{aligned}$$

For nitrogen uptake, we assume that ammonium is preferentially acquired by algae. To determine the nitrogen uptake needed for each algal growth rate of μ, first determine the "potential" uptake rate of ammonium:

$$U'_{\text{NH}_4} = \text{NH}_{4lim}\mu_o$$

Then

$$\begin{aligned}\tilde{U}_{\text{NH}_4} = &\quad \min(\mu', U'_{\text{NH}_4})\\ \tilde{U}_{\text{NO}_3} = &\quad \mu' - \tilde{U}_{\text{NH}_4}\end{aligned}$$

We require that each rate not exceed a maximum loss rate l_{max}/dt.This is particularly important when multiple species are present. In this case, the accumulated uptake rate for each nutrient is found and the fraction (fU^i) of uptake due to algal species i is saved. Then the total uptake rate is compared with the maximum loss condition. For example, the net uptake of nitrate when there are three algal species is

$$\tilde{U}^{tot}_{\mathrm{NO_3}} = \sum_{i=1}^{3} \tilde{U}^{i}_{\mathrm{NO_3}} \quad .$$

Then the uptake fraction for species i and the adjusted total uptake is

$$\begin{aligned} fU^i_{\mathrm{NO_3}} &= \frac{\tilde{U}^i_{\mathrm{NO_3}}}{\tilde{U}^{tot}_{\mathrm{NO_3}}} \\ U^{tot}_{\mathrm{NO_3}} &= \min(\tilde{U}^{tot}_{\mathrm{NO_3}}, l_{max}[\mathrm{NO_3}]/dt) \end{aligned}$$

Now, for each algal species the nitrate uptake is

$$U^i_{\mathrm{NO_3}} = fU^i_{\mathrm{NO_3}} U^{tot}_{\mathrm{NO_3}}$$

Similar expressions are found for all potentially limiting nutrients. Then the true growth rate for each algal species i is

$$\mu^i = \min(U^i_{\mathrm{SiO_3}}/R_{si:n}, U^i_{\mathrm{NO_3}} + U^i_{\mathrm{NH_4}}, U^i_{\mathrm{fed}}/R_{fe:n})$$

Preferential ammonium uptake is assumed once again and the remaining nitrogen is taken from the nitrate pool.

BGC Tuning Parameters

Biogeochemical tuning parameters are specified as namelist options in **icepack_in**. Table *Table 5* provides a list of parameters used in the reaction equations, their representation in the code, a short description of each and the default values. Please keep in mind that there has only been minimal tuning of the model.

Table 5 :*Biogeochemical Reaction Parameters*

Table 1.5: Table 5

Text Variable	Variable in code	Description	Value	units
f_{graze}	fr_graze(1:3)	fraction of growth grazed	0, 0.1, 0.1	1
f_{res}	fr_resp	fraction of growth respired	0.05	1
l_{max}	max_loss	maximum tracer loss fraction	0.9	1
m_{pre}	mort_pre(1:3)	maximum mortality rate	0.007, 0.007, 0.007	day^{-1}
m_T	mort_Tdep(1:3)	mortality temperature decay	0.03, 0.03, 0.03	^{o}C^{-1}
T_{max}	T_max	maximum brine temperature	0	^{o}C
k_{nitr}	k_nitrif	nitrification rate	0	day^{-1}
f_{ng}	fr_graze_e	fraction of grazing excreted	0.5	1

Continued on next page

Table 1.5 – continued from previous page

Text Variable	Variable in code	Description	Value	units
f_{gs}	fr_graze_s	fraction of grazing spilled	0.5	1
f_{nm}	fr_mort2min	fraction of mortality to NH_4	0.5	1
f_{dg}	f_don	frac. spilled grazing to DON	0.6	1
k_{nb}	kn_bac [a]	bacterial degradation of DON	0.03	day^{-1}
f_{cg}	f_doc(1:3)	fraction of mortality to DOC	0.4, 0.4, 0.2	1
$R^c_{c:n}$	R_C2N(1:3)	algal carbon to nitrogen ratio	7.0, 7.0, 7.0	mol/mol
k_{cb}	k_bac1:3[a]	bacterial degradation of DOC	0.03, 0.03, 0.03	day^{-1}
τ_{fe}	t_iron_conv	conversion time pFe $\leftrightarrow$ dFe	3065.0	day
$r^{max}_{fed:doc}$	max_dfe_doc1	max ratio of dFe to saccharids	0.1852	nM Fe$/\mu$M C
f_{fa}	fr_dFe	fraction of remin. N to dFe	0.3	1
$R_{fe:n}$	R_Fe2N(1:3)	algal Fe to N ratio	0.023, 0.023, 0.7	mmol/mol
$R_{s:n}$	R_S2N(1:3)	algal S to N ratio	0.03, 0.03, 0.03	mol/mol
f_{sr}	fr_resp_s	resp. loss as DMSPd	0.75	1
τ_{dmsp}	t_sk_conv	Stefels rate	3.0	day
τ_{dms}	t_sk_ox	DMS oxidation rate	10.0	day
y_{dms}	y_sk_DMS	yield for DMS conversion	0.5	1
$K_{\mathrm{NO_3}}$	K_Nit(1:3)	NO_3 half saturation constant	1,1,1	mmol/m^3
$K_{\mathrm{NH_4}}$	K_Am(1:3)	NH_4 half saturation constant	0.3, 0.3, 0.3	mmol/m^{-3}
$K_{\mathrm{SiO_3}}$	K_Sil(1:3)	silicate half saturation constant	4.0, 0, 0	mmol/m^{-3}
K_{fed}	K_Fe(1:3)	iron half saturation constant	1.0, 0.2, 0.1	μmol/m^{-3}
op_{min}	op_dep_min	boundary for light attenuation	0.1	1
$chlabs$	chlabs(1:3)	light absorption length per chla conc.	0.03, 0.01, 0.05	1$/$m$/$(mg:math:/m^3)
α	alpha2max_low(1:3)	light limitation factor	0.25, 0.25, 0.25	m^2/W
β	beta2max(1:3)	light inhibition factor	0.018, 0.0025, 0.01	m^2/W
μ_{max}	mu_max(1:3)	maximum algal growth rate	1.44, 0.851, 0.851	day^{-1}
μ_T	grow_Tdep(1:3)	temperature growth factor	0.06, 0.06, 0.06	day^{-1}
f_{sal}	fsal	salinity growth factor	1	1
$R_{si:n}$	R_Si2N(1:3)	algal silicate to nitrogen	1.8, 0, 0	mol/mol

^{a} only (1:2) of DOC and DOC parameters have physical meaning

Biogeochemistry History Fields

The biogeochemical history fields specified in icefields_bgc_nml are written when 'x' is replaced with a time interval: step ('1'), daily ('d'), monthly ('m'), or yearly ('y'). Several of these flags turn on multiple history variables according to the particular ecosystem prescribed in **icepack_in**. For example, biogeochemical fluxes from the ice to ocean will be saved monthly in the history output if

```
f_fbio = 'm'
```

However, only the biogeochemical tracers which are active will be saved. This includes at most fNit nitrate, fAm ammonium, fN algal nitrogen, fDOC dissolved organic carbon, fDON dissolved organic nitrogen, fFep particulate iron, fFed dissolved iron, fSil silicate, fhum humic matter, fPON passive mobile tracer, fDMS DMS, fDMSPd dissolved DMSP and fDMSPp particulate DMSP.

Table 6 lists the biogeochemical tracer history flags along with a short description and the variable or variables saved. Not listed are flags appended with _ai, i.e. f_fbio_ai. These fields are identical to their counterpart. i.e. f_fbio, except they are averaged by ice area.

Table 6 :*Biogeochemical History variables*

Table 1.6: Table 6

History Flag	Definition	Variable(s)	Units
f_faero_atm	atmospheric aerosol deposition flux	faero_atm	kg m^{-2} s^{-1}
f_faero_ocn	aerosol flux from ice to ocean	faero_ocn	kg m^{-2} s^{-1}
f_aero	aerosol mass (snow and ice ssl and int)	aerosnossl, aerosnoint,aeroicessl, aeroiceint	kg/kg
f_fbio	biological ice to ocean flux	fN, fDOC, fNit, fAm,fDON,fFep^a, fFed^a, fSil,fhum, fPON, fDMSPd,fDMS, fDMSPp, fzaero	mmol m^{-2} s^{-1}
f_zaero	bulk z-aerosol mass fraction	zaero	kg/kg
f_bgc_S	bulk z-salinity	bgc_S	ppt
f_bgc_N	bulk algal N concentration	bgc_N	mmol m^{-3}
f_bgc_C	bulk algal C concentration	bgc_C	mmol m^{-3}
f_bgc_DOC	bulk DOC concentration	bgc_DOC	mmol m^{-3}
f_bgc_DON	bulk DON concentration	bgc_DON	mmol m^{-3}
f_bgc_DIC	bulk DIC concentration	bgc_DIC	mmol m^{-3}
f_bgc_chl	bulk algal chlorophyll concentration	bgc_chl	mg chl m^{-3}
f_bgc_Nit	bulk nitrate concentration	bgc_Nit	mmol m^{-3}
f_bgc_Am	bulk ammonium concentration	bgc_Am	mmol m^{-3}
f_bgc_Sil	bulk silicate concentration	bgc_Sil	mmol m^{-3}
f_bgc_DMSPp	bulk particulate DMSP concentration	bgc_DMSPp	mmol m^{-3}
f_bgc_DMSPd	bulk dissolved DMSP concentration	bgc_DMSPd	mmol m^{-3}
f_bgc_DMS	bulk DMS concentration	bgc_DMS	mmol m^{-3}
f_bgc_Fe	bulk dissolved and particulate iron conc.	bgc_Fed, bgc_Fep	μ mol m^{-3}
f_bgc_hum	bulk humic matter concentration	bgc_hum	mmol m^{-3}
f_bgc_PON	bulk passive mobile tracer conc.	bgc_PON	mmol m^{-3}
f_upNO	Total algal NO_3 uptake rate	upNO	mmol m^{-2} d^{-1}

Continued on next page

Table 1.6 – continued from previous page

History Flag	Definition	Variable(s)	Units
f_upNH	Total algal NH_4 uptake rate	upNH	mmol m^{-2} d^{-1}
f_bgc_ml	upper ocean tracer concentrations	ml_N, ml_DOC, ml_Nit,ml_Am, ml_DON, ml_Fep^b,ml_Fed^b, ml_Sil, ml_hum, ml_PON,ml_DMS, ml_DMSPd, ml_DMSPp	mmol m^{-3}
f_bTin	ice temperature on the bio grid	bTizn	^{o}C
f_bphi	ice porosity on the bio grid	bphizn	%
f_iDin	brine eddy diffusivity on the interface bio grid	iDin	m^2 d^{-1}
f_iki	ice permeability on the interface bio grid	ikin	mm^2
f_fbri	ratio of brine tracer height to ice thickness	fbrine	1
f_hbri	brine tracer height	hbrine	m
f_zfswin	internal ice PAR on the interface bio grid	zfswin	W m^{-2}
f_bionet	brine height integrated tracer concentration	algalN_net, algalC_net, chl_net, pFe^c_net, dFe^c_net, Sil_net, Nit_net, Am_net, hum_net, PON_net, DMS_net, DMSPd_net, DMSPp_net, DOC_net, zaero_net, DON_net	mmol m^{-2}
f_biosnow	snow integrated tracer concentration”	algalN_snow, algalC_snow,chl_snow, pFe^c_snow, dFe^c_snow,Sil_snow, Nit_snow, Am_snow, hum_snow, PON_snow, DMS_snow, DMSPd_snow, DMSPp_snow, DOC_snow, zaero_snow, DON_snow	mmol m^{-2}
f_grownet	Net specific algal growth rate	grow_net	m d^{-1}
f_PPnet	Net primary production	PP_net	mgC m^{-2} d^{-1}
f_algalpeak	interface bio grid level of peak chla	peak_loc	1
f_zbgc_frac	mobile fraction of tracer	algalN_frac, chl_frac, pFe_frac,dFe_frac, Sil_frac, Nit_frac,Am_frac, hum_frac, PON_frac,DMS_frac, DMSPd_frac, DMSPp_frac,DOC_frac, zaero_frac, DON_frac	1

[a] units are μmol m^{-2} s^{-1}

[b] units are μmol m^{-3}

[c] units are μmol m^{-2}

Flux-Corrected, Positive Definite Transport Scheme

Numerical solution of the vertical tracer transport equation is accomplished using the finite element Galerkin discretization. Multiply [eqn:mobile_transport] by "w" and integrate by parts

$$
\begin{aligned}
&\int_h \left[w\frac{\partial C}{\partial t} - \frac{\partial w}{\partial x}\left(-\left[\frac{v}{h}+\frac{w_f}{h\phi}\right]C + \frac{D_{in}}{\phi^2}\frac{\partial \phi}{\partial x}C - \frac{D_{in}}{\phi}\frac{\partial C}{\partial x}\right)\right]dx \\
+ \quad & w\left(-\left[\frac{1}{h}\frac{dh_b}{dt}+\frac{w_f}{h\phi}\right]C + \frac{D_{in}}{\phi^2}\frac{\partial \phi}{\partial x}C - \frac{D_{in}}{\phi}\frac{\partial C}{\partial x}\right)\Bigg|_{bottom} + w\left[\frac{1}{h}\frac{dh_t}{dt}+\frac{w_f}{h\phi}\right]C|_{top} = 0
\end{aligned}
$$

The bottom boundary condition indicated by $|_{bottom}$ satisfies

$$
\begin{aligned}
-w\left(-\left[\frac{1}{h}\frac{dh_b}{dt}+\frac{w_f}{h\phi}\right]C + \frac{D_{in}}{\phi^2}\frac{\partial \phi}{\partial x}C - \frac{D_{in}}{\phi}\frac{\partial C}{\partial x}\right)\Bigg|_{bottom} &= \\
w\left[\frac{1}{h}\frac{dh_b}{dt}+\frac{w_f}{h\phi_{N+1}}\right](C_{N+2} \text{ or } C_{N+1}) - w\frac{D_{in}}{\phi_{N+1}(\Delta h+g_o)}(C_{N+1}-C_{N+2})
\end{aligned}
$$

where $C_{N+2} = h\phi_{N+1}[c]_{ocean}$ and $w = 1$ at the bottom boundary and the top. The component C_{N+2} or C_{N+1} depends on the sign of the advection boundary term. If $dh_b + w_f/\phi > 0$ then use C_{N+2} otherwise C_{N+1}.

Define basis functions as linear piecewise, with two nodes (boundary nodes) in each element. Then for $i > 1$ and $i < N+1$

$$
w_i(x) = \begin{cases} 0 & x < x_{i-1} \\ (x-x_{i-1})/\Delta & x_{i-1} < x \le x_i \\ 1-(x-x_i)/\Delta & x_i \le x < x_{i+1} \\ 0, & x \ge x_{i+1} \end{cases}
$$

For $i = 1$

$$
w_1(x) = \begin{cases} 1-x/\Delta & x < x_2 \\ 0, & x \ge x_2 \end{cases}
$$

and $i = N+1$

$$
w_{N+1}(x) = \begin{cases} 0, & x < x_N \\ (x-x_N)/\Delta & x \ge x_N \end{cases}
$$

Now assume a form

$$
C_h = \sum_j^{N+1} c_j w_j
$$

Then

$$
\begin{aligned}
\int_h C_h dx = \quad & c_1\int_0^{x_2}\left(1-\frac{x}{\Delta}\right)dx + c_{N+1}\int_{x_N}^{x_{N+1}}\frac{x-x_N}{\Delta}dx \\
+ \quad & \sum_{j=2}^{N} c_j\left\{\int_{j-1}^{j}\frac{x-x_{j-1}}{\Delta}dx + \int_j^{j+1}\left[1-\frac{(x-x_j)}{\Delta}\right]dx\right\} \\
= \quad & \Delta\left[\frac{c_1}{2}+\frac{c_{N+1}}{2}+\sum_{j=2}^{N}c_j\right]
\end{aligned}
$$

Now this approximate solution form is substituted into the variational equation with $w = w_h \in \{w_j\}$

$$
\begin{aligned}
0 = \quad & \int_h \left[w_h\frac{\partial C_h}{\partial t} - \frac{\partial w_h}{\partial x}\left(\left[-\frac{v}{h}-\frac{w_f}{h\phi}+\frac{D_{in}}{\phi^2}\frac{\partial \phi}{\partial x}\right]C_h - \frac{D_{in}}{\phi}\frac{\partial C_h}{\partial x}\right)\right]dx \\
+ \quad & w_h\left(-\left[\frac{1}{h}\frac{dh_b}{dt}+\frac{w_f}{h\phi}\right]C_h + \frac{D_{in}}{\phi^2}\frac{\partial \phi}{\partial x}C - \frac{D_{in}}{\phi}\frac{\partial C_h}{\partial x}\right)\Bigg|_{bottom} + w_h\left[\frac{1}{h}\frac{dh_t}{dt}+\frac{w_f}{h\phi}\right]C_h|_{top}
\end{aligned}
$$

The result is a linear matrix equation

$$M_{jk}\frac{\partial C_k(t)}{\partial t} = [K_{jk} + S_{jk}]C_k(t) + q_{in}$$

where

$$\begin{aligned}
M_{jk} = & \int_h w_j(x) w_k(x) dx \\
K_{jk} = & \left[-\frac{v}{h} - \frac{w_f}{h\phi} + \frac{D_{in}}{\phi^2}\frac{\partial \phi}{\partial x}\right]\int_h \frac{\partial w_j}{\partial x} w_k dx \\
- & w_j\left(-\left[\frac{v}{h} + \frac{w_f}{h\phi_k}\right]w_k + \frac{D_{in}}{\phi^2}\frac{\partial \phi_k}{\partial x}w_k - \frac{D_{in}}{\phi_k}\frac{\partial w_k}{\partial x}\right)\Bigg|_{bot} \\
= & -V_k\int_h \frac{\partial w_j}{\partial x} w_k dx - w_j\left(-V_k w_k - \frac{D_{in}}{\phi_k}\frac{\partial w_k}{\partial x}\right)\Bigg|_{bot} \\
S_{jk} = & -\frac{D_{in}}{\phi_k}\int_h \frac{\partial w_j}{\partial x}\cdot\frac{\partial w_k}{\partial x}dx \\
q_{in} = & -VC_t w_j(x)|_t
\end{aligned}$$

And $C_{N+2} = h\phi_{N+1}[c]_{ocean}$

For the top condition q_{in} is applied to the upper value C_2 when $VC_t < 0$, i.e. q_{in} is a source.

Compute the M_{jk} integrals:

$$\begin{aligned}
M_{jj} = & \int_{x_{j-1}}^{x_j}\frac{(x-x_{j-1})^2}{\Delta^2}dx + \int_{x_j}^{x_{j+1}}\left[1 - \frac{(x-x_j)}{\Delta}\right]^2 dx = \frac{2\Delta}{3} \quad \text{for } 1 < j < N+1 \\
M_{11} = & \int_{x_1}^{x_2}\left[1 - \frac{(x-x_2)}{\Delta}\right]^2 dx = \frac{\Delta}{3} \\
M_{N+1,N+1} = & \int_{x_N}^{x_{N+1}}\frac{(x-x_N)^2}{\Delta^2}dx = \frac{\Delta}{3}
\end{aligned}$$

Off diagonal components:

$$\begin{aligned}
M_{j,j+1} = & \int_{x_j}^{x_{j+1}}\left[1 - \frac{(x-x_j)}{\Delta}\right]\left[\frac{x-x_j}{\Delta}\right]dx = \frac{\Delta}{6} \quad \text{for } j < N+1 \\
M_{j,j-1} = & \int_{x_{j-1}}^{x_j}\left[\frac{x-x_{j-1}}{\Delta}\right]\left[1 - \frac{(x-x_{j-1})}{\Delta}\right]dx = \frac{\Delta}{6} \quad \text{for } j > 1
\end{aligned}$$

Compute the K_{jk} integrals:

$$\begin{aligned}
K_{jj} = & k'_{jj}\left[\int_{x_{j-1}}^{x_j}\frac{\partial w_j}{\partial x}w_j dx + \int_{x_j}^{x_{j+1}}\frac{\partial w_j}{\partial x}w_j dx\right] \\
= & \frac{1}{2} + -\frac{1}{2} = 0 \quad \text{for } 1 < j < N+1 \\
K_{11} = & -\frac{k'_{11}}{2} = \frac{1}{2}\left[\frac{v}{h} + \frac{w_f}{h\phi}\right] \\
K_{N+1,N+1} = & \frac{k'_{N+1,N+1}}{2} + \min\left[0, \left(\frac{1}{h}\frac{dh_b}{dt} + \frac{w_f}{h\phi_{N+1}}\right)\right] - \frac{D_{in}}{\phi_{N+1}(g_o/h)} \\
= & \left[-\frac{v}{h} - \frac{w_f}{h\phi} + \frac{D_{in}}{\phi^2}\frac{\partial\phi}{\partial x}\right]\frac{1}{2} + \min\left[0, \left(\frac{1}{h}\frac{dh_b}{dt} + \frac{w_f}{h\phi_{N+1}}\right)\right] - \frac{D_{in}}{\phi_{N+1}(g_o/h)}
\end{aligned}$$

Off diagonal components:

$$\begin{aligned}
K_{j(j+1)} = & \quad k'_{j(j+1)} \int_{x_j}^{x_{j+1}} \frac{\partial w_j}{\partial x} w_{j+1} dx = -k'_{j(j+1)} \int_{x_j}^{x_{j+1}} \frac{(x-x_j)}{\Delta^2} dx \\
= & \quad -\frac{k'_{j(j+1)}}{\Delta^2}\frac{\Delta^2}{2} = -\frac{k'_{j(j+1)}}{2} = p5 * \left[\frac{v}{h} + \frac{w_f}{h\phi} - \frac{D_{in}}{\phi^2}\frac{\partial \phi}{\partial x}\right]_{(j+1)} \quad \text{for } j < N+1 \\
K_{j(j-1)} = & \quad k'_{j(j-1)} \int_{x_{j-1}}^{x_j} \frac{\partial w_j}{\partial x} w_{j-1} dx = k'_{j(j-1)} \int_{x_{j-1}}^{x_j} \left[1 - \frac{(x-x_{j-1})}{\Delta^2}\right] dx \\
= & \quad \frac{k'_{j(j-1)}}{\Delta^2}\frac{\Delta^2}{2} = \frac{k'_{j(j-1)}}{2} = -p5 * \left[\frac{v}{h} + \frac{w_f}{h\phi} - \frac{D_{in}}{\phi^2}\frac{\partial \phi}{\partial x}\right]_{(j-1)} \quad \text{for } j > 1
\end{aligned}$$

for $K_{N+1,N}$, there's a boundary contribution .

$$K_{N+1,N} = \frac{k'_{N+1(N)}}{2} - \frac{D_N}{\Delta \phi_N}$$

Note. The bottom condition works if $C_{bot} = h\phi_{N+2}[c]_{ocean}$, ϕ^2 is $\phi_{N+1}\phi_{N+2}$ and

$$\left.\frac{\partial \phi}{\partial x}\right|_{bot} = \frac{\phi_{N+2} - \phi_N}{2\Delta}$$

Then the $D_{N+1}/\phi_{N+1}/\Delta$ cancels properly with the porosity gradient. In general

$$\left.\frac{\partial \phi}{\partial x}\right|_{k} = \frac{\phi_{k+2} - \phi_k}{2\Delta}$$

When evaluating the integrals for the diffusion term, we will assume that D/ϕ is constant in an element. For $D_{in}/i\phi$ defined on interface points, $D_1 = 0$ for $j = 2, ..., N$ $D_j/b\phi_j = (D_{in}(j) + D_{in}(j+1))/(i\phi_j + i\phi_{j+1})$. Then the above integrals will be modified as follows:

Compute the S_{jk} integrals:

$$\begin{aligned}
S_{jj} = & \quad -\left[\frac{D_{j-1}}{b\phi_{j-1}} \int_{x_{j-1}}^{x_j} \left(\frac{\partial w_j}{\partial x}\right)^2 dx + \frac{D_j}{b\phi_j} \int_{x_j}^{x_{j+1}} \left(\frac{\partial w_j}{\partial x}\right)^2 dx\right] \\
= & \quad -\frac{1}{\Delta}\left[\frac{D_{j-1}}{b\phi_{j-1}} + \frac{D_j}{b\phi j}\right] \quad \text{for } 1 < j < N+1 \\
S_{11} = & \quad \frac{s'_{11}}{\Delta} = 0 \\
S_{N+1,N+1} = & \quad \frac{s'_{N+1,N+1}}{\Delta} = -\frac{(D_N)}{b\phi_N \Delta}
\end{aligned}$$

Compute the off-diagonal components of S_{jk}:

$$\begin{aligned}
S_{j(j+1)} = & \quad s'_{j(j+1)} \int_{x_j}^{x_{j+1}} \frac{\partial w_j}{\partial x}\frac{\partial w_{j+1}}{\partial x} dx = -\frac{s'_{j(j+1)}}{\Delta} = \frac{D_j}{b\phi_j \Delta} \quad \text{for } j < N+1 \\
S_{j(j-1)} = & \quad s'_{j(j-1)} \int_{x_{j-1}}^{x_j} \frac{\partial w_j}{\partial x}\frac{\partial w_{j-1}}{\partial x} dx = -\frac{s'_{j(j-1)}}{\Delta} = \frac{D_{j-1}}{b\phi_{j-1}} \quad \text{for } j > 1
\end{aligned}$$

We assume that $D/\phi^2 \partial\phi/\partial x$ is constant in the element i. If D/ϕ_j is constant, and $\partial\phi/\partial x$ is constant then both the Darcy and D terms go as ϕ^{-1}. Then $\phi = (\phi_j - \phi_{j-1})(x - x_j)/\Delta + \phi_j$ and $m = (\phi_j - \phi_{j-1})/\Delta$ and $b = \phi_j - mx_j$.

The first integral contribution to the Darcy term is:

$$\begin{aligned}
K^1_{jj} = & \frac{-1}{\Delta^2}\left(\frac{w_f}{h} - \frac{D}{\phi}\frac{\partial\phi}{\partial x}\right)\int_{j-1}^{j}(x - x_{j-1})\frac{1}{mx+b}dx \\
= & -\left(\frac{w_f}{h} - \frac{D}{\phi}\frac{\partial\phi}{\partial x}\right)\frac{1}{\Delta^2}\left[\int_{j-1}^{j}\frac{x}{mx+b}dx - x_{j-1}\int_{j-1}^{j}\frac{1}{mx+b}dx\right] \\
= & -\left(\frac{w_f}{h} - \frac{D}{\phi}\frac{\partial\phi}{\partial x}\right)\frac{1}{\Delta^2}\left[\frac{mx - b\log(b+mx)}{m^2} - x_{j-1}\frac{\log(b+mx)}{m}\right]^{x_j}_{x_{j-1}} \\
= & -\left(\frac{w_f}{h} - \frac{D}{\phi}\frac{\partial\phi}{\partial x}\right)\frac{1}{\Delta_\phi}\left[1 + \log\left(\frac{\phi_j}{\phi_{j-1}}\right) - \frac{\phi_j}{\Delta_{\phi_j}}\log\left(\frac{\phi_j}{\phi_{j-1}}\right)\right] \\
= & -\left(\frac{w_f}{h} - \frac{D}{\phi}\frac{\partial\phi}{\partial x}\right)\frac{1}{\Delta_\phi}\left[1 + \frac{\phi_{j-1}}{\Delta_\phi}\log\left(\frac{\phi_j}{\phi_{j-1}}\right)\right]
\end{aligned}$$

$$\begin{aligned}
K^2_{jj} = & \left(\frac{w_f}{h} - \frac{D}{\phi}\frac{\partial\phi}{\partial x}\right)\frac{1}{\Delta}\int_{x_j}^{x_{j+1}}\left[1 - \frac{(x - x_j)}{\Delta}\right]\frac{1}{mx+b}dx \\
= & \left(\frac{w_f}{h} - \frac{D}{\phi}\frac{\partial\phi}{\partial x}\right)\frac{1}{\Delta}\left[\frac{(b + m(x_j+\Delta))\log(b+mx) - mx}{\Delta m^2}\right]^{x_{j+1}}_{x_j} \\
= & \left(\frac{w_f}{h} - \frac{D}{\phi}\frac{\partial\phi}{\partial x}\right)\frac{1}{\Delta_\phi}\left[1 - \frac{\phi_{j+1}}{\Delta_\phi}\log\left(\frac{\phi_{j+1}}{\phi_j}\right)\right]
\end{aligned}$$

Now $m = (\phi_{j+1} - \phi_j)/\Delta$ and $b = \phi_{j+1} - mx_{j+1}$.

Source terms $q_{bot} = q_{N+1}$ and $q_{top} = q_1$ (both positive)

$$\begin{aligned}
q_{bot} = & \max\left[0, \left(\frac{1}{h}\frac{dh_b}{dt} + \frac{w_f}{h\phi_{N+1}}\right)\right]C|_{bot} + \frac{D_{in}}{\phi_{N+1}(g_o/h)}C|_{bot} \\
C|_{bot} = & \phi_{N+1}[c]_{ocean}
\end{aligned}$$

where g_o is not zero.

$$\begin{aligned}
q_{in} = & -\min\left[0, \left(\frac{1}{h}\frac{dh_t}{dt} + \frac{w_f}{h\phi}\right)C|_{top}\right] \\
C|_{top} = & h[c]_o\phi_{min}
\end{aligned}$$

Calculating the low order solution: 1) Find the lumped mass matrix $M_l = diag\{m_i\}$

$$\begin{aligned}
m_j = & \sum_i m_{ji} = m_{j(j+1)} + m_{j(j-1)} + m_{jj} \\
= & \frac{\Delta}{6} + \frac{\Delta}{6} + \frac{2\Delta}{3} = \Delta \quad \text{for } 1 < j < N+1 \\
m_1 = & m_{11} + m_{12} = \frac{\Delta}{3} + \frac{\Delta}{6} = \frac{\Delta}{2} \\
m_{N+1} = & m_{N+1,N} + m_{N+1,N+1} = \frac{\Delta}{6} + \frac{\Delta}{3} = \frac{\Delta}{2}
\end{aligned}$$

2. Define artificial diffusion D_a

$$\begin{aligned}
d_{j,(j+1)} = & \max\{-k_{j(j+1)}, 0, -k_{(j+1)j}\} = d_{(j+1)j} \\
d_{jj} = & -\sum_{i\neq j} d_{ji}
\end{aligned}$$

3) Add artificial diffusion to K: $L = K + D_a$. 4) Solve for the low order predictor solution:

$$(M_l - \Delta t[L + S])C^{n+1} = M_l C^n + \Delta t q$$

Conservations terms for the low order solution are:

$$\begin{aligned}\int \left[C^{n+1} - C^n\right] w(x)dx = & \quad \Delta \left[\frac{c_1^{n+1} - c_1^n}{2} + \frac{c_{N+1}^{n+1} - c_{N+1}^n}{2} + \sum_{j=2}^{N}(c_j^{n+1} - c_j^n)\right] \\ = & \quad \Delta t \left[q_{bot} + q_{in} + (K_{N+1,N+1} + K_{N,N+1})C_{N+1}^{n+1} + (K_{1,1} + K_{2,1})C_1^{n+1}\right]\end{aligned}$$

Now add the antidiffusive flux: 1) compute the F matrix using the low order solution c^{n+1}. Diagonal components are zero. For $i \neq j$

$$f_{ij} = \quad m_{ij}\left[\frac{\Delta c_i}{\Delta t} - \frac{\Delta c_j}{\Delta t} + d_{ij}(c_i^{n+1} - c_j^{n+1}\right]$$

1.3 User Guide

1.3.1 Implementation

Icepack is written in FORTRAN90 and runs on platforms using UNIX, LINUX, and other operating systems. The code is not parallelized. (CHANGE IF OPENMP IS IMPLEMENTED)

Icepack consists of the sea ice column physics code, contained in the **columnphysics/** directory, and a **configuration/** directory that includes a driver for testing the column physics and a set of scripts for configuring the tests. Icepack is designed such that the column physics code may be used by a host sea ice model without direct reference to the driver or scripts, although these may be consulted for guidance when coupling the column physics code to the host sea ice model (CICE may also be useful for this.) Information about the interface between the column physics and the driver or host sea ice model is located in the *Initialization and Forcing* section.

Directory structure

The present code distribution includes source code for the column physics, source code for the driver, and the scripts. Forcing data is available from the ftp site. The directory structure of Icepack is as follows. All columnphysics filename have a prefix of icepack_ and all driver files are prefixed with icedrv_*.

LICENSE.pdf license and policy for using and sharing the code

DistributionPolicy.pdf license and policy for using and sharing the code

README.md basic information and pointers

columnphysics/ directory for columnphysics source code, see *Icepack Column Physics*

configuration/scripts/ directory of support scripts, see *Scripts Implementation*

configuration/driver/ directory of icepack driver code, see *Driver Implementation*

doc/ documentation

icepack.create.case main icepack script for creating cases

A case (compile) directory is created upon initial execution of the script **icepack.create.case** at the user-specified location provided after the -c flag. Executing the command `./icepack.create.case -h` provides helpful information for this tool.

Grid and boundary conditions

The driver configures a collection of grid cells on which the column physics code will be run. This "horizontal" grid is a vector of length `nx`, with a minimum length of 4. The grid vector is initialized with different sea ice conditions, such as open water, a uniform slab of ice, a multi-year ice thickness distribution with snow, and land. For simplicity, the same forcing values are applied to all grid cells.

Icepack includes two vertical grids. The basic vertical grid contains `nilyr` equally spaced grid cells. History variables available for column output are ice and snow temperature, `Tinz` and `Tsnz`. These variables also include thickness category as a fourth dimension.

In addition, there is a bio-grid that can be more finely resolved and includes additional nodes for boundary conditions. It is used for solving the brine height variable h_b and for discretizing the vertical transport equations of biogeochemical tracers. The bio-grid is a non-dimensional vertical grid which takes the value zero at h_b and one at the ice–ocean interface. The number of grid levels is specified during compilation by setting the variable `NBGCLYR` equal to an integer (n_b) .

Ice tracers and microstructural properties defined on the bio-grid are referenced in two ways: as `bgrid` $= n_b + 2$ points and as igrid= $n_b + 1$ points. For both bgrid and igrid, the first and last points reference h_b and the ice–ocean interface, respectively, and so take the values 0 and 1, respectively. For bgrid, the interior points $[2, n_b + 1]$ are spaced at $1/n_b$ intervals beginning with *bgrid(2)* = $1/(2n_b)$. The `igrid` interior points $[2, n_b]$ are also equidistant with the same spacing, but physically coincide with points midway between those of `bgrid`.

Test configurations

(CHECK) UPDATE with similar, correct information

The column is located near Barrow (71.35N, 156.5W). Options for choosing the column configuration are given in **comp_ice** (choose *RES col*) and in the namelist file, **input_templates/col/ice_in**. Here, `istep0` and the initial conditions are set such that the run begins September 1 with no ice.

Initialization and Forcing

Icepack's parameters and variables are initialized in several steps. Many constants and physical parameters are set in **icepack_constants.F90**. In the current driver implementation, a namelist file is read to setup the model. Namelist values are given default values in the code, which may then be changed when the input file **icepack_in** is read. Other physical constants, numerical parameters, and variables are first set in initialization routines for each ice model component or module. Then, if the ice model is being restarted from a previous run, core variables are read and reinitialized in *restartfile*, while tracer variables needed for specific configurations are read in separate restart routines associated with each tracer or specialized parameterization. Finally, albedo and other quantities dependent on the initial ice state are set. Some of these parameters will be described in more detail in the *Table of namelist inputs*.

Two namelist variables control model initialization, `ice_ic` and `restart`. Setting `ice_ic` = 'default' causes the model to run using initial values set in the code. To start from a file **filename**, set `restart` = .true. and `ice_ic` = **filename**. When restarting using the Icepack driver, for simplicity the tracers are assumed to be set the same way (on/off) as in the run that created the restart file; i.e. that the restart file contains exactly the information needed for the new run. CICE is more flexible in this regard.

For stand-alone runs, routines in **icedrv_forcing.F90** read and interpolate data from files, and are intended merely for testing, although they can also provide guidance for the user to write his or her own routines.

Choosing an appropriate time step

Transport in thickness space imposes a restraint on the time step, given by the ice growth/melt rate and the smallest range of thickness among the categories, $\Delta t < \min(\Delta H)/2\max(f)$, where ΔH is the distance between category boundaries and f is the thermodynamic growth rate. For the 5-category ice thickness distribution used as the default in this distribution, this is not a stringent limitation: $\Delta t < 19.4$ hr, assuming $\max(f) = 40$ cm/day.

Model output

History output from Icepack is not currently supported in the Icepack driver, except in restart files. The sea ice model CICE provides extensive options for model output, including many derived output variables.

Diagnostic files

Icepack writes diagnostic information for each grid cell as a separate file, **ice_diag.***, identified by the initial ice state of the grid cell (no ice, slab, land, etc).

Restart files

CHECK and CHANGE as needed re netCDF

CICE provides restart data in binary unformatted or netCDF formats, via the `IO_TYPE` flag in **comp_ice** and namelist variable `restart_format`.

The restart files created by the Icepack driver contain all of the variables needed for a full, exact restart. The filename begins with the character string 'iced.', and the restart dump frequency is given by the namelist variable `dumpfreq`. The namelist variable `ice_ic` contains the pointer to the filename from which the restart data is to be read.

1.3.2 Running Icepack

Quick-start instructions are provided in the *Quick Start* section.

Scripts

The icepack scripts are written to allow quick setup of cases and tests. Once a case is generated, users can manually modify the namelist and other files to custom configure the case. Several settings are available via scripts as well.

Most of the scripts that configure, build and run Icepack are contained in the directory **configuration/scripts/**, except for **icepack.create.case**, which is in the main directory. **icepack.create.case** is the main script that generates a case.

Users may need to port the scripts to their local machine. Specific instructions for porting are provided in *Porting*.

`icepack.create.case -h` will provide the latest information about how to use the tool. There are three usage modes,

- `-c` creates individual stand alone cases.
- `-t` creates individual tests. Tests are just cases that have some extra automation in order to carry out particular tests such as exact restart.
- `-ts` creates a test suite. Test suites are predefined sets of tests and `-ts` provides the ability to quick setup, build, and run a full suite of tests.

All modes will require use of `-m` to specify the machine and case and test modes can use `-s` to define specific options. `-t` and `-ts` will require `-testid` to be set and both of the test modes can use `-bd`, `-bg`, `-bc`, and `-td` to compare with other results. Testing will be described in greater detail in the *Testing Icepack* section.

Again, `icepack.create.case -h` will show the latest usage information including the available `-s` options, the current ported machines, and the test choices.

To create a case, run **icepack.create.case**:

```
icepack.create.case -c mycase -m machine
cd mycase
```

Once a case/test is created, several files are placed in the case directory

- **env.[machine]** defines the environment
- **icepack.settings** defines many variables associated with building and running the model
- **makdep.c** is a tool that will automatically generate the make dependencies
- **Macros.[machine]** defines the Makefile macros
- **Makefile** is the makefile used to build the model
- **icepack.build** is a script that builds and compiles the model
- **icepack_in** is the namelist input file
- **icepack.run** is a batch run script
- **icepack.submit** is a simple script that submits the icepack.run script

All scripts and namelist are fully resolved in the case. Users can edit any of the files in the case directory manually to change the model configuration, build options, or batch settings. The file dependency is indicated in the above list. For instance, if any of the files before **icepack.build** in the list are edited, **icepack.build** should be rerun.

The **casescripts/** directory holds scripts used to create the case and can largely be ignored. Once a case is created, the **icepack.build** script should be run interactively and then the case should be submitted by executing the **icepack.submit** script interactively. The **icepack.submit** script simply submits the **icepack.run script**. You can also submit the **icepack.run** script on the command line.

Some hints:

- To change namelist, manually edit the **icepack_in** file
- To change batch settings, manually edit the top of the **icepack.run** file
- To turn on the debug compiler flags, set `ICE_BLDDEBUG` in **icepack.setttings** to true
- To change compiler options, manually edit the Macros file
- To clean the build before each compile, set `ICE_CLEANBUILD` in **icepack.settings** to true To not clean before the build, set `ICE_CLEANBUILD` in **icepack.settings** to false

To build and run:

```
./icepack.build
./icepack.submit
```

The build and run log files will be copied into the logs directory in the case directory. Other model output will be in the run directory. The run directory is set in **icepack.settings** via the **ICE_RUNDIR** variable. To modify the case setup, changes should be made in the case directory, NOT the run directory.

Porting

To port, an **env.[machine]** and **Macros.[machine]** file have to be added to the **configuration/scripts/machines/** directory and the **configuration/scripts/icepack.run.setup.csh** file needs to be modified.

- cd to **configuration/scripts/machines/**
- Copy an existing env and a Macros file to new names for your new machine

- Edit your env and Macros files
- cd .. to **configuration/scripts/**
- Edit the **icepack.run.setup.csh** script to add a section for your machine with batch settings and job launch settings
- Download and untar a forcing dataset to the location defined by `ICE_MACHINE_INPUTDATA` in the env file

In fact, this process almost certainly will require some iteration. The easiest way to carry this out is to create an initial set of changes as described above, then create a case and manually modify the **env.[machine]** file and **Macros.[machine]** file until the case can build and run. Then copy the files from the case directory back to **configuration/scripts/machines/** and update the **configuration/scripts/icepack.run.setup.csh** file, retest, and then add and commit the updated machine files to the repository.

Input Data

The input data space is defined on a per machine basis by the `ICE_MACHINE_INPUTDATA` variable in the **env.[machine]** file. That file space is often shared among multiple users, and it can be desirable to consider using a common file space with group read and write permissions such that a set of users can update the inputdata area as new datasets are available.

The latest input data will be available thru the CICE Consortium github wiki.

Machine Account Settings

The machine account default is specified by the variable `ICE_MACHINE_ACCT` in the **env.[machine]** file. The easiest way to change a user's default is to create a file in your home directory called **.cice_proj** and add your preferred account name to the first line. There is also an option (`-a`) in **icepack.create.case** to define the account number. The order of precedent is **icepack.create.case** command line option, **.cice_proj** setting, and then value in the **env.[machine]** file.

Run Directories

The **icepack.create.case** script creates a case directory. However, the model is actually built and run under the `ICE_OBJDIR` and `ICE_RUNDIR` directories as defined in the **icepack.settings** file.

Build and run logs will be copied from the run directory into the case **logs/** directory when complete.

Local modifications

Scripts and other case settings can be changed manually in the case directory and used. Source code can be modified in the main sandbox. When changes are made, the code should be rebuilt before being resubmitted. It is always recommended that users modify the scripts and input settings in the case directory, NOT the run directory. In general, files in the run directory are overwritten by versions in the case directory when the model is built, submitted, and run.

Forcing data

CHECK once we've settled on a forcing suite:

The code is currently configured to run in standalone mode on a 4-cell grid using atmospheric data, available as detailed on the wiki. These data files are designed only for testing the code, not for use in production runs or as observational data. Please do not publish results based on these data sets. Module **configuration/driver/icedrv_forcing.F90** can be modified to change the forcing data.

1.3.3 Testing Icepack

The section documents primarily how to use the Icepack scripts to carry out icepack testing. Exactly what to test is a separate question and depends on the kinds of code changes being made. Prior to merging changes to the CICE Consortium master, changes will be reviewed and developers will need to provide a summary of the tests carried out.

There is a base suite of tests provided by default with Icepack and this may be a good starting point for testing.

Individual Tests

The Icepack scripts support both setup of individual tests as well as test suites. Individual tests are run from the command line:

```
./icepack.create.case -t smoke -m wolf -s diag1,debug -testid myid
```

where `-m` designates a specific machine and testid is a user defined string that allows test cases to be uniquely identified. The format of the case directory name for a test will always be `[machine]_[test]_[grid]_[pes]_[soptions].[testid]`

To build and run a test, the process is the same as a case. cd to the test directory, run the build script, and run the submit script:

```
cd [test_case]
./icepack.build
./icepack.submit
```

The test results will be copied into a local file called **test_output**. To check those results:

```
cat test_output
```

Tests are defined under **configuration/scripts/tests/**. The tests currently supported are:

- **smoke - Runs the model for default length. The length and options can** be set with the `-s` command line option. The test passes if the model completes successfully.
- **restart - Runs the model for 14 months, writing a restart file at month 3 and** again at the end of the run. Runs the model a second time starting from the month 3 restart and writing a restart at month 12 of the model run. The test passes if both runs complete and if the restart files at month 12 from both runs are bit-for-bit identical.

Please run `./icepack.create.case -h` for the latest information.

Test suites

Test suites are multiple tests that are specified via an input file. When invoking the test suite option (`-ts`) with **icepack.create.case**, all tests will be created, built, and submitted automatically under a directory called [suite_name].[testid]:

```
./icepack.create.case -ts base_suite -m wolf -testid myid
```

Like an individual test, the `-testid` option must be specified and can be any string. Once the tests are complete, results can be checked by running the results.csh script in the [suite_name].[testid]:

```
cd base_suite.[testid]
./results.csh
```

Please run `./icepack.create.case -h` for additional details.

The predefined test suites are defined under **configuration/scripts/tests** and the files defining the suites have a suffix of .ts in that directory. The format for the test suite file is relatively simple. It is a text file with white space delimited columns, e.g. **base_suite.ts**

Table 1.7: Table 7

Test	Grid	PEs	Sets	BFB-compare
smoke	col	1x1	diag1	
smoke	col	1x1	diag1,run1year	smoke_col_1x1_diag1_run1year
smoke	col	1x1	debug,run1year	
restart	col	1x1	debug	
restart	col	1x1	diag1	
restart	col	1x1	pondcesm	
restart	col	1x1	pondlvl	
restart	col	1x1	pondtopo	

The first column is the test name, the second the grid, the third the pe count, the fourth column is the `-s` options and the fifth column is the `-td` argument. (The grid and PEs columns are provided for compatibility with the similar CICE scripts.) The fourth and fifth columns are optional. The argument to `-ts` defines which filename to choose and that argument can contain a path. **icepack.create.case** will look for the filename in the local directory, in **configuration/scripts/tests/**, or in the path defined by the `-ts` option.

Regression testing

The **icepack.create.case** options `-bg`, `-bc`, and `-bd` are used for regression testing. There are several additional options on the **icepack.create.case** command line for testing that provide the ability to regression test and compare tests to each other. These options only work in test (`-t`) or test suite (`-ts`) mode, not in case (`-c`) mode.

> `-bd` defines a top level directory where tests can be stored for regression testing. The default is defined by `ICE_MACHINE_BASELINE` defined in `env.[machine]`.
>
> `-bg` defines a directory name where the current tests can be saved for regression testing. It's handy for this name to be related to the model version. This directory will be created below the directory associated with `-bd`.
>
> `-bc` defines the directory name that the current tests should be compared to for regression testing. This directory will be added to the directory associated with `-bd`.

To create a baseline, use `-bg`:

```
icepack.create.case -ts base_suite -m wolf -testid v1 -bg version1 -bd $SCRATCH/
↪ICEPACK_BASELINES
```

will copy all the results from the test suite to `$SCRATCH/ICEPACK_BASELINES/version1`.

To compare to a prior result, use `-bc`:

```
icepack.create.case -ts base_suite -m wolf -testid v2 -bc version1 -bd $SCRATCH/
↪ICEPACK_BASELINES
```

will compare all the results from this test suite to results saved before in $SCRATCH/ICEPACK_BASELINES/version1‘‘.

To both create and compare, `-bc` and `-bg` can be combined:

```
icepack.create.case -ts base_suite -m wolf -testid v2 -bg version2 -bc version1 -bd
↪$SCRATCH/ICEPACK_BASELINES
```

will save the current results to `$SCRATCH/ICEPACK_BASELINES/version2` and compare the current results to results save before in `$SCRATCH/ICEPACK_BASELINES/version1`.

In summary,

- an individual test will have a case name like `[machine]_[test]_[grid]_[pes]_[soptions].[testid]`.
- A test suite will generate the individual tests under a directory called `[suite_name].[testid]`.
- `-bg` will copy test results to the `[bd_directory]/[bg_directory]/[test_name]`.
- `-bc` will compare results from `[bd_directory]/[bc_directory]/[test_name]`.

Comparison testing

This feature is primarily used in test suites and has limited use in icepack, but is being described for completeness.

`-td` provides a way to compare tests with each other. The test is always compared relative to the current case directory. For instance:

```
icepack.create.case -t smoke -m wolf -testid t01
```

creates a test case named wolf_smoke_col_1x1.t01:

```
icepack.create.case -t smoke -m wolf -s run1year -testid t01 -td smoke_col_1x1
```

will create a test case named wolf_smoke_col_1x1_run1year.t01. An additional check will be done for the second test (because of the `-td` argument), and it will compare the output from the first test “smoke_col_1x1” to the output from its test “smoke_col_1x1_run1year” and generate a result for that. It’s important that the first test complete before the second test is done and that the tests are created in parallel directories. The `-td` option works only if the testid and the machine are the same for the baseline run and the current run, a basic feature associated with test suites.

Icepack Test Reporting

The Icepack testing scripts have the capability of posting the test results to an online dashboard, located on CDash. There are two ways to post Icepack results to CDash, an automated method and a manual method.

To automatically post test suite results to CDash, add the `-report` option to **icepack.create.case**. The base_suite will attempt to post the test results on CDash when the suite is complete.

To manually post the test results to the Icepack CDash dashboard, run a standard test suite. Then once all the tests are complete:

```
cd [suite_name].[testid]
./results.csh
ctest -S steer.cmake
```

Examples

To generate a baseline dataset for a test case

```
./icepack.create.case -t smoke -m wolf -bg icepackv6.0.0 -testid t00
cd wolf_smoke_col_1x1.t00
./icepack.build
./icepack.submit
```

After job finishes, check output:

```
cat test_output
```

To run a test case and compare to a baseline dataset

```
./icepack.create.case -t smoke -m wolf -bc icepackv6.0.0 -testid t01
cd wolf_smoke_col_1x1.t01
./icepack.build
./icepack.submit
```

After job finishes, check output:

```
cat test_output
```

To run a test suite to generate baseline data, review results, plot timeseries, and manually report results

```
./icepack.create.case -m wolf -ts base_suite -testid t02 -bg icepackv6.0.0bs
```

Once all jobs finish, concatenate all output and manually report results:

```
cd base_suite.t02
./results.csh
ctest -S steer.cmake
```

To plot a timeseries of "total ice extent", "total ice area", and "total ice volume":

```
./timeseries.csh <directory>
ls *.png
```

To run a test suite, compare to baseline data, generate a new baseline, and automatically report the results

```
./icepack.create.case -m wolf -ts base_suite -testid t03 -bc icepackv6.0.0bs -bg␣
↪icepackv6.0.0new -report
```

Once all jobs finish, concatenate all output:

```
cd base_suite.t03
./results.csh
```

1.3.4 Case Settings

There are two important files that define the case, **icepack.settings** and **icepack_in**. **icepack.settings** is a list of env variables that define many values used to setup, build and run the case. **icepack_in** is the input namelist file for the icepack driver. Variables in both files are described below.

Table of icepack settings

The **icepack.settings** file is reasonably well self documented. Several of the variables defined in the file are not used in Icepack. They exist to support the CICE model.

Table 1.8: Table 9

variable	options/format	description	recommended value
ICE_MACHINE		machine name	set by icepack.create.case
ICE_CASENAME		case name	set by icepack.create.case
ICE_SANDBOX		sandbox directory	set by icepack.create.case
ICE_SCRIPTS		scripts directory	set by icepack.create.case
ICE_CASEDIR		case directory	set by icepack.create.case
ICE_RUNDIR		run directory	set by icepack.create.case
ICE_OBJDIR		compile directory	${ICE_RUNDIR}/compile
ICE_RSTDIR		unused	${ICE_RUNDIR}/restart
ICE_HSTDIR		unused	${ICE_RUNDIR}/history
ICE_LOGDIR		log directory	${ICE_CASEDIR}/logs
ICE_RSTPFILE		unused	undefined
ICE_DRVOPT		unused	icepack
ICE_CONSTOPT		unused	cice
ICE_IOTYPE		unused	none
ICE_CLEANBUILD	true,false	automatically clean before building	true
ICE_GRID	col	grid	col
ICE_NXGLOB	4	number of gridcells	4
ICE_NTASKS	1	number of tasks, must be set to 1	1
ICE_NTHRDS	1	number of threads per task, must be set to 1	1
ICE_TEST		test setting if using a test	set by icepack.create.case
ICE_TESTNAME		test name if using a test	set by icepack.create.case
ICE_BASELINE		baseline directory name, associated with icepack.create.case -bd	set by icepack.create.case
ICE_BASEGEN		baseline directory name for regression generation, associated with icepack.create.case -bg	set by icepack.create.case
ICE_BASECOM		baseline directory name for regression comparison, associated with icepack.create.case -bc	set by icepack.create.case
ICE_BFBCOMP		location of case for comparison, associated with icepack.create.case -td	set by icepack.create.case
ICE_SPVAL		unused	UnDeFiNeD
ICE_RUNLENGTH		batch run length default	00:10:00

Continued on next page

Table 1.8 – continued from previous page

variable	options/format	description	recommended value
ICE_THREADED	true,false	force threading in compile, will always compile threaded if NTHRDS is gt 1	false
NICELYR		number of vertical layers in the ice	7
NSNWLYR		number of vertical layers in the snow	1
NICECAT		number of ice thickness categories	5
TRAGE	0,1	ice age tracer	1
TRFY	0,1	first-year ice area tracer	1
TRLVL	0,1	deformed ice tracer	1
TRPND	0,1	melt pond tracer	1
NTRAERO		number of aerosol tracers	1
TRBRI	0,1	brine height tracer	0
TRZS	0,1	zsalinity tracer, needs TRBRI=1	0
TRBGCS	0,1	skeletal layer tracer, needs TRBGCZ=0	0
TRBGCZ	0,1	zbgc tracers, needs TRBGCS=0 and TRBRI=1	0
NBGCLYR		number of zbgc layers	7
TRZAERO	0-6	number of z aerosol tracers	0
TRALG	0,1,2,3	number of algal tracers	0
TRDOC	0,1,2,3	number of dissolved organic carbon	0
TRDIC	0,1	number of dissolved inorganic carbon	0
TRDON	0,1	number of dissolved organic nitrogen	0
TRFEP	0,1,2	number of particulate iron tracers	0
TRFED	0,1,2	number of dissolved iron tracers	0
CAM_ICE		unused	no
DITTO		unused	no
BARRIERS		unused	no
ICE_BLDDEBUG	true,false	turn on compile debug flags	false
NUMIN		unused	11
NUMAX		unused	99

Table of namelist inputs

CHECK

Namelist is part of the icepack driver code and is used to setup testing of the column physics.

Table 1.9: Table 8

variable	options/format	description	recommended value
setup_nml			
		Time, Diagnostics	
`days_per_year`	`360` or `365`	number of days in a model year	365
`use_leap_years`	true/false	if true, include leap days	
`year_init`	yyyy	the initial year, if not using restart	

Continued on next page

Table 1.9 – continued from previous page

variable	options/format	description	recommended value
`istep0`	integer	initial time step number	0
`dt`	seconds	thermodynamics time step length	3600.
`npt`	integer	total number of time steps to take	
`ndtd`	integer	number of dynamics/advection/ridging/steps per thermo timestep	1
		Initialization/Restarting	
`ice_ic`	`default`	latitude and sst dependent	default
	`none`	no ice	
	path/file	restart file name	
`restart_dir`	path/	path to restart directory	
`dumpfreq`	`y`	write restart every `dumpfreq_n` years	y
	`m`	write restart every `dumpfreq_n` months	
	`d`	write restart every `dumpfreq_n` days	
		Model Output	
`diagfreq`	integer	frequency of diagnostic output in `dt`	24
	e.g., 10	once every 10 time steps	
`diag_file`	filename	diagnostic output file (script may reset)	
grid_nml			
		Grid	
`kcatbound`	`0`	original category boundary formula	0
	`1`	new formula with round numbers	
	`2`	WMO standard categories	
	`-1`	one category	
tracer_nml			
		Tracers	
`tr_iage`	true/false	ice age	
`tr_FY`	true/false	first-year ice area	
`tr_lvl`	true/false	level ice area and volume	
`tr_pond_cesm`	true/false	CESM melt ponds	
`tr_pond_topo`	true/false	topo melt ponds	
`tr_pond_lvl`	true/false	level-ice melt ponds	
`tr_aero`	true/false	aerosols	
thermo_nml			
		Thermodynamics	
`kitd`	`0`	delta function ITD approximation	1
	`1`	linear remapping ITD approximation	
`ktherm`	`0`	zero-layer thermodynamic model	
	`1`	Bitz and Lipscomb thermodynamic model	
	`2`	mushy-layer thermodynamic model	
`conduct`	`MU71`	conductivity [34]	
	`bubbly`	conductivity [37]	
`a_rapid_mode`	real	brine channel diameter	0.5x10 $^{-3}$ m

Continued on next page

Table 1.9 – continued from previous page

variable	options/format	description	recommended value
`Rac_rapid_mode`	real	critical Rayleigh number	10
`aspect_rapid_mode`	real	brine convection aspect ratio	1
`dSdt_slow_mode`	real	drainage strength parameter	-1.5x10 $^{-7}$ m/s/K
`phi_c_slow_mode`	$0 < \phi_c < 1$	critical liquid fraction	0.05
`phi_i_mushy`	$0 < \phi_i < 1$	solid fraction at lower boundary	0.85
dynamics_nml			
		Dynamics	
`kstrength`	`0`	ice strength formulation [17]	1
	`1`	ice strength formulation [38]	
`krdg_partic`	`0`	old ridging participation function	1
	`1`	new ridging participation function	
`krdg_redist`	`0`	old ridging redistribution function	1
	`1`	new ridging redistribution function	
`mu_rdg`	real	e-folding scale of ridged ice	
`Cf`	real	ratio of ridging work to PE change in ridging	17.
shortwave_nml			
		Shortwave	
`shortwave`	`ccsm3`	NCAR CCSM3 distribution method	'dEdd'
	`dEdd`	Delta-Eddington method	
`albedo_type`	`ccsm3`	NCAR CCSM3 albedos	'ccsm3'
	`constant`	four constant albedos	
`albicev`	$0 < \alpha < 1$	visible ice albedo for thicker ice	
`albicei`	$0 < \alpha < 1$	near infrared ice albedo for thicker ice	
`albsnowv`	$0 < \alpha < 1$	visible, cold snow albedo	
`albsnowi`	$0 < \alpha < 1$	near infrared, cold snow albedo	
`ahmax`	real	albedo is constant above this thickness	0.3 m
`R_ice`	real	tuning parameter for sea ice albedo from Delta-Eddington shortwave	
`R_pnd`	real	… for ponded sea ice albedo …	
`R_snw`	real	… for snow (broadband albedo) …	
`dT_mlt`	real	Δ temperature per Δ snow grain radius	
`rsnw_mlt`	real	maximum melting snow grain radius	
`kalg`	real	absorption coefficient for algae	
ponds_nml			
		Melt Ponds	
`hp1`	real	critical ice lid thickness for topo ponds	0.01 m
`hs0`	real	snow depth of transition to bare sea ice	0.03 m
`hs1`	real	snow depth of transition to pond ice	0.03 m
`dpscale`	real	time scale for flushing in permeable ice	1×10^{-3}
`frzpnd`	`hlid`	Stefan refreezing with pond ice thickness	'hlid'
	`cesm`	CESM refreezing empirical formula	
`rfracmin`	$0 \le r_{min} \le 1$	minimum melt water added to ponds	0.15
`rfracmax`	$0 \le r_{max} \le 1$	maximum melt water added to ponds	1.0

Continued on next page

Table 1.9 – continued from previous page

variable	options/format	description	recommended value
`pndaspect`	real	aspect ratio of pond changes (depth:area)	0.8
zbgc_nml			
		Biogeochemistry	
`tr_brine`	true/false	brine height tracer	
`skl_bgc`	true/false	biogeochemistry	
`bgc_flux_type`	`Jin2006`	ice–ocean flux velocity of [23]	
	`constant`	constant ice–ocean flux velocity	
`restore_bgc`	true/false	restore nitrate/silicate to data	
`sil_data_type`	`default`	default forcing value for silicate	
	`clim`	silicate forcing from ocean climatology [15]	
`nit_data_type`	`default`	default forcing value for nitrate	
	`clim`	nitrate forcing from ocean climatology [15]	
	`sss`	nitrate forcing equals salinity	
`tr_bgc_C_sk`	true/false	algal carbon tracer	
`tr_bgc_chl_sk`	true/false	algal chlorophyll tracer	
`tr_bgc_Am_sk`	true/false	ammonium tracer	
`tr_bgc_Sil_sk`	true/false	silicate tracer	
`tr_bgc_DMSPp_sk`	true/false	particulate DMSP tracer	
`tr_bgc_DMSPd_sk`	true/false	dissolved DMSP tracer	
`tr_bgc_DMS_sk`	true/false	DMS tracer	
`phi_snow`	real	snow porosity for brine height tracer	
forcing_nml			
		Forcing	
`formdrag`	true/false	calculate form drag	
`atmbndy`	`default`	stability-based boundary layer	'default'
	`constant`	bulk transfer coefficients	
`fyear_init`	yyyy	first year of atmospheric forcing data	
`ycycle`	integer	number of years in forcing data cycle	
`atm_data_type`	`default`	constant values defined in the code	
	`clim`	monthly climatology	
	`CFS`	CFS model output	
	`ISPOL`	ISPOL experiment data	
`data_dir`	path/	path to forcing data directory	
`calc_strair`	true	calculate wind stress and speed	
	false	read wind stress and speed from files	
`highfreq`	true/false	high-frequency atmo coupling	
`natmiter`	integer	number of atmo boundary layer iterations	
`calc_Tsfc`	true/false	calculate surface temperature	`.true.`
`precip_units`	`mks`	liquid precipitation data units	
	`mm_per_month`		
	`mm_per_sec`	(same as MKS units)	
`tfrz_option`	`minus1p8`	constant ocean freezing temperature ($-1.8°C$)	
	`linear_salt`	linear function of salinity (ktherm=1)	
	`mushy`	matches mushy-layer thermo (ktherm=2)	
`ustar_min`	real	minimum value of ocean friction velocity	0.0005 m/s

Continued on next page

Table 1.9 – continued from previous page

variable	options/format	description	recommended value
`fbot_xfer_type`	`constant`	constant ocean heat transfer coefficient	
	`Cdn_ocn`	variable ocean heat transfer coefficient	
`update_ocn_f`	true	include frazil water/salt fluxes in ocn fluxes	
	false	do not include (when coupling with POP)	
`l_mpond_fresh`	true	retain (topo) pond water until ponds drain	
	false	release (topo) pond water immediately to ocean	
`oceanmixed_ice`	true/false	active ocean mixed layer calculation	`.true.` (if uncoupled)
`ocn_data_type`	`default`	constant values defined in the code	
	`ISPOL`	ISPOL experiment data	
`bgc_data_type`	`default`	constant values defined in the code	
	`ISPOL`	ISPOL experiment data	
`oceanmixed_file`	filename	data file containing ocean forcing data	
`restore_ocn`	true/false	restore sst to data	
`trestore`	integer	sst restoring time scale (days)	

1.3.5 Adding things

We require that any changes made to the code be implemented in such a way that they can be "turned off" through namelist flags. In most cases, code run with such changes should be bit-for-bit identical with the unmodified code. Occasionally, non-bit-for-bit changes are necessary, e.g. associated with an unavoidable change in the order of operations. In these cases, changes should be made in stages to isolate the non-bit-for-bit changes, so that those that should be bit-for-bit can be tested separately.

Tracers

Tracers added to Icepack will also require extensive modifications to the host sea ice model, including initialization on the horizontal grid, namelist flags and restart capabilities. Modifications to the Icepack driver should reflect the modifications needed in the host model but are not expected to match completely. We recommend that the logical namelist variable `tr_[tracer]` be used for all calls involving the new tracer outside of **ice_[tracer].F90**, in case other users do not want to use that tracer.

A number of optional tracers are available in the code, including ice age, first-year ice area, melt pond area and volume, brine height, aerosols, and level ice area and volume (from which ridged ice quantities are derived). Salinity, enthalpies, age, aerosols, level-ice volume, brine height and most melt pond quantities are volume-weighted tracers, while first-year area, pond area, level-ice area and all of the biogeochemistry tracers in this release are area-weighted tracers. In the absence of sources and sinks, the total mass of a volume-weighted tracer such as aerosol (kg) is conserved under transport in horizontal and thickness space (the mass in a given grid cell will change), whereas the aerosol concentration (kg/m) is unchanged following the motion, and in particular, the concentration is unchanged when there is surface or basal melting. The proper units for a volume-weighted mass tracer in the tracer array are kg/m.

In several places in the code, tracer computations must be performed on the conserved "tracer volume" rather than the tracer itself; for example, the conserved quantity is $h_{pnd}a_{pnd}a_{lvl}a_i$, not h_{pnd}. Conserved quantities are thus computed according to the tracer dependencies, and code must be included to account for new dependencies (e.g., a_{lvl} and a_{pnd} in **ice_itd.F90** and **ice_mechred.F90**).

To add a tracer, follow these steps using one of the existing tracers as a pattern.

1. **icedrv_domain_size.F90**: increase `max_ntrcr` (can also add option to **icepack.settings** and **icepack.build**)
2. **icedrv_state.F90**: declare `nt_[tracer]` and `tr_[tracer]`
3. **icepack_[tracer].F90**: create initialization, physics routines
4. **ice_drv_init.F90**: (some of this may be done in **ice_[tracer].F90** instead)
   - add new module and `tr_[tracer]` to list of used modules and variables
   - add logical namelist variable `tr_[tracer]`
   - initialize namelist variable
   - print namelist variable to diagnostic output file
   - increment number of tracers in use based on namelist input (`ntrcr`)
   - define tracer types (`trcr_depend` = 0 for ice area tracers, 1 for ice volume, 2 for snow volume, 2+``nt_``[tracer] for dependence on other tracers)
5. **icepack_itd.F90**, **icepack_mechred.F90**: Account for new dependencies if needed.
6. **icedrv_InitMod.F90**: initialize tracer (includes reading restart file)
7. **icedrv_RunMod.F90**, **icedrv_step_mod.F90**:
   - call routine to write tracer restart data
   - call physics routines in **icepack_[tracer].F90** (often called from **icedrv_step_mod.F90**)
8. **icedrv_restart.F90**: define restart variables
9. **icepack_in**: add namelist variables to *tracer_nml* and *icefields_nml*
10. If strict conservation is necessary, add diagnostics as noted for topo ponds in Section *Melt ponds*.

1.3.6 Troubleshooting

Check the FAQ: https://github.com/CICE-Consortium/Icepack/wiki

Initial setup

If there are problems, you can manually edit the env, Macros, and **icepack.run** files in the case directory until things are working properly. Then you can copy the env and Macros files back to **configuration/scripts/machines**.

- Changes made directly in the run directory, e.g. to the namelist file, will be overwritten if scripts in the case directory are run again later.
- If changes are needed in the **icepack.run.setup.csh** script, it must be manually modified.

Restarts

- Manual restart tests require the path to the restart file be included in `ice_in` in the namelist file.
- Ensure that `kcatbound` is the same as that used to create the restart file. Other configuration parameters, such as `NICELYR`, must also be consistent between runs.

Underflows

- Tests using a debug flag that traps underflows will fail unless a "flush-to-zero" flag is set in the Macros file. This is due to very small exponential values in the delta-Eddington radiation scheme.

Debugging hints

CHECK write utility in column physics interface, for checking parameter values

A printing utility is available in the driver that can be helpful when debugging the code. Not all of these will work everywhere in the code, due to possible conflicts in module dependencies.

***debug_icepack* (configuration/driver/ice_diagnostics.F90)** A wrapper for *print_state* that is easily called from numerous points during initialization and the timestepping loop

***print_state* (configuration/driver/ice_diagnostics.F90)** Print the ice state and forcing fields for a given grid cell.

Known bugs

- With the old CCSM radiative scheme (`shortwave` = 'default' or 'ccsm3'), a sizable fraction (more than 10%) of the total shortwave radiation is absorbed at the surface but should be penetrating into the ice interior instead. This is due to use of the aggregated, effective albedo rather than the bare ice albedo when `snowpatch` < 1.

Interpretation of albedos

The snow-and-ice albedo, `albsni`, and diagnostic albedos `albice`, `albsno`, and `albpnd` are merged over categories but not scaled (divided) by the total ice area. (This is a change from CICE v4.1 for `albsni`.) The latter three history variables represent completely bare or completely snow- or melt-pond-covered ice; that is, they do not take into account the snow or melt pond fraction (`albsni` does, as does the code itself during thermodyamic computations). This is to facilitate comparison with typical values in measurements or other albedo parameterizations. The melt pond albedo `albpnd` is only computed for the Delta-Eddington shortwave case.

With the Delta-Eddington parameterization, the albedo depends on the cosine of the zenith angle ($\cos\varphi$, `coszen`) and is zero if the sun is below the horizon ($\cos\varphi < 0$). Therefore time-averaged albedo fields would be low if a diurnal solar cycle is used, because zero values would be included in the average for half of each 24-hour period. To rectify this, a separate counter is used for the averaging that is incremented only when $\cos\varphi > 0$. The albedos will still be zero in the dark, polar winter hemisphere.

Proliferating subprocess parameterizations

With the addition of several alternative parameterizations for sea ice processes, a number of subprocesses now appear in multiple parts of the code with differing descriptions. For instance, sea ice porosity and permeability, along with associated flushing and flooding, are calculated separately for mushy thermodynamics, topo and level-ice melt ponds, and for the brine height tracer, each employing its own equations. Likewise, the BL99 and mushy thermodynamics compute freeboard and snow–ice formation differently, and the topo and level-ice melt pond schemes both allow fresh ice to grow atop melt ponds, using slightly different formulations for Stefan freezing. These various process parameterizations will be compared and their subprocess descriptions possibly unified in the future.

1.4 Developers Guide

The Icepack model consists of three different parts, the column physics code, the icepack driver, and the scripts. Development of each of these pieces will be described below separately.

1.4.1 Icepack Column Physics

File List

The column physics source code contains the following files

columnphysics/ the column physics code

- **icepack_aerosol.F90** handles most work associated with the aerosol tracers
- **icepack_age.F90** handles most work associated with the age tracer
- **icepack_algae.F90** biogeochemistry
- **icepack_atmo.F90** stability-based parameterization for calculation of turbulent ice–atmosphere fluxes
- **icepack_brine.F90** evolves the brine height tracer
- **icepack_constants.F90** physical and numerical constants required for column package
- **icepack_firstyear.F90** handles most work associated with the first-year ice area tracer
- **icepack_flux.F90** fluxes needed/produced by the model
- **icepack_intfc.F90** interface routines for linking Icepack with a host sea ice model
- **icepack_itd.F90** utilities for managing ice thickness distribution
- **icepack_kinds.F90** basic definitions of reals, integers, etc.
- **icepack_mechred.F90** mechanical redistribution (ridging)
- **icepack_meltpond_cesm.F90** CESM melt pond parameterization
- **icepack_meltpond_lvl.F90** level-ice melt pond parameterization
- **icepack_meltpond_topo.F90** topo melt pond parameterization
- **icepack_mushy_physics.F90** physics routines for mushy thermodynamics
- **icepack_ocean.F90** mixed layer ocean model
- **icepack_orbital.F90** orbital parameters for Delta-Eddington shortwave parameterization
- **icepack_parameters.F90** basic model parameters
- **icepack_shortwave.F90** shortwave and albedo parameterizations
- **icepack_therm_0layer.F90** zero-layer thermodynamics of *[40]*
- **icepack_therm_bl99.F90** multilayer thermodynamics of *[6]*
- **icepack_therm_itd.F90** thermodynamic changes mostly related to ice thickness distribution
- **icepack_therm_mushy.F90** mushy-theory thermodynamics of *[47]*
- **icepack_therm_shared.F90** code shared by all thermodynamics parameterizations
- **icepack_therm_vertical.F90** vertical growth rates and fluxes
- **icepack_tracers.F90** tracer information
- **icepack_warnings.F90** utilities for writing warning and error messages
- **icepack_zbgc.F90** driver for ice biogeochemistry and brine tracer motion
- **icepack_zbgc_shared.F90** parameters and shared code for biogeochemistry and brine height
- **icepack_zsalinity.F90** vertical salinity parameterization of *[22]*

Coding Standard

The column physics is a library that solves the sea ice column physics on a timestep by timestep and gridpoint by gridpoint basis. It consists of Fortran routines with input and output arguments. The model state is saved in the host model. There is no communication between gridcells so the underlying implementation supports no parallelization. It however can be called in parallel by a driver that is running on multiple pes with a decomposed grid.

The column physics does not have a time manager. Calendaring is expected to be dealt with by the host model. The column physics does not read any forcing data, that is passed into the column physics thru interfaces. In fact, there are no direct IO capabilities in the column physics. That is to say, the column physics does not open files to read or write. The column physics is able to write data via several different routines that specifically have a fortran unit number as an input argument. In addition, there is a warning package (see section *Calling Sequence*) that provides the column package with the ability to store log output. That output can be queried by the host model or it can be written directly via a specific routine. The warning package also provides access to an abort flag that the host model can query after each call to check for successful completion of the column physics package.

All column physics public interfaces and public data are defined in the **icepack_intfc.F90** file. Internal column physics settings should all be accessible thru interfaces. The internal constants, parameters, and tracer settings have init (set), query (get), and write interfaces that provides access to internal column physics settings. The host model should not have to use "use" statements to access any of the column physics data outside of what is provided thru the icepack_intfc module. The public column physics interfaces use optional arguments where it makes sense and there is an ongoing effort to make more of the interfaces support keyword=value arguments for clarity and backwards compatibility.

Using Icepack

In this section, the various public icepack interfaces will be defined and how to use them will be described.

Interfaces

Column physics data and subroutines are made public thru the **icepack_intfc.F90** file. That file contains the entire list of data and subroutines needed to initialize, setup, and run the column physics package. That file points to other modules within the column physics where the interfaces are located.

Within **icepack_intfc.F90**, internal icepack kinds are defined via the icepack_kinds module:

```
use icepack_kinds, only: icepack_char_len  => char_len
use icepack_kinds, only: icepack_char_len_long  => char_len_long
use icepack_kinds, only: icepack_log_kind  => log_kind
use icepack_kinds, only: icepack_int_kind  => int_kind
use icepack_kinds, only: icepack_real_kind => real_kind
use icepack_kinds, only: icepack_dbl_kind  => dbl_kind
use icepack_kinds, only: icepack_r16_kind  => r16_kind
```

icepack_tracers defines a handful of parameters constants that provide information about maximum array sizes for static dimensioning:

```
use icepack_tracers,   only: icepack_max_nbtrcr => max_nbtrcr
use icepack_tracers,   only: icepack_max_algae  => max_algae
use icepack_tracers,   only: icepack_max_dic    => max_dic
use icepack_tracers,   only: icepack_max_doc    => max_doc
use icepack_tracers,   only: icepack_max_don    => max_don
use icepack_tracers,   only: icepack_max_fe     => max_fe
use icepack_tracers,   only: icepack_max_aero   => max_aero
use icepack_tracers,   only: icepack_nmodal1    => nmodal1
use icepack_tracers,   only: icepack_nmodal2    => nmodal2
use icepack_constants, only: icepack_nspint     => nspint
```

icepack_constants provides a list of static parameter constants:

```
use icepack_constants, only: c0 => c0
```

icepack_constants provides init, query, write, and recompute methods to define constant values. These constants have defaults that the caller can query or reset:

```
use icepack_constants, only: icepack_init_constants
use icepack_constants, only: icepack_query_constants
use icepack_constants, only: icepack_write_constants
use icepack_constants, only: icepack_recompute_constants
```

icepack_parameters provides init, query, and write methods to define model parameters. These parameters have defaults that the caller can query or reset:

```
use icepack_parameters, only: icepack_init_parameters
use icepack_parameters, only: icepack_query_parameters
use icepack_parameters, only: icepack_write_parameters
```

icepack_tracers provides init, query, and write methods to define various tracer sizes, flags, indices, and numbers. The tracers have some defaults that the caller can query or reset:

```
use icepack_tracers, only: icepack_compute_tracers
use icepack_tracers, only: icepack_query_tracer_sizes
use icepack_tracers, only: icepack_write_tracer_sizes
use icepack_tracers, only: icepack_init_tracer_flags
use icepack_tracers, only: icepack_query_tracer_flags
use icepack_tracers, only: icepack_write_tracer_flags
use icepack_tracers, only: icepack_init_tracer_indices
use icepack_tracers, only: icepack_query_tracer_indices
use icepack_tracers, only: icepack_write_tracer_indices
use icepack_tracers, only: icepack_init_tracer_numbers
use icepack_tracers, only: icepack_query_tracer_numbers
use icepack_tracers, only: icepack_write_tracer_numbers
```

icepack_itd provides three public interfaces to compute the ice thickness distribution:

```
use icepack_itd, only: icepack_init_itd
use icepack_itd, only: icepack_init_itd_hist
use icepack_itd, only: icepack_aggregate
```

icepack_mechred contains two public interfaces to compute ridging and ice strength:

```
use icepack_mechred, only: icepack_step_ridge
use icepack_mechred, only: icepack_ice_strength
```

icepack_shortwave provides a routine to initialize the radiation computation and an routine to update the radiation computation:

```
use icepack_shortwave, only: icepack_prep_radiation
use icepack_shortwave, only: icepack_step_radiation
```

icepack_brine address brine and zsalinity computations:

```
use icepack_brine, only: icepack_init_hbrine
use icepack_brine, only: icepack_init_zsalinity
```

icepack_zbgc contains several public interfaces to support initialization and computation for the skeletal layer bgc and zbgc options:

```
use icepack_zbgc , only: icepack_init_bgc
use icepack_zbgc , only: icepack_init_zbgc
use icepack_zbgc , only: icepack_biogeochemistry
use icepack_zbgc , only: icepack_init_OceanConcArray
use icepack_zbgc , only: icepack_init_ocean_conc
```

There are a couple of routines to support computation of an atmosphere and ocean interaction:

```
use icepack_atmo , only: icepack_atm_boundary
use icepack_ocean, only: icepack_ocn_mixed_layer
```

icepack_step_therm1 and icepack_step_therm2 compute the ice thermodynamics in two steps:

```
use icepack_therm_vertical, only: icepack_step_therm1
use icepack_therm_itd     , only: icepack_step_therm2
```

icepack_therm_shared provides several methods to compute different internal terms:

```
use icepack_therm_shared  , only: icepack_ice_temperature
use icepack_therm_shared  , only: icepack_snow_temperature
use icepack_therm_shared  , only: icepack_liquidus_temperature
use icepack_therm_shared  , only: icepack_sea_freezing_temperature
use icepack_therm_shared  , only: icepack_enthalpy_snow
use icepack_therm_shared  , only: icepack_init_thermo
use icepack_therm_shared  , only: icepack_init_trcr
```

icepack_orbital provides a routine to set orbital parameters needed for some albedo computations:

```
use icepack_orbital , only: icepack_init_orbit
```

icepack_warnings provides several methods for getting, writing, and clearing messages. There is also a function that returns a logical flag indicating whether the column physics has aborted:

```
use icepack_warnings, only: icepack_warnings_clear
use icepack_warnings, only: icepack_warnings_getall
use icepack_warnings, only: icepack_warnings_print
use icepack_warnings, only: icepack_warnings_flush
use icepack_warnings, only: icepack_warnings_aborted
```

icepack_configure is a standalone icepack method that should always be called first:

```
public :: icepack_configure
```

Calling Sequence

The calling sequence required to setup and run the column physics is generally described below. Several steps may be needed to be taken by the host between icepack calls in order to support the icepack interfaces. The icepack driver and the CICE model provide working examples of how to do this in practice. The sample below does not include bgc.

start driver

- call *icepack_configure*

initialize driver and read in driver namelist

- call *icepack_init_constants*
- call *icepack_init_parameters*
- call *icepack_init_tracers_*
- call *icepack_init_trcr*
- call *icepack_init_thermo*

- call *icepack_init_itd*
- call *icepack_init_itd_hist*
- call *icepack_step_radiation*
- call *icepack_init_zsalinity*
- call *icepack_init_hbrine*
- call *icepack_aggregate*

loop over timesteps loop over gridcells

- call *icepack_prep_radiation*
- call *icepack_step_therm1*
- call *icepack_step_therm2*
- call *icepack_aggregate*
- call *icepack_step_ridge*
- call *icepack_step_radiation*
- call *icepack_atm_boundary*
- call *icepack_ocn_mixed_layer*

end loop over gridcells end loop over timesteps

end driver

The Warning Package

Icepack has no IO capabilities. It does not have direct knowledge of any input or output files. However, it can write output thru specific interfaces that pass in a fortran file unit number. There are several methods in icepack that support writing data to a file this way including the various *icepack_write_* interfaces.

Separately, the icepack warning package is where icepack stores internal output and error messages not directly set in the various write routines. The warning package also contains an *icepack_warnings_aborted* function that will be set to true if icepack detects an abort. In that case, icepack will return to the driver. As a general rule, after each call to icepack, the driver should call:

```
call icepack_warnings_flush(nu_diag)
if (icepack_warnings_aborted()) call icedrv_system_abort(string=subname, &
    file=__FILE__, line=__LINE__)
```

to flush (print and clear) the icepack warning buffer and to check whether icepack aborted. If icepack aborts, it's actually up to the driver to cleanly shut the model down.

Alternatively, *icepack_warnings_getall* provides the saved icepack messages to the driver via an array of strings in the argument list. This allows the driver to reformat that output as needed. *icepack_warnings_print* writes out the messages but does not clear them, and *icepack_warnings_clear* zeros out the icepack warning messages.

1.4.2 Driver Implementation

The icepack driver is Fortran source code and exists to test the column physics in a stand-alone mode for some simple column configurations.

File List

The icepack driver consists of the following files

configuration/driver/ driver for testing Icepack in stand-alone mode

- **icedrv_MAIN.F90** main program
- **icedrv_InitMod.F90** routines for initializing a run
- **icedrv_RunMod.F90** main driver routines for time stepping
- **icedrv_arrays_column.F90** essential arrays to describe the state of the ice
- **icedrv_calendar.F90** keeps track of what time it is
- **icedrv_constants.F90** physical and numerical constants and parameters
- **icedrv_diagnostics.F90** miscellaneous diagnostic and debugging routines
- **icedrv_diagnostics_bgc.F90** diagnostic routines for biogeochemistry
- **icedrv_domain_size.F90** domain sizes
- **icedrv_flux.F90** fluxes needed/produced by the model
- **icedrv_forcing.F90** routines to read and interpolate forcing data for stand-alone model runs
- **icedrv_forcing_bgc.F90** routines to read and interpolate forcing data for bgc stand-alone model runs
- **icedrv_init.F90** general initialization routines
- **icedrv_init_column.F90** initialization routines specific to the column physics
- **icedrv_restart.F90** driver for reading/writing restart files
- **icedrv_restart_column.F90** (CHECK: RENAME bgc) restart routines specific to the column physics
- **icedrv_restart_shared.F90** code shared by all restart options
- **icedrv_state.F90** essential arrays to describe the state of the ice
- **icedrv_step.F90** routines for time stepping the major code components
- **icedrv_system.F90** overall system management calls

Overview

The icepack driver exists to test the column physics. At the present time, it is hardwired to run 4 different gridcells on one processor with the same forcing used for all gridcells. There is no MPI and no threading built into the icepack driver. There is limited IO capabilities, no history files, and no netcdf restart files. The model generally runs very quickly.

There are a few different forcings available.

1.4.3 Scripts Implementation

The scripts are the third part of the icepack package. They support setting up cases, building, and running the icepack stand-alone model.

File List

The directory structure under configure/scripts is as follows.

configuration/scripts/

- **Makefile** primary makefile

icepack.batch.csh creates batch scripts for particular machines
icepack.build compiles the code
icepack.launch.csh creates script logic that runs the executable
icepack.run.setup.csh sets up the run scripts
icepack.run.suite.csh sets up the test suite
icepack.settings defines environment, model configuration and run settings
icepack.test.setup.csh creates configurations for testing the model
icepack_decomp.csh defines the grid size
icepack_in namelist input data
machines/ machine specific files to set env and Macros
makdep.c determines module dependencies
options/ other namelist configurations available from the icepack.create.case command line
parse_namelist.sh replaces namelist with command-line configuration
parse_namelist_from_settings.sh replaces namelist with values from icepack.settings
parse_settings.sh replaces settings with command-line configuration
tests/ scripts for configuring and running basic tests

:: _dev_strategy:

Strategy

The icepack scripts are implemented such that everything is resolved after **icepack.create.case** is called. This is done by both copying specific files into the case directory and running scripts as part of the **icepack.create.case** command line to setup various files.

icepack.create.case drives the case setup. It is written in csh. All supporting scripts are relatively simple csh or sh scripts.

The file **icepack.settings** specifies a set of env defaults for the case. The file **icepack_in** defines the namelist input for the icepack driver.

:: _dev_options:

Preset Case Options

`icepack.create.case -s` option allows the user to choose some predetermined icepack settings and namelist. Those options are defined in **configurations/scripts/options/** and the files are prefixed by either set_env, set_nml, or test_nml. When **icepack.create.case** is executed, the appropriate files are read from **configurations/scripts/options/** and the **icepack.settings** and/or **icepack_in** files are updated in the case directory based on the values in those files.

The filename suffix determines the name of the -s option. So, for instance,

```
icepack.create.case -s diag1,debug,bgcISPOL
```

will search for option files with suffixes of diag1, debug, and bgcISPOL and then apply those settings.

parse_namelist.sh, **parse_settings.sh**, and **parse_namelist_from_settings.sh** are the three scripts that modify **icepack_in** and **icepack.settings**.

To add new options, just add new files to the **configurations/scripts/options/** directory with appropriate names and syntax. The set_nml file syntax is the same as namelist syntax and the set_env files are consistent with csh setenv syntax. See other files for examples of the syntax.

:: _dev_machines:

Machines

Machine specific information is contained in **configuration/scripts/machines**. That directory contains a Macros file and an env file for each supported machine. One other files will need to be changed to support a port, that is **configuration/scripts/icepack.batch.csh**. To port to a new machine, see *Porting*.

:: _dev_testing:

Test scripts

Under **configuration/scripts/tests** are several files including the scripts to setup the smoke and restart tests (**test_smoke.script**, **test_restart.script*). A baseline test script (**baseline.script**) is also there to setup the regression and comparison testing. That directory also contains the preset test suites (ie. **base_suite.ts**) and a file that supports post-processing on the model output (**timeseries.csh**).

There is a subdirectory, **configuration/scripts/tests/CTest**, that supports the CTest scripts. These scripts allow test reporting to CDash.

To add a new test, a file associated with that test will need to be added to the **configuration/scripts/tests** directory similar to **test_smoke.script** and **test_restart.script**. In addition, some new options files in **configuration/scripts/options** may need to be added similar to **test_nml.restart1**, **test_nml.restart2**, and **set_nml.restart**.

1.5 Index of primary variables and parameters

This index defines many of the symbols used frequently in the ice model code. Values appearing in this list are fixed or recommended; most namelist parameters are indicated ($E_\circ$) with their default values. For other namelist options, see Section *Table 8*. All quantities in the code are expressed in MKS units (temperatures may take either Celsius or Kelvin units).

1.5.1 Comprehensive Alphabetical Index

Table 1.10: Alphabetical Index

A			
a11,a12	structure tensor components		
a2D	history field accumulations, 2d		
a3Dz	history field accumulations, 3D vertical		
a3Db	history field accumulations, 3D bio grid		
a3Dc	history field accumulations, 3D categories		
a4Di	history field accumulations, 4D categories, vertical ice		
a4Db	history field accumulations, 4D categories, vertical bio grid		
a4Ds	history field accumulations, 4D categories, vertical snow		
a_min	minimum area concentration for computing velocity	0.001	
a_rapid_mode	• brine channel diameter		
advection	• type of advection algorithm used (‘remap’ or ‘upwind’)	remap	
ahmax	• thickness above which ice albedo is constant	0.3m	
aice_extmin	minimum value for ice extent diagnostic	0.15	

Continued on next page

Table 1.10 – continued from previous page

aice_init	concentration of ice at beginning of timestep		
aice0	fractional open water area		
aice(n)	total concentration of ice in grid cell (in category n)		
albedo_type	• type of albedo parameterization ('default' or 'constant')		
albcnt	counter for averaging albedo		
albice	bare ice albedo		
albicei	• near infrared ice albedo for thicker ice		
albicev	• visible ice albedo for thicker ice		
albocn	ocean albedo	0.06	
albpnd	melt pond albedo		
albsno	snow albedo		
albsnowi	• near infrared, cold snow albedo		
albsnowv	• visible, cold snow albedo		
algalN	algal nitrogen concentration	mmol/m^3	
alv(n)dr(f)	albedo: visible (near IR), direct (diffuse)		
alv(n)dr(f)_ai	grid-box-mean value of alv(n)dr(f)		
amm	ammonia/um concentration	mmol/m^3	
ANGLE	for conversions between the POP grid and latitude-longitude grids	radians	
ANGLET	ANGLE converted to T-cells	radians	
aparticn	participation function		
apeff_ai	grid-cell-mean effective pond fraction		
apondn	area concentration of melt ponds		
arlx1i	relaxation constant for dynamics (stress)		
araftn	area fraction of rafted ice		
aredistrn	redistribution function: fraction of new ridge area		
ardgn	fractional area of ridged ice		
aspect_rapid_mode	• brine convection aspect ratio	1	
astar	e-folding scale for participation function	0.05	
atm_data_dir	• directory for atmospheric forcing data		
atm_data_format	• format of atmospheric forcing files		
atm_data_type	• type of atmospheric forcing		
atmbndy	• atmo boundary layer parameterization ('default' or 'constant')		
avail_hist_fields	type for history field data		
awtidf	weighting factor for near-ir, diffuse albedo	0.36218	
awtidr	weighting factor for near-ir, direct albedo	0.00182	
awtvdf	weighting factor for visible, diffuse albedo	0.63282	
awtvdr	weighting factor for visible, direct albedo	0.00318	
B			
bfb_flag	• for bit-for-bit reproducible diagnostics		
bgc_data_dir	• data directory for bgc		
bgc_flux_type	• ice–ocean flux velocity		
bgc_tracer_type	tracer_type for bgc tracers		
bgrid	nondimensional vertical grid points for bio grid		
bignum	a large number	10^{30}	
block	data type for blocks		
block_id	global block number		

Continued on next page

Table 1.10 – continued from previous page

block_size_x(y)	number of cells along x(y) direction of block		
blockGlobalID	global block IDs		
blockLocalID	local block IDs		
blockLocation	processor location of block		
blocks_ice	local block IDs		
bphi	porosity of ice layers on bio grid		
brlx	relaxation constant for dynamics (momentum)		
bTiz	temperature of ice layers on bio grid		
C			
c<n>	real(n)		
calc_strair	• if true, calculate wind stress	T	
calc_Tsfc	• if true, calculate surface temperature	T	
Cdn_atm	atmospheric drag coefficient		
Cdn_ocn	ocean drag coefficient		
Cf	• ratio of ridging work to PE change in ridging	17.	
cgrid	vertical grid points for ice grid (compare bgrid)		
char_len	length of character variable strings	80	
char_len_long	length of longer character variable strings	256	
check_step	time step on which to begin writing debugging data		
check_umax	if true, check for ice speed $>$ umax_stab		
cldf	cloud fraction		
cm_to_m	cm to meters conversion	0.01	
coldice	value for constant albedo parameterization	0.70	
coldsnow	value for constant albedo parameterization	0.81	
conduct	• conductivity parameterization		
congel	basal ice growth	m	
cosw	cosine of the turning angle in water	1.	
coszen	cosine of the zenith angle		
Cp	proportionality constant for potential energy	kg/m^2/s^2	
cp_air	specific heat of air	1005.0 J/kg/K	
cp_ice	specific heat of fresh ice	2106. J/kg/K	
cp_ocn	specific heat of sea water	4218. J/kg/K	
cp_wv	specific heat of water vapor	1.81x10^3 J/kg/K	
cp063	diffuse fresnel reflectivity (above)	0.063	
cp455	diffuse fresnel reflectivity (below)	0.455	
Cs	fraction of shear energy contributing to ridging	0.25	
Cstar	constant in Hibler ice strength formula	20.	
cxm	combination of HTN values		
cxp	combination of HTN values		
cym	combination of HTE values		

Continued on next page

Table 1.10 – continued from previous page

cyp	combination of HTE values		
D			
daice_da	data assimilation concentration increment rate		
daidtd	ice area tendency due to dynamics/transport	1/s	
daidtt	ice area tendency due to thermodynamics	1/s	
dalb_mlt	[see **ice_shortwave.F90**]	-0.075	
dalb_mlti	[see **ice_shortwave.F90**]	-0.100	
dalb_mltv	[see **ice_shortwave.F90**]	-0.150	
darcy_V	Darcy velocity used for brine height tracer		
dardg1(n)dt	rate of fractional area loss by ridging ice (category n)	1/s	
dardg2(n)dt	rate of fractional area gain by new ridges (category n)	1/s	
daymo	number of days in one month		
daycal	day number at end of month		
days_per_year	• number of days in one year	365	
dbl_kind	definition of double precision	selected_real_kind(13)	
dbug	• write extra diagnostics	.false.	
Delta	function of strain rates	1/s	
denom1	combination of constants for stress equation		
depressT	ratio of freezing temperature to salinity of brine	0.054 deg/ppt	
dhbr_bt	change in brine height at the bottom of the column		
dhbr_top	change in brine height at the top of the column		
dhsn	depth difference for snow on sea ice and pond ice		
diag_file	• diagnostic output file (alternative to standard out)		
diag_type	• where diagnostic output is written	stdout	
diagfreq	• how often diagnostic output is written (10 = once per 10 dt)		
distrb	distribution data type		
distrb_info	block distribution information		
distribution_type	• method used to distribute blocks on processors		
distribution_weight	• weighting method used to compute work per block		
divu	strain rate I component, velocity divergence	1/s	
divu_adv	divergence associated with advection	1/s	
dms	dimethyl sulfide concentration	mmol/m^3	
dmsp	dimethyl sulfoniopropionate concentration	mmol/m^3	
dpscale	• time scale for flushing in permeable ice	1×10^{-3}	
dragio	drag coefficient for water on ice	0.00536	
dSdt_slow_mode	• drainage strength parameter		
dsnow	change in snow thickness	m	
dt	• thermodynamics time step	3600. s	
dt_dyn	dynamics/ridging/transport time step		
dT_mlt	• Δ temperature per Δ snow grain radius	1. deg	
dte	subcycling time step for EVP dynamics (Δt_e)	s	
dte2T	dte / 2(damping time scale)		
dtei	1/dte, where dte is the EVP subcycling time step	1/s	
dump_file	• output file for restart dump		

Continued on next page

Table 1.10 – continued from previous page

dumpfreq	• dump frequency for restarts, y, m or d		
dumpfreq_n	• restart output frequency		
dump_last	• if true, write restart on last time step of simulation		
dxhy	combination of HTE values		
dxt	width of T cell (Δx) through the middle	m	
dxu	width of U cell (Δx) through the middle	m	
dyhx	combination of HTN values		
dyn_dt	dynamics and transport time step (Δt_{dyn})	s	
dyt	height of T cell (Δy) through the middle	m	
dyu	height of U cell (Δy) through the middle	m	
dvidtd	ice volume tendency due to dynamics/transport	m/s	
dvidtt	ice volume tendency due to thermodynamics	m/s	
dvirdg(n)dt	ice volume ridging rate (category n)	m/s	
E			
e11, e12, e22	strain rate tensor components		
ecci	yield curve minor/major axis ratio, squared	1/4	
eice(n)	energy of melting of ice per unit area (in category n)	J/m^2	
emissivity	emissivity of snow and ice	0.95	
eps13	a small number	10^{-13}	
eps16	a small number	10^{-16}	
esno(n)	energy of melting of snow per unit area (in category n)	J/m^2	
evap	evaporative water flux	kg/m^2/s	
ew_boundary_type	• type of east-west boundary condition		
eyc	coefficient for calculating the parameter E, 0< eyc <1	0.36	
F			
faero_atm	aerosol deposition rate	kg/m^2/s	
faero_ocn	aerosol flux to the ocean	kg/m^2/s	
fbot_xfer_type	• type of heat transfer coefficient under ice		
fcondtop(n)(_f)	conductive heat flux	W/m^2	
fcor_blk	Coriolis parameter	1/s	
ferrmax	max allowed energy flux error (thermodynamics)	1x 10^{-3} W/m^2	
ffracn	fraction of fsurfn used to melt pond ice		
fhocn	net heat flux to ocean	W/m^2	
fhocn_ai	grid-box-mean net heat flux to ocean (fhocn)	W/m^2	
field_loc_center	field centered on grid cell	1	
field_loc_Eface	field centered on east face	4	
field_loc_NEcorner	field on northeast corner	2	
field_loc_Nface	field centered on north face	3	
field_loc_noupdate	ignore location of field	-1	
field_loc_unknown	unknown location of field	0	
field_loc_Wface	field centered on west face	5	
field_type_angle	angle field type	3	
field_type_noupdate	ignore field type	-1	
field_type_scalar	scalar field type	1	
field_type_unknown	unknown field type	0	
field_type_vector	vector field type	2	
first_ice	flag for initial ice formation		
flat	latent heat flux	W/m^2	

Continued on next page

Table 1.10 – continued from previous page

floediam	effective floe diameter for lateral melt	300. m	
floeshape	floe shape constant for lateral melt	0.66	
flux_bio	all biogeochemistry fluxes passed to ocean		
flux_bio_ai	all biogeochemistry fluxes passed to ocean, grid cell mean		
flw	incoming longwave radiation	W/m^2	
flwout	outgoing longwave radiation	W/m^2	
fm	Coriolis parameter * mass in U cell	kg/s	
formdrag	• calculate form drag		
fpond	fresh water flux to ponds	kg/m^2/s	
fr_resp	bgc respiration fraction	0.05	
frain	rainfall rate	kg/m^2/s	
frazil	frazil ice growth	m	
fresh	fresh water flux to ocean	kg/m^2/s	
fresh_ai	grid-box-mean fresh water flux (fresh)	kg/m^2/s	
frz_onset	day of year that freezing begins		
frzmlt	freezing/melting potential	W/m^2	
frzmlt_init	freezing/melting potential at beginning of time step	W/m^2	
frzmlt_max	maximum magnitude of freezing/melting potential	1000. W/m^2	
frzpnd	• Stefan refreezing of melt ponds	'hlid'	
fsalt	net salt flux to ocean	kg/m^2/s	
fsalt_ai	grid-box-mean salt flux to ocean (fsalt)	kg/m^2/s	
fsens	sensible heat flux	W/m^2	
fsnow	snowfall rate	kg/m^2/s	
fsnowrdg	snow fraction that survives in ridging	0.5	
fsurf(n)(_f)	net surface heat flux excluding fcondtop	W/m^2	
fsw	incoming shortwave radiation	W/m^2	
fswabs	total absorbed shortwave radiation	W/m^2	
fswfac	scaling factor to adjust ice quantities for updated data		
fswint	shortwave absorbed in ice interior	W/m^2	
fswpenl	shortwave penetrating through ice layers	W/m^2	
fswthru	shortwave penetrating to ocean	W/m^2	
fswthru_ai	grid-box-mean shortwave penetrating to ocean (fswthru)	W/m^2	
fyear	current data year		
fyear_final	last data year		
fyear_init	• initial data year		
G			
gravit	gravitational acceleration	9.80616 m/s^2	
grid_file	• input file for grid info		
grid_format	• format of grid files		
grid_type	• 'rectangular', 'displaced_pole', 'column' or 'regional'		
gridcpl_file	• input file for coupling grid info		
grow_net	specific biogeochemistry growth rate per grid cell	s $^{-1}$	

Continued on next page

Table 1.10 – continued from previous page

Gstar	piecewise-linear ridging participation function parameter	0.15	
H			
halo_info	information for updating ghost cells		
heat_capacity	• if true, use salinity-dependent thermodynamics	T	
hfrazilmin	minimum thickness of new frazil ice	0.05 m	
hi_min	minimum ice thickness for thinnest ice category	0.01 m	
hi_ssl	ice surface scattering layer thickness	0.05 m	
hicen	ice thickness in category n	m	
highfreq	• high-frequency atmo coupling	F	
hin_old	ice thickness prior to growth/melt	m	
hin_max	category thickness limits	m	
hist_avg	• if true, write averaged data instead of snapshots	T	
histfreq	• units of history output frequency: y, m, w, d or 1		
histfreq_n	• integer output frequency in histfreq units		
history_dir	• path to history output files		
history_file	• history output file prefix		
hm	land/boundary mask, thickness (T-cell)		
hmix	ocean mixed layer depth	20. m	
hour	hour of the year		
hp0	pond depth at which shortwave transition to bare ice occurs	0.2 m	
hp1	• critical ice lid thickness for topo ponds (dEdd)	0.01 m	
hpmin	minimum melt pond depth (shortwave)	0.005 m	
hpondn	melt pond depth	m	
hs_min	minimum thickness for which T_s is computed	$1.\times10^{-4}$ m	
hs0	• snow depth at which transition to ice occurs (dEdd)	0.03 m	
hs1	• snow depth of transition to pond ice	0.03 m	
hs_ssl	snow surface scattering layer thickness	0.04 m	
Hstar	determines mean thickness of ridged ice	25. m	
HTE	length of eastern edge (Δy) of T-cell	m	
HTN	length of northern edge (Δx) of T-cell	m	
HTS	length of southern edge (Δx) of T-cell	m	
HTW	length of western edge of (Δy) T-cell	m	
I			
i(j)_glob	global domain location for each grid cell		
i0vis	fraction of penetrating visible solar radiation	0.70	
iblkp	block on which to write debugging data		
i(j)block	Cartesian i,j position of block		
ice_hist_field	type for history variables		
ice_ic	• choice of initial conditions		
ice_stdout	unit number for standard output		
ice_stderr	unit number for standard error output		
ice_ref_salinity	reference salinity for ice–ocean exchanges	4. ppt	

Continued on next page

Table 1.10 – continued from previous page

icells	number of grid cells with specified property (for vectorization)		
iceruf	ice surface roughness	$5.\times10^{-4}$ m	
icetmask	ice extent mask (T-cell)		
iceumask	ice extent mask (U-cell)		
idate	the date at the end of the current time step (yyyymmdd)		
idate0	initial date		
ierr	general-use error flag		
igrid	interface points for vertical bio grid		
i(j)hi	last i(j) index of physical domain (local)		
i(j)lo	first i(j) index of physical domain (local)		
incond_dir	• directory to write snapshot of initial condition		
incond_file	• prefix for initial condition file name		
int_kind	definition of an integer	selected_real_kind(6)	
integral_order	polynomial order of quadrature integrals in remapping	3	
ip, jp	local processor coordinates on which to write debugging data		
istep	local step counter for time loop		
istep0	• number of steps taken in previous run	0	
istep1	total number of steps at current time step		
Iswabs	shortwave radiation absorbed in ice layers	W/m^2	
J			
K			
kalg	• absorption coefficient for algae		
kappav	visible extinction coefficient in ice, wavelength$<$700nm	1.4 m^{-1}	
kcatbound	• category boundary formula		
kdyn	• type of dynamics (1 = EVP, 0 = off)	1	
kg_to_g	kg to g conversion factor	1000.	
kice	thermal conductivity of fresh ice ([6])	2.03 W/m/deg	
kitd	• type of itd conversions (0 = delta function, 1 = linear remap)	1	
kmt_file	• input file for land mask info		
krdg_partic	• ridging participation function	1	
krdg_redist	• ridging redistribution function	1	
krgdn	mean ridge thickness per thickness of ridging ice		
kseaice	thermal conductivity of ice for zero-layer thermodynamics	2.0 W/m/deg	
ksno	thermal conductivity of snow	0.30 W/m/deg	
kstrength	• ice stength formulation (1= [38], 0 = [17])	1	
ktherm	• thermodynamic formulation (0 = zero-layer, 1 = [6], 2 = mushy)		
L			
l_brine	flag for brine pocket effects		
l_conservation_check	if true, check conservation when ridging		
l_fixed_area	flag for prescribing remapping fluxes		
l_mpond_fresh	• if true, retain (topo) pond water until ponds drain		
latpnt	• desired latitude of diagnostic points	degrees N	

Continued on next page

Table 1.10 – continued from previous page

latt(u)_bounds	latitude of T(U) grid cell corners	degrees N	
lcdf64	• if true, use 64-bit format		
Lfresh	latent heat of melting of fresh ice = Lsub - Lvap	J/kg	
lhcoef	transfer coefficient for latent heat		
lmask_n(s)	northern (southern) hemisphere mask		
local_id	local address of block in current distribution		
log_kind	definition of a logical variable	kind(.true.)	
lonpnt	• desired longitude of diagnostic points	degrees E	
lont(u)_bounds	longitude of T(U) grid cell corners	degrees E	
Lsub	latent heat of sublimation for fresh water	2.835×10^6 J/kg	
ltripole_grid	flag to signal use of tripole grid		
Lvap	latent heat of vaporization for fresh water	2.501×10^6 J/kg	
M			
m_min	minimum mass for computing velocity	0.01 kg/m^2	
m_to_cm	meters to cm conversion	100.	
m1	constant for lateral melt rate	1.6×10^{-6} m/s deg^{-m2}	
m2	constant for lateral melt rate	1.36	
m2_to_km2	m^2 to km^2 conversion	1×10^{-6}	
maskhalo_bound	• turns on *bound_state* halo masking		
maskhalo_dyn	• turns on dynamics halo masking		
maskhalo_remap	• turns on transport halo masking		
master_task	task ID for the controlling processor		
max_blocks	maximum number of blocks per processor		
max_ntrcr	maximum number of tracers available	5	
maxraft	maximum thickness of ice that rafts	1. m	
mday	day of the month		
meltb	basal ice melt	m	
meltl	lateral ice melt	m	
melts	snow melt	m	
meltt	top ice melt	m	
min_salin	threshold for brine pockets	0.1 ppt	
mlt_onset	day of year that surface melt begins		
month	the month number		
monthp	previous month number		
mps_to_cmpdy	m per s to cm per day conversion	8.64×10^6	
mtask	local processor number that writes debugging data		
mu_rdg	• e-folding scale of ridged ice		
my_task	task ID for the current processor		
N			
n_aero	number of aerosol species		
natmiter	• number of atmo boundary layer iterations	5	
nblocks	number of blocks on current processor		
nblocks_tot	total number of blocks in decomposition		
nblocks_x(y)	total number of blocks in x(y) direction		
nbtrcr	number of biology tracers		

Continued on next page

Table 1.10 – continued from previous page

ncat	number of ice categories	5	
ncat_hist	number of categories written to history		
ndte	• number of subcycles	120	
ndtd	• number of dynamics/advection steps under thermo	1	
new_day	flag for beginning new day		
new_hour	flag for beginning new hour		
new_month	flag for beginning new month		
new_year	flag for beginning new year		
nghost	number of rows of ghost cells surrounding each subdomain	1	
ngroups	number of groups of flux triangles in remapping	5	
nhlat	northern latitude of artificial mask edge	30°S	
nilyr	number of ice layers in each category	4	
nit	nitrate concentration	mmol/m^3	
nit_data_type	• forcing type for nitrate		
nlt_bgc_[chem]	ocean sources and sinks for biogeochemistry		
nml_filename	namelist file name		
nprocs	• total number of processors		
npt	• total number of time steps (dt)		
ns_boundary_type	• type of north-south boundary condition		
nslyr	number of snow layers in each category		
nspint	number of solar spectral intervals		
nstreams	number of history output streams (frequencies)		
nt_<trcr>	tracer index		
ntrace	number of fields being transported		
ntrcr	number of tracers		
nu_diag	unit number for diagnostics output file		
nu_dump	unit number for dump file for restarting		
nu_dump_eap	unit number for EAP dynamics dump file for restarting		
nu_dump_[tracer]	unit number for tracer dump file for restarting		
nu_forcing	unit number for forcing data file		
nu_grid	unit number for grid file		
nu_hdr	unit number for binary history header file		
nu_history	unit number for history file		
nu_kmt	unit number for land mask file		
nu_nml	unit number for namelist input file		
nu_restart	unit number for restart input file		
nu_restart_eap	unit number for EAP dynamics restart input file		
nu_restart_[tracer]	unit number for tracer restart input file		
nu_rst_pointer	unit number for pointer to latest restart file		
num_avail_hist_fields_[shape]	number of history fields of each array shape		
nvar	number of horizontal grid fields written to history		
nvarz	number of category, vertical grid fields written to history		
nx(y)_block	total number of gridpoints on block in x(y) direction		
nx(y)_global	number of physical gridpoints in x(y) direction, global domain		
nyr	year number		
O			

Continued on next page

Table 1.10 – continued from previous page

ocean_bio	concentrations of bgc constituents in the ocean		
oceanmixed_file	• data file containing ocean forcing data		
oceanmixed_ice	• if true, use internal ocean mixed layer		
ocn_data_dir	• directory for ocean forcing data		
ocn_data_format	• format of ocean forcing files		
omega	angular velocity of Earth	7.292×10^{-5} rad/s	
opening	rate of ice opening due to divergence and shear	1/s	
P			
p001	1/1000		
p01	1/100		
p025	1/40		
p027	1/36		
p05	1/20		
p055	1/18		
p1	1/10		
p111	1/9		
p15	15/100		
p166	1/6		
p2	1/5		
p222	2/9		
p25	1/4		
p333	1/3		
p4	2/5		
p5	1/2		
p52083	25/48		
p5625m	-9/16		
p6	3/5		
p666	2/3		
p75	3/4		
phi_c_slow_mode	• critical liquid fraction		
phi_i_mushy	• solid fraction at lower boundary		
phi_sk	skeletal layer porosity		
phi_snow	• snow porosity for brine height tracer		
pi	π		
pi2	2π		
pih	$\pi/2$		
piq	$\pi/4$		
pi(j,b,m)loc	x (y, block, task) location of diagnostic points		
plat	grid latitude of diagnostic points		
plon	grid longitude of diagnostic points		
pndaspect	• aspect ratio of pond changes (depth:area)	0.8	
pointer_file	• input file for restarting		
potT	atmospheric potential temperature	K	
PP_net	total primary productivity per grid cell	mg C/m^2/s	
precip_units	• liquid precipitation data units		
print_global	• if true, print global data	F	
print_points	• if true, print point data	F	
processor_shape	• descriptor for processor aspect ratio		
prs_sig	replacement pressure	N/m	

Continued on next page

Table 1.10 – continued from previous page

Pstar	ice strength parameter	$2.75{\times}10^4$N/m	
puny	a small positive number	$1{\times}10^{-11}$	
Q			
Qa	specific humidity at 10 m	kg/kg	
qdp	deep ocean heat flux	W/m^2	
qqqice	for saturated specific humidity over ice	$1.16378{\times}10^7$kg/m^3	
qqqocn	for saturated specific humidity over ocean	$6.275724{\times}10^6$kg/m^3	
Qref	2m atmospheric reference specific humidity	kg/kg	
R			
R_C2N	algal carbon to nitrate factor	7. mole/mole	
R_gC2molC	mg/mmol carbon	12.01 mg/mole	
R_chl2N	algal chlorophyll to nitrate factor	3. mg/mmol	
R_ice	• parameter for Delta-Eddington ice albedo		
R_pnd	• parameter for Delta-Eddington pond albedo		
R_S2N	algal silicate to nitrate factor	0.03 mole/mole	
R_snw	• parameter for Delta-Eddington snow albedo		
r16_kind	definition of quad precision	selected_real_kind(26)	
Rac_rapid_mode	• critical Rayleigh number	10	
rad_to_deg	degree-radian conversion	$180/\pi$	
radius	earth radius	$6.37{\times}10^6$ m	
rdg_conv	convergence for ridging	1/s	
rdg_shear	shear for ridging	1/s	
real_kind	definition of single precision real	selected_real_kind(6)	
refindx	refractive index of sea ice	1.310	
revp	real(revised_evp)		
restart	• if true, initialize using restart file instead of defaults	T	
restart_age	• if true, read age restart file		
restart_bgc	• if true, read bgc restart file		
restart_dir	• path to restart/dump files		
restart_file	• restart file prefix		
restart_format	• restart file format		
restart_[tracer]	• if true, read tracer restart file		
restart_ext	• if true, read/write halo cells in restart file		
restore_bgc	• if true, restore nitrate/silicate to data		
restore_ice	• if true, restore ice state along lateral boundaries		
restore_sst	• restore sst to data		
revised_evp	• if true, use revised EVP parameters and approach		
rfracmin	• minimum melt water fraction added to ponds	0.15	
rfracmax	• maximum melt water fraction added to ponds	1.0	
rhoa	air density	kg/m^3	
rhofresh	density of fresh water	1000.0 kg/m^3	
rhoi	density of ice	917. kg/m^3	

Continued on next page

Table 1.10 – continued from previous page

rhos	density of snow	330. kg/m^3	
rhosi	average sea ice density (for hbrine tracer)	940. kg/m^3	
rhow	density of seawater	1026. kg/m^3	
rnilyr	real(nlyr)		
rside	fraction of ice that melts laterally		
rsnw_fresh	freshly fallen snow grain radius	100. $\times 10^{-6}$ m	
rsnw_melt	• melting snow grain radius	1000. $\times 10^{-6}$ m	
rsnw_nonmelt	nonmelting snow grain radius	500. $\times 10^{-6}$ m	
rsnw_sig	standard deviation of snow grain radius	250. $\times 10^{-6}$ m	
runid	• identifier for run		
runtype	• type of initialization used		
S			
s11, s12, s22	stress tensor components		
salinz	ice salinity profile	ppt	
saltmax	max salinity, at ice base ([6])	3.2 ppt	
scale_factor	scaling factor for shortwave radiation components		
sec	seconds elasped into idate		
secday	number of seconds in a day	86400.	
shcoef	transfer coefficient for sensible heat		
shear	strain rate II component	1/s	
shlat	southern latitude of artificial mask edge	30°N	
shortwave	• flag for shortwave parameterization ('default' or 'dEdd')		
sig1(2)	principal stress components (diagnostic)		
sil	silicate concentration	mmol/m^3	
sil_data_type	• forcing type for silicate		
sinw	sine of the turning angle in water	0.	
sk_l	skeletal layer thickness	0.03 m	
snoice	snow–ice formation	m	
snowpatch	length scale for parameterizing nonuniform snow coverage	0.02 m	
skl_bgc	• biogeochemistry on/off		

Continued on next page

Table 1.10 – continued from previous page

spval	special value (single precision)	10^{30}	
spval_dbl	special value (double precision)	10^{30}	
ss_tltx(y)	sea surface in the x(y) direction	m/m	
sss	sea surface salinity	ppt	
sss_data_type	• source of surface salinity data		
sst	sea surface temperature	C	
sst_data_type	• source of surface temperature data		
Sswabs	shortwave radiation absorbed in snow layers	W/m^2	
stefan-boltzmann	Stefan-Boltzmann constant	5.67×10^{-8} W/m^{2}K^4	
stop_now	if 1, end program execution		
strairx(y)	stress on ice by air in the x(y)-direction (centered in U cell)	N/m^2	
strairx(y)T	stress on ice by air, x(y)-direction (centered in T cell)	N/m^2	
strax(y)	wind stress components from data	N/m^2	
strength	ice strength (pressure)	N/m	
stress12	internal ice stress, σ_{12}	N/m	
stressm	internal ice stress, $\sigma_{11} - \sigma_{22}$	N/m	
stressp	internal ice stress, $\sigma_{11} + \sigma_{22}$	N/m	
strintx(y)	divergence of internal ice stress, x(y)	N/m^2	
strocnx(y)	ice–ocean stress in the x(y)-direction (U-cell)	N/m^2	
strocnx(y)T	ice–ocean stress, x(y)-dir. (T-cell)	N/m^2	
strtltx(y)	surface stress due to sea surface slope	N/m^2	
swv(n)dr(f)	incoming shortwave radiation, visible (near IR), direct (diffuse)	W/m^2	
T			
Tair	air temperature at 10 m	K	
tarea	area of T-cell	m^2	
tarean	area of northern hemisphere T-cells	m^2	
tarear	1/tarea	1/m^2	
tareas	area of southern hemisphere T-cells	m^2	
tcstr	string identifying T grid for history variables		
tday	absolute day number		
Tf	freezing temperature	C	
Tffresh	freezing temp of fresh ice	273.15 K	
tfrz_option	• form of ocean freezing temperature		
thinS	minimum ice thickness for brine tracer		
time	total elapsed time	s	
time_beg	beginning time for history averages		
time_bounds	beginning and ending time for history averages		
time_end	ending time for history averages		
time_forc	time of last forcing update	s	
Timelt	melting temperature of ice top surface	0. C	
tinyarea	puny * tarea	m^2	
TLAT	latitude of cell center	radians	
TLON	longitude of cell center	radians	
tmask	land/boundary mask, thickness (T-cell)		
tmass	total mass of ice and snow	kg/m^2	

Continued on next page

Table 1.10 – continued from previous page

Tmin	minimum allowed internal temperature	-100. C	
Tmltz	melting temperature profile of ice		
Tocnfrz	temperature of constant freezing point parameterization	-1.8 C	
tr_aero	• if true, use aerosol tracers		
tr_bgc_[tracer]	• if true, use biogeochemistry tracers		
tr_brine	• if true, use brine height tracer		
tr_FY	• if true, use first-year area tracer		
tr_iage	• if true, use ice age tracer		
tr_lvl	• if true, use level ice area and volume tracers		
tr_pond_cesm	• if true, use CESM melt pond scheme		
tr_pond_lvl	• if true, use level-ice melt pond scheme		
tr_pond_topo	• if true, use topo melt pond scheme		
trcr	ice tracers		
trcr_depend	tracer dependency on basic state variables		
Tref	2m atmospheric reference temperature	K	
trestore	• restoring time scale	days	
tripole	if true, block lies along tripole boundary		
tripoleT	if true, tripole boundary is T-fold; if false, U-fold		
Tsf_errmax	max allowed T_{sf} error (thermodynamics)	$5.\times10^{-4}$deg	
Tsfc(n)	temperature of ice/snow top surface (in category n)	C	
Tsmelt	melting temperature of snow top surface	0. C	
TTTice	for saturated specific humidity over ice	5897.8 K	
TTTocn	for saturated specific humidity over ocean	5107.4 K	
U			
uarea	area of U-cell	m ^{2}	
uarear	1/uarea	m $^{-2}$	
uatm	wind velocity in the x direction	m/s	
ULAT	latitude of U-cell centers	radians	
ULON	longitude of U-cell centers	radians	
umask	land/boundary mask, velocity (U-cell)		
umax_stab	ice speed threshold (diagnostics)	1. m/s	
umin	min wind speed for turbulent fluxes	1. m/s	
uocn	ocean current in the x-direction	m/s	
update_ocn_f	• if true, include frazil ice fluxes in ocean flux fields		
use_leap_years	• if true, include leap days		
use_restart_time	• if true, use date from restart file		
ustar_min	• minimum friction velocity under ice		
ucstr	string identifying U grid for history variables		
uvel	x-component of ice velocity	m/s	
uvel_init	x-component of ice velocity at beginning of time step	m/s	
uvm	land/boundary mask, velocity (U-cell)		
V			

Continued on next page

Table 1.10 – continued from previous page

vatm	wind velocity in the y direction	m/s	
vice(n)	volume per unit area of ice (in category n)	m	
vicen_init	ice volume at beginning of timestep	m	
viscosity_dyn	dynamic viscosity of brine	1.79×10^{-3} kg/m/s	
vocn	ocean current in the y-direction	m/s	
vonkar	von Karman constant	0.4	
vraftn	volume of rafted ice	m	
vrdgn	volume of ridged ice	m	
vredistrn	redistribution function: fraction of new ridge volume		
vsno(n)	volume per unit area of snow (in category n)	m	
vvel	y-component of ice velocity	m/s	
vvel_init	y-component of ice velocity at beginning of time step	m/s	
W			
warmice	value for constant albedo parameterization	0.68	
warmsno	value for constant albedo parameterization	0.77	
wind	wind speed	m/s	
write_history	if true, write history now		
write_ic	• if true, write initial conditions		
write_restart	if 1, write restart now		
X			
Y			
ycycle	• number of years in forcing data cycle		
yday	day of the year		
yield_curve	type of yield curve	ellipse	
yieldstress11(12, 22)	yield stress tensor components		
year_init	• the initial year		
Z			
zlvl	atmospheric level height	m	
zref	reference height for stability	10. m	
zTrf	reference height for T_{ref}, Q_{ref}, U_{ref}	2. m	
zvir	gas constant (water vapor)/gas constant (air) - 1	0.606	

1.6 References

References

1.7 Useful tools

- search

BIBLIOGRAPHY

[1] T.L. Amundrud, H. Malling, and R.G. Ingram. Geometrical constraints on the evolution of ridged sea ice. *J. Geophys. Res. Oceans*, 2004. URL: http://dx.doi.org/10.1029/2003JC002251.

[2] K.C. Armour, C.M. Bitz, L. Thompson, and E.C. Hunke. Controls on Arctic sea ice from first-year and multi-year ice survivability. *J. Climate*, 24:2378–2390, 2011. URL: http://dx.doi.org/10.1175/2010JCLI3823.1.

[3] S.P.S. Arya. A drag partition theory for determining the large-scale roughness parameter and wind stress on the Arctic pack ice. *J. Geophys. Res.*, 80:3447–3454, 1975. URL: http://dx.doi.org/10.1029/JC080i024p03447.

[4] A. Assur. Composition of sea ice and its tensile strength. In *Arctic sea ice; conference held at Easton, Maryland, February 24–27, 1958*, volume 598, pages 106–138. Publs. Natl. Res. Coun. Wash., Washington, D.C., 1958.

[5] C.M. Bitz, M.M. Holland, M. Eby, and A.J. Weaver. Simulating the ice-thickness distribution in a coupled climate model. *J. Geophys. Res. Oceans*, 106:2441–2463, 2001. URL: http://dx.doi.org/10.1029/1999JC000113.

[6] C.M. Bitz and W.H. Lipscomb. An energy-conserving thermodynamic sea ice model for climate study. *J. Geophys. Res. Oceans*, 104(C7):15669–15677, 1999. URL: http://dx.doi.org/10.1029/1999JC900100.

[7] B.P. Briegleb and B. Light. *A Delta-Eddington multiple scattering parameterization for solar radiation in the sea ice component of the Community Climate System Model*. NCAR Technical Note NCAR/TN-472+STR, National Center for Atmospheric Research, 2007. URL: https://github.com/CICE-Consortium/CICE/blob/master/doc/PDF/BL_NCAR2007.pdf.

[8] E.E. Ebert, J.L. Schramm, and J.A. Curry. Disposition of solar radiation in sea ice and the upper ocean. *J. Geophys. Res. Oceans*, 100:15965–15975, 1995. URL: http://dx.doi.org/10.1029/95JC01672.

[9] H. Eicken, T.C. Grenfell, D.K. Perovich, J.A Richter-Menge, and K. Frey. Hydraulic controls of summer Arctic pack ice albedo. *J. Geophys. Res. Oceans*, 2004. URL: http://dx.doi.org/10.1029/2003JC001989.

[10] D.L. Feltham, N. Untersteiner, J.S. Wettlaufer, and M.G. Worster. Sea ice is a mushy layer. *Geophys. Res. Lett.*, 2006. URL: http://dx.doi.org/10.1029/2006GL026290.

[11] G.M. Flato and W.D. Hibler. Ridging and strength in modeling the thickness distribution of Arctic sea ice. *J. Geophys. Res. Oceans*, 100:18611–18626, 1995. URL: http://dx.doi.org/10.1029/95JC02091.

[12] D. Flocco and D.L. Feltham. A continuum model of melt pond evolution on Arctic sea ice. *J. Geophys. Res. Oceans*, 2007. URL: http://dx.doi.org/10.1029/2006JC003836.

[13] D. Flocco, D.L. Feltham, and A.K. Turner. Incorporation of a physically based melt pond scheme into the sea ice component of a climate model. *J. Geophys. Res. Oceans*, 2010. URL: http://dx.doi.org/10.1029/2009JC005568.

[14] D. Flocco, D. Schroeder, D.L. Feltham, and E.C. Hunke. Impact of melt ponds on Arctic sea ice simulations from 1990 to 2007. *J. Geophys. Res. Oceans*, 2012. URL: http://dx.doi.org/10.1029/2012JC008195.

[15] H.E. Garcia, R.A. Locarnini, T.P. Boyer, and J.I. Antonov. Nutrients (phosphate, nitrate, silicate). In *World Ocean Atlas 2005*, volume 4. NOAA Atlas NESDIS 64, NOAA, 2006.

[16] K.M. Golden, H. Eicken, A.L. Heaton, J. Miner, D.J. Pringle, and J. Zhu. Thermal evolution of permeability and microstructure in sea ice. *Geophys. Res. Lett.*, 2007. URL: http://dx.doi.org/10.1029/2007GL030447.

[17] W.D. Hibler. A dynamic thermodynamic sea ice model. *J. Phys. Oceanogr.*, 9:817–846, 1979. URL: http://dx.doi.org/10.1175/1520-0485(1979)009\T1\textless{}0815:ADTSIM\T1\textgreater{}2.0.CO;2.

[18] W.D. Hibler. Modeling a variable thickness sea ice cover. *Mon. Wea. Rev.*, 108:1943–1973, 1980. URL: http://dx.doi.org/10.1175/1520-0493(1980)108\T1\textless{}1943:MAVTSI\T1\textgreater{}2.0.CO;2.

[19] M.M. Holland, D.A. Bailey, B.P. Briegleb, B. Light, and E. Hunke. Improved sea ice shortwave radiation physics in CCSM4: The impact of melt ponds and aerosols on Arctic sea ice. *J. Climate*, 25:1413–1430, 2012. URL: http://dx.doi.org/10.1175/JCLI-D-11-00078.1.

[20] E.C. Hunke and C.M. Bitz. Age characteristics in a multidecadal Arctic sea ice simulation. *J. Geophys. Res. Oceans*, 2009. URL: http://dx.doi.org/10.1029/2008JC005186.

[21] E.C. Hunke, D.A. Hebert, and O. Lecomte. Level-ice melt ponds in the Los Alamos Sea Ice Model, CICE. *Ocean Modelling*, 71:26–42, 2013. URL: http://dx.doi.org/10.1016/j.ocemod.2012.11.008.

[22] N. Jeffery, E.C. Hunke, and S.M. Elliott. Modeling the transport of passive tracers in sea ice. *J. Geophys. Res. Oceans*, 116:2156–2202, 2011. URL: https://doi.org/10.1029/2010JC006527.

[23] M. Jin, C. Deal, J. Wang, K.H. Shin, N. Tanaka, T.E. Whiteledge, S.H. Lee, and R.R. Gradinger. Controls of the landfast ice-ocean ecosystem offshore Barrow, Alaska. *Ann. Glaciol.*, 44:63–72, 2006. URL: https://github.com/CICE-Consortium/CICE/blob/master/doc/PDF/JDWSTWLG06.pdf.

[24] R.E. Jordan, E.L. Andreas, and A.P. Makshtas. Heat budget of snow-covered sea ice at North Pole 4. *J. Geophys. Res. Oceans*, 104(C4):7785–7806, 1999. URL: http://dx.doi.org/10.1029/1999JC900011.

[25] B.G. Kauffman and W.G. Large. The CCSM coupler, version 5.0.1. *NCAR Tech. Note*, 2002. URL: https://github.com/CICE-Consortium/CICE/blob/master/doc/PDF/KL_NCAR2002.pdf.

[26] W.H. Lipscomb. *Modeling the Thickness Distribution of Arctic Sea Ice*. Dept. of Atmospheric Sciences University of Washington, Seattle, 1998. PhD thesis. URL: http://hdl.handle.net/1773/10081.

[27] W.H. Lipscomb. Remapping the thickness distribution in sea ice models. *J. Geophys. Res. Oceans*, 106:13989–14000, 2001. URL: http://dx.doi.org/10.1029/2000JC000518.

[28] W.H. Lipscomb, E.C. Hunke, W. Maslowski, and J. Jakacki. Ridging, strength, and stability in high-resolution sea ice models. *J. Geophys. Res. Oceans*, 2007. URL: http://dx.doi.org/10.1029/2005JC003355.

[29] P. Lu, Z. Li, B. Cheng, and M. Lepparanta. A parametrization fo the ice-ocean drag coefficient. *J. Geophys. Res. Oceans*, 2011. URL: http://dx.doi.org/10.1029/2010JC006878.

[30] C. Lüpkes, V.M. Gryanik, J. Hartmann, and E.L. Andreas. A parametrization, based on sea ice morphology, of the neutral atmospheric drag coefficients for weather prediction and climate models. *J. Geophys. Res. Atmos.*, 2012. URL: http://dx.doi.org/10.1029/2012JD017630.

[31] G.A. Maykut. Large-scale heat exchange and ice production in the central Arctic. *J. Geophys. Res. Oceans*, 87:7971–7984, 1982. URL: http://dx.doi.org/10.1029/JC087iC10p07971.

[32] G.A. Maykut and M.G. McPhee. Solar heating of the Arctic mixed layer. *J. Geophys. Res. Oceans*, 100:24691–24703, 1995. URL: http://dx.doi.org/10.1029/95JC02554.

[33] G.A. Maykut and D.K. Perovich. The role of shortwave radiation in the summer decay of a sea ice cover. *J. Geophys. Res. Oceans*, 92:7032–7044, 1987. URL: http://dx.doi.org/10.1029/JC092iC07p07032.

[34] G.A. Maykut and N. Untersteiner. Some results from a time dependent thermodynamic model of sea ice. *J. Geophys. Res.*, 76:1550–1575, 1971. URL: http://dx.doi.org/10.1029/JC076i006p01550.

[35] D. Notz. *Thermodynamic and Fluid-Dynamical Processes in Sea Ice*. University of Cambridge, UK, 2005. PhD thesis. URL: http://ulmss-newton.lib.cam.ac.uk/vwebv/holdingsInfo?bibId=27224.

[36] N. Ono. Specific heat and heat of fusion of sea ice. In H. Oura, editor, *Physics of Snow and Ice*, volume I, pages 599–610. Institute of Low Temperature Science, Hokkaido, Japan, 1967.

[37] D.J. Pringle, H. Eicken, H.J. Trodahl, and L.G.E. Backstrom. Thermal conductivity of landfast Antarctic and Arctic sea ice. *J. Geophys. Res. Oceans*, 2007. URL: http://dx.doi.org/10.1029/2006JC003641.

[38] D.A. Rothrock. The energetics of plastic deformation of pack ice by ridging. *J. Geophys. Res.*, 80:4514–4519, 1975. URL: http://dx.doi.org/10.1029/JC080i033p04514.

[39] W. Schwarzacher. Pack ice studies in the Arctic Ocean. *J. Geophys. Res.*, 64:2357–2367, 1959. URL: http://dx.doi.org/10.1029/JZ064i012p02357.

[40] A.J. Semtner. A Model for the Thermodynamic Growth of Sea Ice in Numerical Investigations of Climate. *J. Phys. Oceanogr.*, 6:379–389, 1976. URL: http://dx.doi.org/10.1175/1520-0485(1976)006\T1\textless{}0379:AMFTTG\T1\textgreater{}2.0.CO;2.

[41] G. Siedler and H. Peters. Physical properties (general) of sea water. In *Landolt-Börnstein: Numerical data and functional relationships in science and technology, New Series V/3a*, pages 233–264. Springer, 1986.

[42] M. Steele. Sea ice melting and floe geometry in a simple ice-ocean model. *J. Geophys. Res. Oceans*, 97:17729–17738, 1992. URL: http://dx.doi.org/10.1029/92JC01755.

[43] P.D. Taylor and D.L. Feltham. A model of melt pond evolution on sea ice. *J. Geophys. Res. Oceans*, 2004. URL: http://dx.doi.org/10.1029/2004JC002361.

[44] A.S. Thorndike, D.A. Rothrock, G.A. Maykut, and R. Colony. The thickness distribution of sea ice. *J. Geophys. Res.*, 80:4501–4513, 1975. URL: http://dx.doi.org/10.1029/JC080i033p04501.

[45] H.J. Trodahl, S.O.F. Wilkinson, M.J. McGuinness, and T.G. Haskeel. Thermal conductivity of sea ice: dependence on temperature and depth. *Geophys. Res. Lett.*, 28:1279–1282, 2001. URL: http://dx.doi.org/10.1029/2000GL012088.

[46] M. Tsamados, D.L. Feltham, D. Schroeder, D. Flocco, S.L. Farrell, N.T. Kurtz, S.W. Laxon, and S. Bacon. Impact of variable atmospheric and oceanic form drag on simulations of Arctic sea ice. *J. Phys. Oceanogr.*, 44:1329–1353, 1999. URL: https://doi.org/10.1175/JPO-D-13-0215.1.

[47] A.K. Turner, E.C. Hunke, and C.M. Bitz. Two modes of sea-ice gravity drainage: a parameterization for large-scale modeling. *J. Geophys. Res. Oceans*, 118:2279–2294, 2013. URL: http://dx.doi.org/10.1002/jgrc.20171.

[48] N. Untersteiner. Calculations of temperature regime and heat budget of sea ice in the Central Arctic. *J. Geophys. Res.*, 69:4755–4766, 1964. URL: http://dx.doi.org/10.1029/JZ069i022p04755.