CICE: the Los Alamos Sea Ice Model
Documentation and Software User's Manual
Version 5.0 DRAFT
LA-CC-06-012 RENEW

Elizabeth C. Hunke, William H. Lipscomb, Adrian K. Turner
T-3 Fluid Dynamics Group, Los Alamos National Laboratory
Los Alamos NM 87545

May 24, 2013

Contents

1 Introduction

The Los Alamos sea ice model (CICE) is the result of an effort to develop a computationally efficient sea ice component for a fully coupled atmosphere-ice-ocean-land global climate model. It was designed to be compatible with the Parallel Ocean Program (POP), an ocean circulation model developed at Los Alamos National Laboratory for use on massively parallel computers [48, 9, 10]. The current version of the model has been enhanced greatly through collaborations with members of the community.

CICE has several interacting components: a thermodynamic model that computes local growth rates of snow and ice due to vertical conductive, radiative and turbulent fluxes, along with snowfall; a model of ice dynamics, which predicts the velocity field of the ice pack based on a model of the material strength of the ice; a transport model that describes advection of the areal concentration, ice volumes and other state variables; and a ridging parameterization that transfers ice among thickness categories based on energetic balances and rates of strain. Additional routines prepare and execute data exchanges with an external "flux coupler," which then passes the data to other climate model components such as POP.

This model release is CICE version TBD, available from http://climate.lanl.gov/Models/CICE/. It updates CICE 4.1, which was released in May 2010:

- Two new explicit melt pond parameterizations (topo and level-ice)
- CESM aerosols
- Gracefully handle the case when an internal layer melts completely
- Gregorian calendar with leap years
- Reduced memory and floating-point operations for tracer calculations
- New history variables for melt ponds and ridging diagnostics
- Read/write extended grid, including ghost cells
- CPP options for categories, layers and tracers
- Corrected bugs, particularly for nonstandard configurations.

Generally speaking, subroutine names are given in *italic* and file names are **boldface** in this document. Symbols used in the code are `typewritten`, while corresponding symbols in this document are in the $math$ font which, granted, looks a lot like italic. A comprehensive index, including a glossary of symbols with many of their values, appears at the end. The organization of this software distribution is described in Section 4.1; most files and subroutines referred to in this documentation are part of the CICE code found in **cice/source/**, unless otherwise noted.

We pronounce the model name as "sea ice," but there has been a small grass-roots movement underway to alter the model name's pronunciation from "sea ice" to what an Italian might say, chē′-chā or "chee-chay." Others choose to say sīs (English, rhymes with "ice"), sēs (French, like "cease"), or shē-ī-so͡o ("Shii-aisu," Japanese).

2 Coupling with other climate model components

The sea ice model exchanges information with the other model components via a flux coupler. CICE has been coupled into numerous climate models with a variety of coupling techniques. This document is oriented primarily toward the CESM Flux Coupler [31] from NCAR, the first major climate model to incorporate CICE. The flux coupler was originally intended to gather state variables from the component models, compute fluxes at the model interfaces, and return these fluxes to the component models for use in the next integration period, maintaining conservation of momentum, heat and fresh water. However, several of these fluxes are now computed in the ice model itself and provided to the flux coupler for distribution to the other components, for two reasons. First, some of the fluxes depend strongly on the state of the ice, and vice versa, implying that an implicit, simultaneous determination of the ice state and the surface fluxes is necessary for consistency and stability. Second, given the various ice types in a single grid cell, it is more efficient for the ice model to determine the net ice characteristics of the grid cell and provide the resulting fluxes, rather than

	Atmosphere		**Ocean**
	Provided by the flux coupler to the sea ice model		
$z_\circ$	Atmosphere level height	F_{frzmlt}	Freezing/melting potential
$\vec{U}_a$	Wind velocity	T_w	Sea surface temperature
Q_a	Specific humidity	S	Sea surface salinity
ρ_a	Air density	$\nabla H_\circ$	Sea surface slope
Θ_a	Air potential temperature	$\vec{U}_w$	Surface ocean currents
T_a	Air temperature		
$F_{sw\downarrow}$	Shortwave radiation (4 bands)		
$F_{L\downarrow}$	Incoming longwave radiation		
F_{rain}	Rainfall rate		
F_{snow}	Snowfall rate		
	Provided by the sea ice model to the flux coupler		
$\vec{\tau}_a$	Wind stress	$F_{sw\Downarrow}$	Penetrating shortwave
F_s	Sensible heat flux	F_{water}	Fresh water flux
F_l	Latent heat flux	F_{hocn}	Net heat flux to ocean
$F_{L\uparrow}$	Outgoing longwave	F_{salt}	Salt flux
F_{evap}	Evaporated water	$\vec{\tau}_w$	Ice-ocean stress
α	Surface albedo (4 bands)		
T_{sfc}	Surface temperature		
a_i	Ice fraction		
T_a^{ref}	2 m reference temperature (diagnostic)		
Q_a^{ref}	2 m reference humidity (diagnostic)		
F_{swabs}	Absorbed shortwave (diagnostic)		

Table 1: Data exchanged between the CESM flux coupler and the sea ice model.

passing several values of the state variables for each cell. These considerations are explained in more detail below.

The fluxes and state variables passed between the sea ice model and the CESM flux coupler are listed in Table 1. By convention, directional fluxes are positive downward.

The ice fraction a_i (`aice`)[1] 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 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. 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.

2.1 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. Wind stress is arguably the primary forcing mechanism for the ice motion, although the ice–ocean stress, Coriolis force, and slope of the ocean surface are also important [51]. 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 Section 3.6, 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.

[1] `Typewritten` equivalents used in the code are described in the index.

Wind stress and transfer coefficients for the turbulent heat fluxes are computed in subroutine *atmo_boundary_layer* following [31]. 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"

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

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:

$$\begin{aligned} u^* &= c_u \left| \vec{U}_a \right| \\ \Theta^* &= c_\theta \left(\Theta_a - T_{sfc} \right) \\ Q^* &= c_q \left(Q_a - Q_{sfc} \right). \end{aligned} \tag{1}$$

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 Section 3.6.2.

The exchange coefficients c_u, c_θ and c_q are initialized as

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

and updated during a short iteration, as they depend upon the turbulent scales. Here, z_{ref} is a reference height of 10 m 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 $\chi = (1 - 16\Upsilon)^{0.25}$ and $\chi \geq 1$, the "integrated flux profiles" for momentum and stability in the unstable ($\Upsilon < 0$) case are given by

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

In a departure from the parameterization used in [31], we use profiles for the stable case following [30],

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

The coefficients are then updated as

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

where $\lambda = \ln\left(z_\circ/z_{ref}\right)$. The first iteration ends with new turbulent scales from equations (1). After five iterations the latent and sensible heat flux coefficients are computed, along with the wind stress:

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

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 [30], 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 Section 3.6.2.

2.2 Ocean

New sea ice forms when the ocean temperature drops below its freezing temperature, $T_f = -\mu S$, where S is the seawater salinity and $\mu = 0.054$ °/ppt is the ratio of the freezing temperature of brine to its salinity. The ocean model performs this calculation; 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.) In general, this ice 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 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 meltwater 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 Section 3.6.2.

Many ice models compute the sea surface slope $\nabla H_\circ$ from geostrophic ocean currents provided by an ocean model or other data source. In our case, the sea surface height $H_\circ$ is a prognostic variable in POP—the flux coupler can provide the surface slope directly, rather than inferring it from the currents. (The option of computing it from the currents is provided in subroutine *evp_prep*.) The sea ice model uses the surface layer currents $\vec{U}_w$ to determine the stress between the ocean and the ice, and subsequently the ice velocity $\vec{u}$. This stress, relative to the ice,

$$\vec{\tau}_w = c_w \rho_w \left|\vec{U}_w - \vec{u}\right| \left[\left(\vec{U}_w - \vec{u}\right)\cos\theta + \hat{k} \times \left(\vec{U}_w - \vec{u}\right)\sin\theta\right]$$

is then passed to the flux coupler (relative to the ocean) for use by the ocean model. Here, θ is the turning angle between geostrophic and surface currents, c_w is the ocean drag coefficient, ρ_w is the density of seawater (`dragw` $= c_w\rho_w$), and $\hat{k}$ is the vertical unit vector. The turning angle is necessary if the top ocean model layers are not able to resolve the Ekman spiral in the boundary layer. If the top layer is sufficiently thin compared to the typical depth of the Ekman spiral, then $\theta = 0$ is a good approximation. Here we assume that the top layer is thin enough.

3 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 [54]:

$$\frac{\partial g}{\partial t} = -\nabla \cdot (g\mathbf{u}) - \frac{\partial}{\partial h}(fg) + \psi, \tag{2}$$

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, 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.

Equation (2) 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 [3, 33].) 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 [36]. The continuous function $g(h)$ is replaced by the discrete variable a_{in}, defined as the fractional area covered by ice in the thickness range (H_{n-1}, H_n). We denote the fractional area of 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. 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 because it was used to create the restart files included with the code distribution. Users may substitute their own preferred boundaries in *init_itd*.

In addition to the fractional ice area, a_{in}, we define the following state variables for each category n:

- v_{in}, the ice volume, equal to the product of a_{in} and the ice thickness h_{in}.
- v_{sn}, the snow volume, equal to the product of a_{in} and the snow thickness h_{sn}.
- e_{ink}, the internal ice energy in layer k, equal to the product of the ice layer volume, v_{in}/N_i, and the ice layer enthalpy, q_{ink}. Here N_i is the total number of ice layers, with a default value $N_i = 4$, and q_{ink} is the negative of the energy needed to melt a unit volume of ice and raise its temperature to 0°C; it is discussed in Section 3.6. (NOTE: In the current code, $e_i < 0$ and $q_i < 0$ with $e_i = v_i q_i$.)
- e_{snk}, the internal snow energy in layer k, equal to the product of the snow layer volume, v_{sn}/N_s, and the snow layer enthalpy, q_{snk}, where N_s is the number of snow layers. (Similarly, $e_s < 0$ in the code.) Earlier versions of CICE had a single snow layer, but multiple layers are now allowed. The default value is $N_s = 1$.

distribution	original	round	WMO		
`kcatbound`	0	1	2		
N_C	5	5	5	6	7
category	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

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

- T_{sfn}, the surface temperature.

Since the fractional area is unitless, the volume variables have units of meters (i.e., m^3 of ice or snow per m^2 of grid cell area), and the energy variables have units of J/m^2.

The three terms on the right-hand side of (2) describe three kinds of sea ice transport: (1) horizontal transport in (x, y) space; (2) transport in thickness space h due to thermodynamic growth and melting; and (3) transport in thickness space h due to ridging and other mechanical processes. We solve the equation by operator splitting in three stages, with two of the three terms on the right set to zero in each stage. We compute horizontal transport using the incremental remapping scheme of [8] as adapted for sea ice by [34]; this scheme is discussed in Section 3.2. Ice is transported in thickness space using the remapping scheme of [33], as described in Section 3.3. The mechanical redistribution scheme, based on [54], [44], [21], [14], and [35], is outlined in Section 3.4. To solve the horizontal transport and ridging equations, we need the ice velocity $\mathbf{u}$, and to compute transport in thickness space, we must know the the ice growth rate f in each thickness category. We use the elastic-viscous-plastic (EVP) ice dynamics scheme of [25], as modified by [7], [23], [26] and [27], to find the velocity, as described in Section 3.5. Finally, we use the thermodynamic model of [4], discussed in Section 3.6, to compute f.

The order in which these computations are performed in the code itself was chosen so that quantities sent to the coupler are consistent with each other and as up-to-date as possible. The Delta-Eddington radiative scheme computes albedo and shortwave components simultaneously, and in order to have the most up-to-date values available for the coupler at the end of the timestep, the order of radiation calculations is shifted. Albedo and shortwave components are computed after the ice state has been modified by both thermodynamics and dynamics, so that they are consistent with the ice area and thickness at the end of the step when sent to the coupler. However, they are computed using the downwelling shortwave from the beginning of the timestep. Rather than recompute the albedo and shortwave components at the beginning of the next timestep using new values of the downwelling shortwave forcing, the shortwave components computed at the end of the last timestep are scaled for the new forcing.

3.1 Tracers

The basic conservation equations for ice area fraction a_{in}, ice volume v_{in}, and snow volume v_{sn} for each thickness category n are

$$\frac{\partial}{\partial t}(a_{in}) + \nabla \cdot (a_{in}\mathbf{u}) = 0, \quad (3)$$

$$\frac{\partial v_{in}}{\partial t} + \nabla \cdot (v_{in}\mathbf{u}) = 0, \quad (4)$$

$$\frac{\partial v_{sn}}{\partial t} + \nabla \cdot (v_{sn}\mathbf{u}) = 0. \quad (5)$$

$$(6)$$

The ice and snow volumes can be written equivalently in terms of tracers, ice thickness h_{in} and snow depth h_{sn}:

$$\frac{\partial h_{in}a_{in}}{\partial t} + \nabla \cdot (h_{in}a_{in}\mathbf{u}) = 0, \quad (7)$$

$$\frac{\partial h_{sn}a_{in}}{\partial t} + \nabla \cdot (h_{sn}a_{in}\mathbf{u}) = 0. \quad (8)$$

$$(9)$$

Although we maintain ice and snow volume instead of the thicknesses as state variables in CICE, the tracer form is used for volume transport (section 3.2). There are many other tracers available, whose values are contained in the `trcrn` array. Their transport equations typically have one of the following three forms

$$\frac{\partial (a_{in}T_n)}{\partial t} + \nabla \cdot (a_{in}T_n\mathbf{u}) = 0, \quad (10)$$

$$\frac{\partial (v_{in}T_n)}{\partial t} + \nabla \cdot (v_{in}T_n\mathbf{u}) = 0, \quad (11)$$

$$\frac{\partial (v_{sn}T_n)}{\partial t} + \nabla \cdot (v_{sn}T_n\mathbf{u}) = 0. \quad (12)$$

Equation (10) describes the transport of surface temperature, whereas (11) and (12) describe the transport of passive tracers such as volume-weighted ice age and snow age. Each tracer field is given an integer index, `trcr_depend`, which has the value 0, 1, or 2 depending on whether the appropriate conservation equation is (10) (11), or (12), respectively. The total number of tracers is $N_{tr} \geq 1$. In the default configuration there are two tracers: surface temperature and volume-weighted ice age. Tracers for melt ponds, level ice area and level ice volume (used to diagnose ridged ice area and volume) are also available. Users may add any number of additional tracers that are transported conservatively provided that `trcr_depend` is defined appropriately. See Section 4.8.3 for guidance on adding tracers.

3.1.1 Tracers that depend on other tracers (e.g., melt ponds)

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 Eq. 103–106). 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

$$\frac{\partial}{\partial t}(a_{pnd}a_i) + \nabla \cdot (a_{pnd}a_i\mathbf{u}) = 0, \quad (13)$$

$$\frac{\partial}{\partial t}(h_{pnd}a_{pnd}a_i) + \nabla \cdot (h_{pnd}a_{pnd}a_i\mathbf{u}) = 0. \quad (14)$$

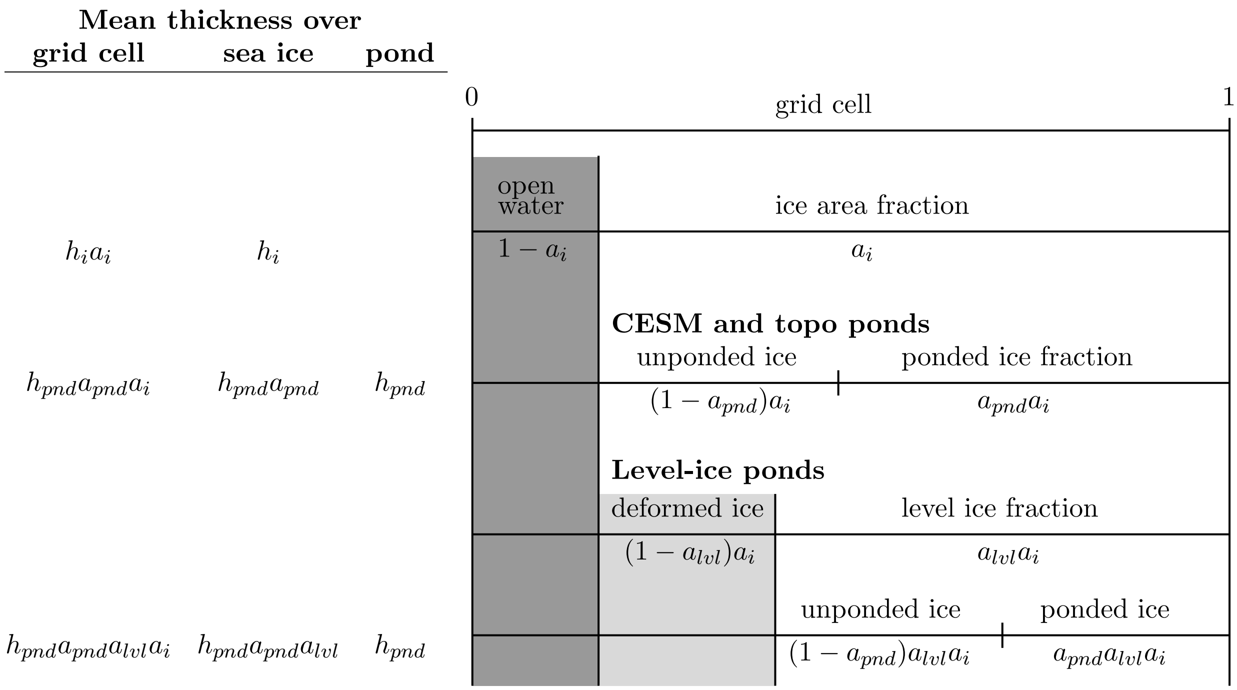

Figure 1: Melt pond tracer definitions. The graphic on the right illustrates the *grid cell* fraction of ponds or level ice as defined by the tracers. The chart on the left provides corresponding ice thickness and pond depth averages over the grid cell, sea ice and pond area fractions.

(These equations represent quantities within one thickness category; all melt pond calculations are performed for each category, separately.) Equation (14) 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 (Eq. 13), 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.

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

$$a_{pnd}^{t+\Delta t} = \frac{a_{pnd}^t a_i^t}{a_i^{t+\Delta t}}. \tag{15}$$

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

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

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,

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

Thus,

$$\begin{aligned} a_{pnd}^{t+\Delta t}(m) &= \frac{\sum_n a_{pnd}^t(n) a_i^t(n) - \sum_{n \neq m} a_{pnd}^{t+\Delta t}(n) a_i^{t+\Delta t}(n)}{a_i^{t+\Delta t}(m)} \\ &= \frac{a_{pnd}^t(m) a_i^t(m) + \sum_{n \neq m} \Delta \left(a_{pnd} a_i\right)^{t+\Delta t}}{a_i^{t+\Delta t}(m)} \end{aligned} \tag{16}$$

This is more complicated because of the Δ term on the right-hand side, which is handled manually in **ice_itd.F90**. 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 section 3.6.1.

3.1.2 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 [24].

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 **ice_therm_itd.F90**. The first-year area tracer is discussed in [2].

3.1.3 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. 11 and 12), 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 **ice_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 **ice_in**, and the number of aerosol species is set in **comp_ice**; CESM uses 3. Results using the aerosol code in the context of CESM are documented in [22]. 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.

3.2 Horizontal transport

We wish to solve the continuity or transport equation (3) for the fractional ice area in each thickness category n. Equation (3) describes the conservation of ice area under horizontal transport. It is obtained from (2) by discretizing g and neglecting the second and third terms on the right-hand side, which are treated separately (Sections 3.3 and 3.4).

There are similar conservation equations for ice volume (eq. 4), snow volume (eq 5), ice energy and snow energy:

$$\frac{\partial e_{ink}}{\partial t} + \nabla \cdot (e_{ink}\mathbf{u}) = 0, \tag{17}$$

$$\frac{\partial e_{snk}}{\partial t} + \nabla \cdot (e_{snk}\mathbf{u}) = 0. \tag{18}$$

By default, ice and snow are assumed to have constant densities, so that volume conservation is equivalent to mass conservation. Variable-density ice and snow layers can be transported conservatively by defining tracers corresponding to ice and snow density, as explained in the introductory comments in **ice_transport_remap.F90**. Prognostic equations for ice and/or snow density may be included in future model versions but have not yet been implemented.

Two transport schemes are available: upwind and the incremental remapping scheme of [8] as modified for sea ice by [34]. The remapping scheme has several desirable features:

- It conserves the quantity being transported (area, volume, or energy).
- It is non-oscillatory; that is, it does not create spurious ripples in the transported fields.
- It preserves tracer monotonicity. That is, it does not create new extrema in the thickness and enthalpy fields; the values at time $m+1$ are bounded by the values at time m.
- It is second-order accurate in space and therefore is much less diffusive than first-order schemes (e.g., upwind). The accuracy may be reduced locally to first order to preserve monotonicity.
- It is efficient for large numbers of categories or tracers. Much of the work is geometrical and is performed only once per grid cell instead of being repeated for each quantity being transported.

The time step is limited by the requirement that trajectories projected backward from grid cell corners are confined to the four surrounding cells; this is what is meant by incremental remapping as opposed to general remapping. This requirement leads to a CFL-like condition,

$$\frac{\max|\mathbf{u}|\Delta t}{\Delta x} \leq 1.$$

For highly divergent velocity fields the maximum time step must be reduced by a factor of two to ensure that trajectories do not cross. However, ice velocity fields in climate models usually have small divergences per time step relative to the grid size.

The remapping algorithm can be summarized as follows:

1. Given mean values of the ice area and tracer fields in each grid cell, construct linear approximations of these fields. Limit the field gradients to preserve monotonicity.

2. Given ice velocities at grid cell corners, identify departure regions for the fluxes across each cell edge. Divide these departure regions into triangles and compute the coordinates of the triangle vertices.

3. Integrate the area and tracer fields over the departure triangles to obtain the area, volume, and energy transported across each cell edge.

4. Given these transports, update the state variables.

Since all scalar fields are transported by the same velocity field, step (2) is done only once per time step. The other three steps are repeated for each field in each thickness category. These steps are described below.

3.2.1 Reconstructing area and tracer fields

First, using the known values of the state variables, the ice area and tracer fields are reconstructed in each grid cell as linear functions of x and y. For each field we compute the value at the cell center (i.e., at the origin of a 2D Cartesian coordinate system defined for that grid cell), along with gradients in the x and y directions. The gradients are limited to preserve monotonicity. When integrated over a grid cell, the reconstructed fields must have mean values equal to the known state variables, denoted by $\bar{a}$ for fractional area, $\tilde{h}$ for thickness, and $\hat{q}$ for enthalpy. The mean values are not, in general, equal to the values at the cell center. For example, the mean ice area must equal the value at the centroid, which may not lie at the cell center.

Consider first the fractional ice area, the analog to fluid density ρ in [8]. For each thickness category we construct a field $a(\mathbf{r})$ whose mean is $\bar{a}$, where $\mathbf{r} = (x, y)$ is the position vector relative to the cell center. That is, we require

$$\int_A a\, dA = \bar{a}\, A, \tag{19}$$

where $A = \int_A dA$ is the grid cell area. Equation (19) is satisfied if $a(\mathbf{r})$ has the form

$$a(\mathbf{r}) = \bar{a} + \alpha_a \langle \nabla a \rangle \cdot (\mathbf{r} - \bar{\mathbf{r}}), \tag{20}$$

where $\langle \nabla a \rangle$ is a centered estimate of the area gradient within the cell, α_a is a limiting coefficient that enforces monotonicity, and $\bar{\mathbf{r}}$ is the cell centroid:

$$\bar{\mathbf{r}} = \frac{1}{A} \int_A \mathbf{r}\, dA.$$

It follows from (20) that the ice area at the cell center ($\mathbf{r} = 0$) is

$$a_c = \bar{a} - a_x \overline{x} - a_y \overline{y},$$

where $a_x = \alpha_a (\partial a / \partial x)$ and $a_y = \alpha_a (\partial a / \partial y)$ are the limited gradients in the x and y directions, respectively, and the components of $\bar{\mathbf{r}}$, $\overline{x} = \int_A x\, dA / A$ and $\overline{y} = \int_A y\, dA / A$, are evaluated using the triangle integration formulas described in Section 3.2.3. These means, along with higher-order means such as $\overline{x^2}$, $\overline{xy}$, and $\overline{y^2}$, are computed once and stored.

Next consider the ice and snow thickness and enthalpy fields. Thickness is analogous to the tracer concentration T in [8], but there is no analog in [8] to the enthalpy. The reconstructed ice or snow thickness $h(\mathbf{r})$ and enthalpy $q(\mathbf{r})$ must satisfy

$$\int_A a\,h\,dA = \bar{a}\,\tilde{h}\,A, \tag{21}$$

$$\int_A a\,h\,q\,dA = \bar{a}\,\tilde{h}\,\hat{q}\,A, \tag{22}$$

where $\tilde{h} = h(\tilde{\mathbf{r}})$ is the thickness at the center of ice area, and $\hat{q} = q(\hat{\mathbf{r}})$ is the enthalpy at the center of ice or snow volume. Equations (21) and (22) are satisfied when $h(\mathbf{r})$ and $q(\mathbf{r})$ are given by

$$h(\mathbf{r}) = \tilde{h} + \alpha_h \langle \nabla h \rangle \cdot (\mathbf{r} - \tilde{\mathbf{r}}), \tag{23}$$

$$q(\mathbf{r}) = \hat{q} + \alpha_q \langle \nabla q \rangle \cdot (\mathbf{r} - \hat{\mathbf{r}}), \tag{24}$$

where α_h and α_q are limiting coefficients. The center of ice area, $\tilde{\mathbf{r}}$, and the center of ice or snow volume, $\hat{\mathbf{r}}$, are given by

$$\tilde{\mathbf{r}} = \frac{1}{\bar{a}\,A}\int_A a\,\mathbf{r}\,dA,$$

$$\hat{\mathbf{r}} = \frac{1}{\bar{a}\,\tilde{h}\,A}\int_A a\,h\,\mathbf{r}\,dA.$$

Evaluating the integrals, we find that the components of $\tilde{\mathbf{r}}$ are

$$\tilde{x} = \frac{a_c\overline{x} + a_x\overline{x^2} + a_y\overline{xy}}{\bar{a}},$$

$$\tilde{y} = \frac{a_c\overline{y} + a_x\overline{xy} + a_y\overline{y^2}}{\bar{a}},$$

and the components of $\hat{\mathbf{r}}$ are

$$\hat{x} = \frac{c_1\overline{x} + c_2\overline{x^2} + c_3\overline{xy} + c_4\overline{x^3} + c_5\overline{x^2y} + c_6\overline{xy^2}}{\bar{a}\,\tilde{h}},$$

$$\hat{y} = \frac{c_1\overline{y} + c_2\overline{xy} + c_3\overline{y^2} + c_4\overline{x^2y} + c_5\overline{xy^2} + c_6\overline{y^3}}{\bar{a}\,\tilde{h}},$$

where

$$\begin{aligned}
c_1 &\equiv a_c h_c, \\
c_2 &\equiv a_c h_x + a_x h_c, \\
c_3 &\equiv a_c h_y + a_y h_c, \\
c_4 &\equiv a_x h_x, \\
c_5 &\equiv a_x h_y + a_y h_x, \\
c_6 &\equiv a_y h_y.
\end{aligned}$$

From (23) and (24), the thickness and enthalpy at the cell center are given by

$$h_c = \tilde{h} - h_x\tilde{x} - h_y\tilde{y},$$

$$q_c = \hat{q} - q_x\hat{x} - q_y\hat{y},$$

where h_x, h_y, q_x and q_y are the limited gradients of thickness and enthalpy. The surface temperature is treated the same way as ice or snow thickness, but it has no associated enthalpy. Tracers obeying conservation equations of the form (11) and (12) are treated in analogy to ice and snow enthalpy, respectively.

We preserve monotonicity by van Leer limiting. If $\bar{\phi}(i,j)$ denotes the mean value of some field in grid cell (i,j), we first compute centered gradients of $\bar{\phi}$ in the x and y directions, then check whether these gradients give values of ϕ within cell (i,j) that lie outside the range of $\bar{\phi}$ in the cell and its eight neighbors. Let $\bar{\phi}_{\max}$ and $\bar{\phi}_{\min}$ be the maximum and minimum values of $\bar{\phi}$ over the cell and its neighbors, and let $\phi_{\max}$ and $\phi_{\min}$ be the maximum and minimum values of the reconstructed ϕ within the cell. Since the reconstruction is linear, $\phi_{\max}$ and $\phi_{\min}$ are located at cell corners. If $\phi_{\max} > \bar{\phi}_{\max}$ or $\phi_{\min} < \bar{\phi}_{\min}$, we multiply the unlimited gradient by $\alpha = \min(\alpha_{\max}, \alpha_{\min})$, where

$$\alpha_{\max} = (\bar{\phi}_{\max} - \bar{\phi})/(\phi_{\max} - \bar{\phi}),$$

$$\alpha_{\min} = (\bar{\phi}_{\min} - \bar{\phi})/(\phi_{\min} - \bar{\phi}).$$

Otherwise the gradient need not be limited.

Earlier versions of CICE (through 3.14) computed gradients in physical space. In version 4.0, gradients are computed in a scaled space in which each grid cell has sides of unit length. The origin is at the cell center, and the four vertices are located at (0.5, 0.5), (-0.5,0.5),(-0.5, -0.5) and (0.5, -0.5). In this coordinate system, several of the above grid-cell-mean quantities vanish (because they are odd functions of x and/or y), but they have been retained in the code for generality.

3.2.2 Locating departure triangles

The method for locating departure triangles is discussed in detail by [8]. The basic idea is illustrated in Figure 2, which shows a shaded quadrilateral departure region whose contents are transported to the target or home grid cell, labeled H. The neighboring grid cells are labeled by compass directions: NW, N, NE, W, and E. The four vectors point along the velocity field at the cell corners, and the departure region is formed by joining the starting points of these vectors. Instead of integrating over the entire departure region, it is convenient to compute fluxes across cell edges. We identify departure regions for the north and east edges of each cell, which are also the south and west edges of neighboring cells. Consider the north edge of the home cell, across which there are fluxes from the neighboring NW and N cells. The contributing region from the NW cell is a triangle with vertices abc, and that from the N cell is a quadrilateral that can be divided into two triangles with vertices acd and ade. Focusing on triangle abc, we first determine the coordinates of vertices b and c relative to the cell corner (vertex a), using Euclidean geometry to find vertex c. Then we translate the three vertices to a coordinate system centered in the NW cell. This translation is needed in order to integrate fields (Section 3.2.3) in the coordinate system where they have been reconstructed (Section 3.2.1). Repeating this process for the north and east edges of each grid cell, we compute the vertices of all the departure triangles associated with each cell edge.

Figure 3, reproduced from [8], shows all possible triangles that can contribute fluxes across the north edge of a grid cell. There are 20 triangles, which can be organized into five groups of four mutually exclusive triangles as shown in Table 3. In this table, (x_1, y_1) and (x_2, y_2) are the Cartesian coordinates of the departure points relative to the northwest and northeast cell corners, respectively. The departure points are joined by a straight line that intersects the west edge at $(0, y_a)$ relative to the northwest corner and intersects the east edge at $(0, y_b)$ relative to the northeast corner. The east cell triangles and selecting conditions are identical except for a rotation through 90 degrees.

This scheme was originally designed for rectangular grids. Grid cells in CICE actually lie on the surface of a sphere and must be projected onto a plane. The projection used in CICE 4.0 maps each grid cell to a

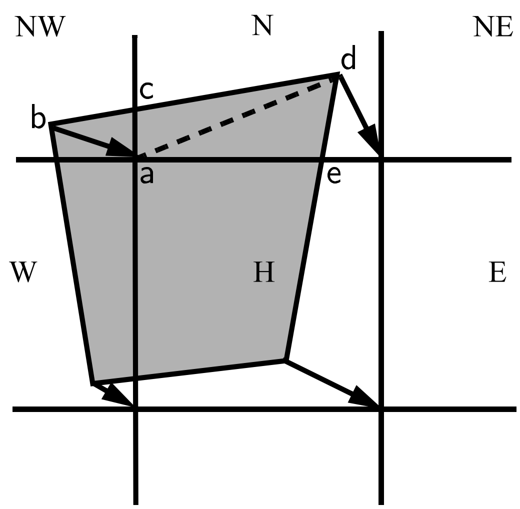

Figure 2: In incremental remapping, conserved quantities are remapped from the shaded departure region, a quadrilateral formed by connecting the backward trajectories from the four cell corners, to the grid cell labeled H. The region fluxed across the north edge of cell H consists of a triangle (abc) in the NW cell and a quadrilateral (two triangles, acd and ade) in the N cell.

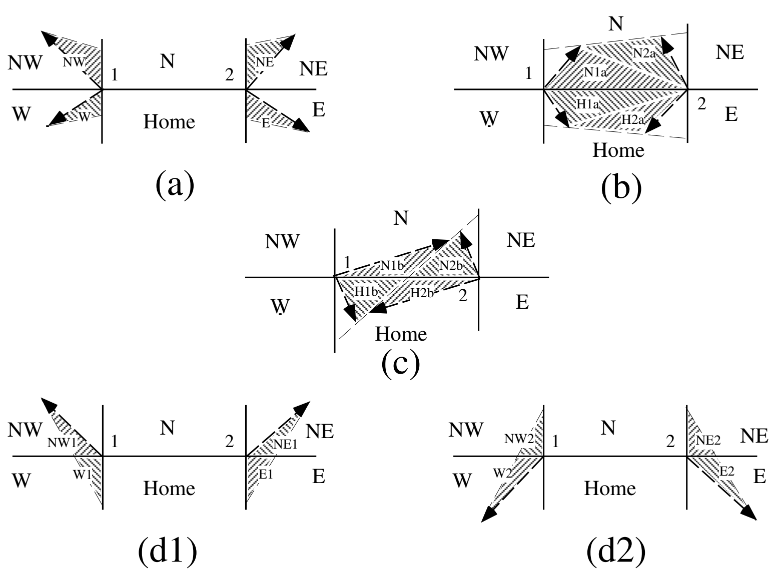

Figure 3: The 20 possible triangles that can contribute fluxes across the north edge of a grid cell.

Triangle group	Triangle label	Selecting logical condition
1	NW	$y_a > 0$ and $y_1 \geq 0$ and $x_1 < 0$
	NW1	$y_a < 0$ and $y_1 \geq 0$ and $x_1 < 0$
	W	$y_a < 0$ and $y_1 < 0$ and $x_1 < 0$
	W2	$y_a > 0$ and $y_1 < 0$ and $x_1 < 0$
2	NE	$y_b > 0$ and $y_2 \geq 0$ and $x_2 > 0$
	NE1	$y_b < 0$ and $y_2 \geq 0$ and $x_2 > 0$
	E	$y_b < 0$ and $y_2 < 0$ and $x_2 > 0$
	E2	$y_b > 0$ and $y_2 < 0$ and $x_2 > 0$
3	W1	$y_a < 0$ and $y_1 \geq 0$ and $x_1 < 0$
	NW2	$y_a > 0$ and $y_1 < 0$ and $x_1 < 0$
	E1	$y_b < 0$ and $y_2 \geq 0$ and $x_2 > 0$
	NE2	$y_b > 0$ and $y_2 < 0$ and $x_2 > 0$
4	H1a	$y_a y_b \geq 0$ and $y_a + y_b < 0$
	N1a	$y_a y_b \geq 0$ and $y_a + y_b > 0$
	H1b	$y_a y_b < 0$ and $\tilde{y}_1 < 0$
	N1b	$y_a y_b < 0$ and $\tilde{y}_1 > 0$
5	H2a	$y_a y_b \geq 0$ and $y_a + y_b < 0$
	N2a	$y_a y_b \geq 0$ and $y_a + y_b > 0$
	H2b	$y_a y_b < 0$ and $\tilde{y}_2 < 0$
	N2b	$y_a y_b < 0$ and $\tilde{y}_2 > 0$

Table 3: Evaluation of contributions from the 20 triangles across the north cell edge. The coordinates x_1, x_2, y_1, y_2, y_a, and y_b are defined in the text. We define $\tilde{y}_1 = y_1$ if $x_1 > 0$, else $\tilde{y}_1 = y_a$. Similarly, $\tilde{y}_2 = y_2$ if $x_2 < 0$, else $\tilde{y}_2 = y_b$.

square with sides of unit length. Departure triangles across a given cell edge are computed in a coordinate system whose origin lies at the midpoint of the edge and whose vertices are at (-0.5, 0) and (0.5, 0). Intersection points are computed assuming Cartesian geometry with cell edges meeting at right angles. Let CL and CR denote the left and right vertices, which are joined by line CLR. Similarly, let DL and DR denote the departure points, which are joined by line DLR. Also, let IL and IR denote the intersection points (0, y_a) and (0,y_b) respectively, and let IC = (x_c, 0) denote the intersection of CLR and DLR. It can be shown that y_a, y_b, and x_c are given by

$$\begin{aligned} y_a &= \frac{x_{CL}(y_{DM} - y_{DL}) + x_{DM}y_{DL} - x_{DL}y_{DM}}{x_{DM} - x_{DL},} \\ y_b &= \frac{x_{CR}(y_{DR} - y_{DM}) - x_{DM}y_{DR} + x_{DR}y_{DM}}{x_{DR} - x_{DM},} \\ x_c &= x_{DL} - y_{DL}\left(\frac{x_{DR} - x_{DL}}{y_{DR} - y_{DL}}\right) \end{aligned}$$

Each departure triangle is defined by three of the seven points (CL, CR, DL, DR, IL, IR, IC).

Given a 2D velocity field $\mathbf{u}$, the divergence $\nabla \cdot \mathbf{u}$ in a given grid cell can be computed from the local velocities and written in terms of fluxes across each cell edge:

$$\nabla\cdot\mathbf{u} = \frac{1}{A}\left[\left(\frac{u_{NE}+u_{SE}}{2}\right)L_E + \left(\frac{u_{NW}+u_{SW}}{2}\right)L_W + \left(\frac{u_{NE}+u_{NW}}{2}\right)L_N + \left(\frac{u_{SE}+u_{SW}}{2}\right)L_S\right], \tag{25}$$

where L is an edge length and the indices N, S, E, W denote compass directions. Equation (25) is equivalent to the divergence computed in the EVP dynamics (Section 3.5). In general, the fluxes in this expression are not equal to those implied by the above scheme for locating departure regions. For some applications it may be desirable to prescribe the divergence by prescribing the area of the departure region for each edge. This can be done in CICE 4.0 by setting `l_fixed_area` = true in **ice_transport_driver.F90** and passing the prescribed departure areas (`edgearea_e` and `edgearea_n`) into the remapping routine. An extra triangle is then constructed for each departure region to ensure that the total area is equal to the prescribed value. This idea was suggested and first implemented by Mats Bentsen of the Nansen Environmental and Remote Sensing Center (Norway), who applied an earlier version of the CICE remapping scheme to an ocean model. The implementation in CICE 4.0 is somewhat more general, allowing for departure regions lying on both sides of a cell edge. The extra triangle is constrained to lie in one but not both of the grid cells that share the edge. Since this option has yet to be fully tested in CICE, the current default is `l_fixed_area` = false.

We made one other change in the scheme of [8] for locating triangles. In their paper, departure points are defined by projecting cell corner velocities directly backward. That is,

$$\mathbf{x_D} = -\mathbf{u}\,\Delta t, \tag{26}$$

where $\mathbf{x}_D$ is the location of the departure point relative to the cell corner and $\mathbf{u}$ is the velocity at the corner. This approximation is only first-order accurate. Accuracy can be improved by estimating the velocity at the midpoint of the trajectory.

3.2.3 Integrating fields

Next, we integrate the reconstructed fields over the departure triangles to find the total area, volume, and energy transported across each cell edge. Area transports are easy to compute since the area is linear in x and y. Given a triangle with vertices $\mathbf{x_i} = (x_i, y_i)$, $i \in \{1, 2, 3\}$, the triangle area is

$$A_T = \frac{1}{2}\left|(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)\right|. \tag{27}$$

The integral F_a of any linear function $f(\mathbf{r})$ over a triangle is given by

$$F_a = A_T f(\mathbf{x_0}), \tag{28}$$

where $\mathbf{x}_0 = (x_0, y_0)$ is the triangle midpoint,

$$\mathbf{x}_0 = \frac{1}{3}\sum_{i=1}^{3} \mathbf{x}_i. \tag{29}$$

To compute the area transport, we evaluate the area at the midpoint,

$$a(\mathbf{x}_0) = a_c + a_x x_0 + a_y y_0, \tag{30}$$

and multiply by A_T. By convention, northward and eastward transport is positive, while southward and westward transport is negative.

Equation (28) cannot be used for volume transport, because the reconstructed volumes are quadratic functions of position. (They are products of two linear functions, area and thickness.) The integral of a quadratic polynomial over a triangle requires function evaluations at three points,

$$F_h = \frac{A_T}{3}\sum_{i=1}^{3} f\left(\mathbf{x}_i'\right), \tag{31}$$

where $\mathbf{x}_i' = (\mathbf{x}_0 + \mathbf{x}_i)/2$ are points lying halfway between the midpoint and the three vertices. [8] use this formula to compute transports of the product ρT, which is analogous to ice volume. Equation (31) does not work for ice and snow energies, which are cubic functions—products of area, thickness, and enthalpy. Integrals of a cubic polynomial over a triangle can be evaluated using a four-point formula [52]:

$$F_q = A_T \left[-\frac{9}{16} f(\mathbf{x}_0) + \frac{25}{48}\sum_{i=1}^{3} f(\mathbf{x}_i'') \right] \tag{32}$$

where $\mathbf{x_i}'' = (3\mathbf{x}_0 + 2\mathbf{x}_i)/5$. To evaluate functions at specific points, we must compute many products of the form $a(\mathbf{x})\, h(\mathbf{x})$ and $a(\mathbf{x})\, h(\mathbf{x})\, q(\mathbf{x})$, where each term in the product is the sum of a cell-center value and two displacement terms. In the code, the computation is sped up by storing some sums that are used repeatedly.

3.2.4 Updating state variables

Finally, we compute new values of the state variables in each ice category and grid cell. The new fractional ice areas $a_{in}'(i,j)$ are given by

$$a_{in}'(i,j) = a_{in}(i,j) + \frac{F_{aE}(i-1,j) - F_{aE}(i,j) + F_{aN}(i,j-1) - F_{aN}(i,j)}{A(i,j)} \tag{33}$$

where $F_{aE}(i,j)$ and $F_{aN}(i,j)$ are the area transports across the east and north edges, respectively, of cell (i,j), and $A(i,j)$ is the grid cell area. All transports added to one cell are subtracted from a neighboring cell; thus (33) conserves total ice area.

The new ice volumes and energies are computed analogously. New thicknesses are given by the ratio of volume to area, and enthalpies by the ratio of energy to volume. Tracer monotonicity is ensured because

$$h' = \frac{\int_A a\, h\, dA}{\int_A a\, dA},$$

$$q' = \frac{\int_A a\,h\,q\,dA}{\int_A a\,h\,dA},$$

where h' and q' are the new-time thickness and enthalpy, given by integrating the old-time ice area, volume, and energy over a Lagrangian departure region with area A. That is, the new-time thickness and enthalpy are weighted averages over old-time values, with non-negative weights a and ah. Thus the new-time values must lie between the maximum and minimum of the old-time values.

3.3 Transport in thickness space

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

$$\frac{\partial g}{\partial t} + \frac{\partial}{\partial h}(fg) = 0, \tag{34}$$

which is obtained from (2) by neglecting the first and third terms on the right-hand side. We use the remapping method of [33], 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 (Section 3.6) 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:

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

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

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

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:

$$\int_{H_L}^{H_R} g\,dh = a_n, \tag{35}$$

$$\int_{H_L}^{H_R} h\,g\,dh = v_n. \tag{36}$$

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 (35) and (36) can be written as

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

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

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

$$g_0 = \frac{6a_n}{\eta_R^2}\left(\frac{2\eta_R}{3} - \eta_n\right), \tag{37}$$

$$g_1 = \frac{12a_n}{\eta_R^3}\left(\eta_n - \frac{\eta_R}{2}\right). \tag{38}$$

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

$$g(0) = \frac{6a_n}{\eta_R^2}\left(\frac{2\eta_R}{3} - \eta_n\right) = g_0, \tag{39}$$

$$g(\eta_R) = \frac{6a_n}{\eta_R^2}\left(\eta_n - \frac{\eta_R}{3}\right). \tag{40}$$

Equation (39) 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, (40) 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 (37) and (38) 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, (37) and (38) 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.

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

$$\Delta a_n = \int_{H_n}^{H_n^*} g\, dh, \tag{41}$$

$$\Delta v_n = \int_{H_n}^{H_n^*} h\, g\, dh. \tag{42}$$

If $H_n^* < H_N$, ice area and volume are transferred from category $n+1$ to n using (41) and (42) 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 (37) and (38). 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.

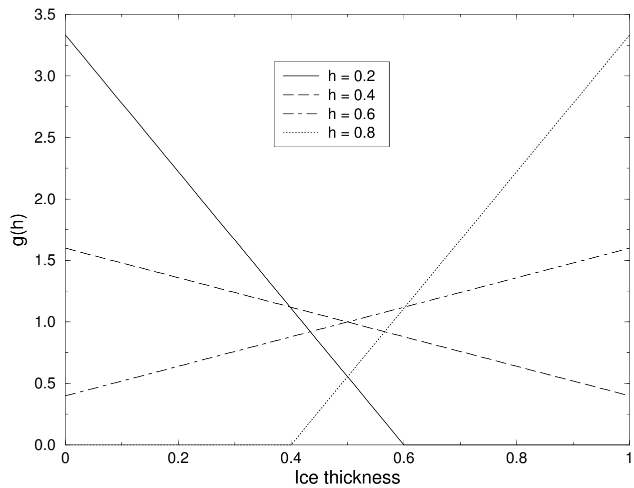

Figure 4: Linear approximation of the thickness distribution function $g(h)$ for an ice category with left boundary $H_L = 0$, right boundary $H_R = 1$, fractional area $a_n = 1$, and mean ice thickness $h_n = 0.2$, 0.4, 0.6, and 0.8.

Other conserved quantities are transferred in proportion to the ice volume Δv_{in}. (We now use the subscripts i and s to distinguish ice from snow.) For example, the transferred snow volume is $\Delta v_{sn} = v_{sn}(\Delta v_{in}/v_{in})$, and 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

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

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 [3], 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.

3.4 Mechanical redistribution

The last term on the right-hand side of (2) is ψ, which describes the redistribution of ice in thickness space due to ridging and other mechanical processes. The mechanical redistribution scheme in CICE is based on [54], [44], [21], [14], and [35]. 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 [54] and set

$$b(h) = \begin{cases} \frac{2}{G^*}(1 - \frac{G(h)}{G^*}) & \text{if } G(h) < G^* \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

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:

$$a_{Pn} = \frac{2}{G^*}(G_n - G_{n-1})\left(1 - \frac{G_{n-1} + G_n}{2G^*}\right), \tag{44}$$

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 (44) applies to categories with $G_n < G^*$. If $G_{n-1} < G^* < G_n$, then (44) 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 [54] we set $G^* = 0.15$.

If the spatial resolution is too fine for a given time step Δt, the weighting function (43) 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 [35]:

$$b(h) = \frac{\exp[-G(h)/a^*]}{a^*[1 - \exp(-1/a^*)]} \tag{45}$$

When integrated between category boundaries, (45) implies

$$a_{Pn} = \frac{\exp(-G_{n-1}/a^*) - \exp(-G_n/a^*)}{1 - \exp(-1/a^*)} \tag{46}$$

This weighting function is used if `krdg_partic` = 1 in the namelist. From (45), the mean value of G for ice participating in ridging is a^*, as compared to $G^*/3$ for (43). 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 [35] 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 [21]. 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 [14]. 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

$$f_m^{\mathrm{area}} = \frac{H_R - H_L}{H_{\max} - H_{\min}}, \tag{47}$$

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

$$f_m^{\mathrm{vol}} = \frac{(H_R)^2 - (H_L)^2}{(H_{\max})^2 - (H_{\min})^2}. \tag{48}$$

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 [35]:

$$g_R(h) \propto \exp[-(h - H_{\min})/\lambda] \tag{49}$$

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 $\mathrm{m}^{1/2}$. Thus the mean ridge thickness increases in proportion to $h_n^{1/2}$, as in [21]. The value $\mu = 4.0\ \mathrm{m}^{1/2}$ gives λ in the range 1–4 m for most ridged ice. Ice strengths with $\mu = 4.0\ \mathrm{m}^{1/2}$ and `krdg_redist` = 1 are roughly comparable to the strengths with $H^* = 50$ m and `krdg_redist` = 0.

From (49) it can be shown that the fractional area going to category m as a result of ridging is

$$f_m^{\mathrm{area}} = \exp[-(H_{m-1} - H_{\min})/\lambda] - \exp[-(H_m - H_{\min})/\lambda]. \tag{50}$$

The fractional volume going to category m is

$$f_m^{\mathrm{vol}} = \frac{(H_{m-1} + \lambda)\exp[-(H_{m-1} - H_{\min})/\lambda] - (H_m + \lambda)\exp[-(H_m - H_{\min})/\lambda]}{H_{min} + \lambda}. \tag{51}$$

Equations (50) and (51) replace (47) and (48) 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_{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 [14], R_{net} is given by

$$R_{\mathrm{net}} = \frac{C_s}{2}(\Delta - |D_D|) - \min(D_D, 0), \tag{52}$$

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

$$R_{\mathrm{net}} = \left[a_{P0} + \sum_{n=1}^{N} a_{Pn}\left(1 - \frac{1}{k_n}\right)\right] R_{\mathrm{tot}}. \tag{53}$$

Given R_{net} from (52), we use (53) to compute R_{tot}. Then the area ridged in category n is given by $a_{rn} = r_n \Delta t$, where $r_n = a_{Pn} R_{\mathrm{tot}}$. 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 (47) and (48) (or 50) and (51)) 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 offered as history output. They are simply related:

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

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 [20]:

$$P = P^* h \exp[-C(1-a_i)], \tag{54}$$

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 [44], 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

$$P = C_f\, C_p\, \beta \sum_{n=1}^{N_C} \left[-a_{Pn}\, h_n^2 + \frac{a_{Pn}}{k_n} \left(\frac{(H_n^{\max})^3 - (H_n^{\min})^3}{3(H_n^{\max} - H_n^{\min})} \right) \right], \tag{55}$$

where $C_P = (g/2)(\rho_i/\rho_w)(\rho_w - \rho_i)$, $\beta = R_{\rm tot}/R_{\rm net} > 1$ from (53), and C_f is an empirical parameter that accounts for frictional energy dissipation. Following [14], 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:

$$P = C_f\, C_p\, \beta \sum_{n=1}^{N_C} \left[-a_{Pn}\, h_n^2 + \frac{a_{Pn}}{k_n} \left(H_{\min}^2 + 2H_{\min}\lambda + 2\lambda^2 \right) \right] \tag{56}$$

The energy-based ice strength given by (55) or (56) is more physically realistic than the strength given by (54). However, use of (54) is less likely to allow numerical instability at a given resolution and time step. See [35] for more details.

3.5 Dynamics

There are now different rheologies available in the CICE code. The elastic-viscous-plastic (EVP) model represents a modification of the standard viscous-plastic (VP) model for sea ice dynamics [20]. The elastic-anisotropic-plastic (EAP) model, on the other hand, explicitly accounts for the observed sub-continuum anisotropy of the sea ice cover [59, 58]. If `kdyn` = 1 in the namelist then the EVP rheology is used (module **ice_dyn_evp.F90**), while `kdyn` = 2 is associated with the EAP rheology (**ice_dyn_eap.F90**). At times scales associated with the wind forcing, the EVP model reduces to the VP model while the EAP model reduces to the anisotropic rheology described in detail in [59, 56]. At shorter time scales the adjustment process takes place in both models by a numerically more efficient elastic wave mechanism. While retaining the essential physics, this elastic wave modification leads to a fully explicit numerical scheme which greatly improves the model's computational efficiency.

The EVP sea ice dynamics model is thoroughly documented in [25], [23], [26] and [27] and the EAP dynamics in [56]. Simulation results and performance of the EVP and EAP models have been compared with the VP model and with each other in realistic simulations of the Arctic respectively in [29] and [56]. Here we summarize the equations and direct the reader to the above references for details. The numerical implementation in this code release is that of [26] and [27], with revisions to the numerical solver as in [5]. The implementation of the EAP sea ice dynamics into CICE is described in detail in [56].

3.5.1 Momentum

The force balance per unit area in the ice pack is given by a two-dimensional momentum equation [20], obtained by integrating the 3D equation through the thickness of the ice in the vertical direction:

$$m\frac{\partial \mathbf{u}}{\partial t} = \nabla\cdot\sigma + \vec{\tau}_a + \vec{\tau}_w - \hat{k}\times mf\mathbf{u} - mg\nabla H_\circ, \tag{57}$$

where m is the combined mass of ice and snow per unit area and $\vec{\tau}_a$ and $\vec{\tau}_w$ are wind and ocean stresses, respectively. The strength of the ice is represented by the internal stress tensor σ_{ij}, and the other two terms on the right hand side are stresses due to Coriolis effects and the sea surface slope. The parameterization for the wind and ice-ocean stress terms must contain the ice concentration as a multiplicative factor to be consistent with the formal theory of free drift in low ice concentration regions. A careful explanation of the issue and its continuum solution is provided in [27] and [7].

The momentum equation is discretized in time as follows, for the classic EVP approach. First, for clarity, the two components of (57) are

$$\begin{aligned}
m\frac{\partial u}{\partial t} &= \frac{\partial \sigma_{1j}}{\partial x_j} + \tau_{ax} + a_i c_w \rho_w \left|\mathbf{U}_w - \mathbf{u}\right| \left[(U_w - u)\cos\theta - (V_w - v)\sin\theta\right] + mfv - mg\frac{\partial H_\circ}{\partial x}, \\
m\frac{\partial v}{\partial t} &= \frac{\partial \sigma_{2j}}{\partial x_j} + \tau_{ay} + a_i c_w \rho_w \left|\mathbf{U}_w - \mathbf{u}\right| \left[(U_w - u)\sin\theta - (V_w - v)\cos\theta\right] - mfu - mg\frac{\partial H_\circ}{\partial y}.
\end{aligned}$$

In the code, $\mathtt{vrel} = a_i c_w \rho_w \left|\mathbf{U}_w - \mathbf{u}^k\right|$, where k denotes the subcycling step. The following equations illustrate the time discretization and define some of the other variables used in the code.

$$\underbrace{\left(\frac{m}{\Delta t_e} + \mathtt{vrel}\cos\theta\right)}_{\mathtt{cca}} u^{k+1} - \underbrace{\left(mf + \mathtt{vrel}\sin\theta\right)}_{\mathtt{ccb}} v^{k+1} = \underbrace{\frac{\partial \sigma_{1j}^{k+1}}{\partial x_j}}_{\mathtt{strintx}} + \underbrace{\tau_{ax} - mg\frac{\partial H_\circ}{\partial x}}_{\mathtt{forcex}} + \mathtt{vrel}\underbrace{\left(U_w\cos\theta - V_w\sin\theta\right)}_{\mathtt{waterx}} + \frac{m}{\Delta t_e}u^k, \tag{58}$$

$$\underbrace{\left(mf + \mathtt{vrel}\sin\theta\right)}_{\mathtt{ccb}} u^{k+1} + \underbrace{\left(\frac{m}{\Delta t_e} + \mathtt{vrel}\cos\theta\right)}_{\mathtt{cca}} v^{k+1} = \underbrace{\frac{\partial \sigma_{2j}^{k+1}}{\partial x_j}}_{\mathtt{strinty}} + \underbrace{\tau_{ay} - mg\frac{\partial H_\circ}{\partial y}}_{\mathtt{forcey}} + \mathtt{vrel}\underbrace{\left(U_w\sin\theta + V_w\cos\theta\right)}_{\mathtt{watery}} + \frac{m}{\Delta t_e}v^k, \tag{59}$$

and `vrel·waterx(y) = taux(y)`.

We solve this system of equations analytically for u^{k+1} and v^{k+1}. Define

$$\hat{u} = F_u + \tau_{ax} - mg\frac{\partial H}{\partial x} + \mathtt{vrel}\left(U_w \cos\theta - V_w \sin\theta\right) + \frac{m}{\Delta t_e}u^k \tag{60}$$

$$\hat{v} = F_v + \tau_{ay} - mg\frac{\partial H}{\partial y} + \mathtt{vrel}\left(U_w \sin\theta + V_w \cos\theta\right) + \frac{m}{\Delta t_e}v^k, \tag{61}$$

where $\mathbf{F} = \nabla \cdot \sigma^{k+1}$. Then

$$\begin{aligned}
\left(\frac{m}{\Delta t_e} + \mathtt{vrel}\cos\theta\right) u^{k+1} - \left(mf + \mathtt{vrel}\sin\theta\right) v^{k+1} &= \hat{u} \\
\left(mf + \mathtt{vrel}\sin\theta\right) u^{k+1} + \left(\frac{m}{\Delta t_e} + \mathtt{vrel}\cos\theta\right) v^{k+1} &= \hat{v}.
\end{aligned}$$

Solving simultaneously for u^{k+1} and v^{k+1},

$$\begin{aligned}
u^{k+1} &= \frac{a\hat{u} + b\hat{v}}{a^2 + b^2} \\
v^{k+1} &= \frac{a\hat{v} - b\hat{u}}{a^2 + b^2},
\end{aligned}$$

where

$$a = \frac{m}{\Delta t_e} + \mathtt{vrel}\cos\theta \tag{62}$$

$$b = mf + \mathtt{vrel}\sin\theta. \tag{63}$$

When the subcycling is finished for each (thermodynamic) time step, the ice-ocean stress must be constructed from `taux(y)` and the terms containing `vrel` on the left hand side of the equations. This is done in subroutine *evp_finish*. [CHECK akt changes and hibler-bryan stress]

3.5.2 Internal stress

For convenience we formulate the stress tensor σ in terms of $\sigma_1 = \sigma_{11} + \sigma_{22}$, $\sigma_2 = \sigma_{11} - \sigma_{22}$, and introduce the divergence, D_D, and the horizontal tension and shearing strain rates, D_T and D_S respectively.

Elastic-Viscous-Plastic

In the EVP model the internal stress tensor is determined from a regularized version of the VP constitutive law,

$$\frac{1}{E}\frac{\partial \sigma_1}{\partial t} + \frac{\sigma_1}{2\zeta} + \frac{P}{2\zeta} = D_D, \tag{64}$$

$$\frac{1}{E}\frac{\partial \sigma_2}{\partial t} + \frac{\sigma_2}{2\eta} = D_T, \tag{65}$$

$$\frac{1}{E}\frac{\partial \sigma_{12}}{\partial t} + \frac{\sigma_{12}}{2\eta} = \frac{1}{2}D_S, \tag{66}$$

where

$$\begin{aligned} D_D &= \dot{\epsilon}_{11} + \dot{\epsilon}_{22}, && (67) \\ D_T &= \dot{\epsilon}_{11} - \dot{\epsilon}_{22}, && (68) \\ D_S &= 2\dot{\epsilon}_{12}, && (69) \\ \dot{\epsilon}_{ij} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \\ \zeta &= \frac{P}{2\Delta}, \\ \eta &= \frac{P}{2\Delta e^2}, \\ \Delta &= \left[D_D^2 + \frac{1}{e^2}\left(D_T^2 + D_S^2\right)\right]^{1/2}, \end{aligned}$$

and P is a function of the ice thickness and concentration, described in Section 3.4. The dynamics component employs a "replacement pressure" (see [18], for example), which serves to prevent residual ice motion due to spatial variations of P when the rates of strain are exactly zero.

Viscosities are updated during the subcycling, so that the entire dynamics component is subcycled within the time step, and the elastic parameter E is defined in terms of a damping timescale T for elastic waves, $\Delta t_e < T < \Delta t$, as

$$E = \frac{\zeta}{T},$$

where $T = E_\circ \Delta t$ and $E_\circ$ (`eyc`) is a tunable parameter less than one. The stress equations (64–66) become

$$\begin{aligned} \frac{\partial \sigma_1}{\partial t} + \frac{\sigma_1}{2T} + \frac{P}{2T} &= \frac{P}{2T\Delta} D_D, \\ \frac{\partial \sigma_2}{\partial t} + \frac{e^2 \sigma_2}{2T} &= \frac{P}{2T\Delta} D_T, \\ \frac{\partial \sigma_{12}}{\partial t} + \frac{e^2 \sigma_{12}}{2T} &= \frac{P}{4T\Delta} D_S. \end{aligned}$$

All coefficients on the left-hand side are constant except for P, which changes only on the longer time step Δt. This modification compensates for the decreased efficiency of including the viscosity terms in the subcycling. (Note that the viscosities do not appear explicitly.) Choices of the parameters used to define E, T and Δt_e are discussed in Section 4.4. [REVISE for EAP and revised evp]

The bilinear discretization used for the stress terms $\partial \sigma_{ij}/\partial x_j$ in the momentum equation is now used, which enabled the discrete equations to be derived from the continuous equations written in curvilinear coordinates. In this manner, metric terms associated with the curvature of the grid are incorporated into the discretization explicitly. Details pertaining to the spatial discretization are found in [26].

Elastic-Anisotropic-Plastic

In the EAP model the internal stress tensor is related to the geometrical properties and orientation of underlying virtual diamond shaped floes (see Fig. 5). In contrast to the isotropic EVP rheology, the anisotropic plastic yield curve within the EAP rheology depends on the relative orientation of the diamond shaped floes (unit vector $\mathbf{r}$ in Fig. 5), with respect to the principal direction of the deformation rate (not shown). Local anisotropy of the sea ice cover is accounted for by an additional prognostic variable, the structure tensor $\mathbf{A}$ defined by

$$\mathbf{A} = \int_{\mathbb{S}} \vartheta(\mathbf{r}) \mathbf{r}\mathbf{r} d\mathbf{r}. \qquad (70)$$

where $\mathbb{S}$ is a unit-radius circle; $\mathbf{A}$ is a unit trace, 2×2 matrix. From now on we shall describe the orientational distribution of floes using the structure tensor. For simplicity we take the probability density function $\vartheta(\mathbf{r})$ to be Gaussian, $\vartheta(z) = \omega_1 exp(-\omega_2 z^2)$, where z is the ice floe inclination with respect to the axis x_1 of preferential alignment of ice floes (see Fig. 5), $\vartheta(z)$ is periodic with period π, and the positive coefficients ω_1 and ω_2 are calculated to ensure normalization of $\vartheta(z)$, i.e. $\int_0^{2\pi} \vartheta(z)dz = 1$. The ratio of the principal components of $\mathbf{A}$, A_1/A_2, are derived from the phenomenological evolution equation for the structure tensor $\mathbf{A}$,

$$\frac{D\mathbf{A}}{Dt} = \mathbf{F}_{iso}(\mathbf{A}) + \mathbf{F}_{frac}(\mathbf{A}, \boldsymbol{\sigma}), \tag{71}$$

where t is the time, and D/Dt is the co-rotational time derivative accounting for advection and rigid body rotation ($D\mathbf{A}/Dt = d\mathbf{A}/dt - \mathbf{W}\cdot\mathbf{A} - \mathbf{A}\cdot\mathbf{W}^T$) with $\mathbf{W}$ being the vorticity tensor. $\mathbf{F}_{iso}$ is a function that accounts for a variety of processes (thermal cracking, melting, freezing together of floes) that contribute to a more isotropic nature to the ice cover. $\mathbf{F}_{frac}$ is a function determining the ice floe re-orientation due to fracture, and explicitly depends upon sea ice stress (but not its magnitude). Following [59], based on laboratory experiments by [45] we consider four failure mechanisms for the Arctic sea ice cover. These are determined by the ratio of the principal values of the sea ice stress σ_1 and σ_2 : (i) under biaxial tension, fractures form across the perpendicular principal axes and therefore counteract any apparent redistribution of the floe orientation; (ii) if only one of the principal stresses is compressive, failure occurs through axial splitting along the compression direction; (iii) under biaxial compression with a low confinement ratio, ($\sigma_1/\sigma_2 < R$), sea ice fails Coulombically through formation of slip lines delineating new ice floes oriented along the largest compressive stress; and finally (iv) under biaxial compression with a large confinement ratio, ($\sigma_1/\sigma_2 \geq R$), the ice is expected to fail along both principal directions so that the cumulative directional effect balances to zero.

The new anisotropic rheology requires solving the evolution Eq. (71) for the structure tensor in addition to the momentum and stress equations. The evolution equation for $\mathbf{A}$ is solved within the EVP subcycling loop, and consistently with the momentum and stress evolution equations, we neglect the advection term for the structure tensor. Eq. (71) then reduces to the system of two equations:

$$\frac{\partial A_{11}}{\partial t} = -k_t\left(A_{11} - \frac{1}{2}\right) + M_{11}, \tag{72}$$

$$\frac{\partial A_{12}}{\partial t} = -k_t A_{12} + M_{12}, \tag{73}$$

where the first terms on the right hand side correspond to the isotropic contribution, F_{iso}, and M_{11} and M_{12} are the components of the term F_{frac} in Eq. (71) that are given in [59] and [56]. These evolution equations are discretized semi-implicitly in time. The degree of anisotropy is measured by the largest eigenvalue (A_1) of this tensor ($A_2 = 1 - A_1$). $A_1 = 1$ corresponds to perfectly aligned floes and $A_1 = 0.5$ to a uniform distribution of floe orientation. Note that while we have specified the aspect ratio of the diamond floes, through prescribing ϕ, we make no assumption about the size of the diamonds so that formally the theory is scale invariant.

As described in greater detail in [59], the internal ice stress for a single orientation of the ice floes can be calculated explicitly and decomposed, for an average ice thickness h, into its ridging (r) and sliding (s) contributions

$$\boldsymbol{\sigma}^b(\mathbf{r}, h) = P_r(h)\boldsymbol{\sigma}_r^b(\mathbf{r}) + P_s(h)\boldsymbol{\sigma}_s^b(\mathbf{r}), \tag{74}$$

where P_r and P_s are the ridging and sliding strengths and the ridging and sliding stresses are functions of the angle $\theta = \arctan(\dot{\epsilon}_{II}/\dot{\epsilon}_I)$, the angle y between the major principal axis of the strain rate tensor (not shown) and the structure tensor (x_1 axis in Fig. 5), and the angle z defined in Fig. 5. In the stress expressions above the underlying floes are assumed parallel, but in a continuum-scale sea ice region the floes can possess

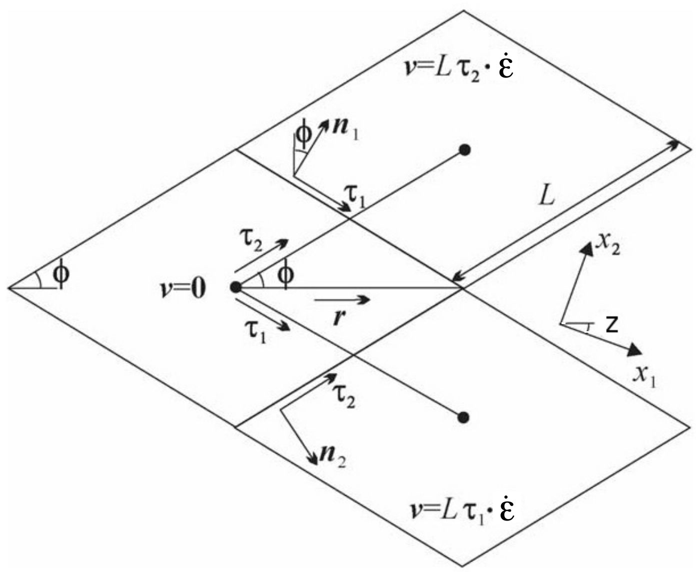

Figure 5: Geometry of interlocking diamond-shaped floes (taken from [59]). ϕ is half of the acute angle of the diamonds. L is the edge length. $\boldsymbol{n}_1$, $\boldsymbol{n}_2$ and $\boldsymbol{\tau}_1$, $\boldsymbol{\tau}_2$ are respectively the normal and tangential unit vectors along the diamond edges. $\mathbf{v} = L\boldsymbol{\tau}_2 \cdot \dot{\boldsymbol{\epsilon}}$ is the relative velocity between the two floes connected by the vector $L\boldsymbol{\tau}_2$. $\mathbf{r}$ is the unit vector along the main diagonal of the diamond. Note that the diamonds illustrated here represent one possible realisation of all possible orientations. The angle z represents the rotation of the diamonds' main axis relative to their preferential orientation along the axis x_1.

different orientations in different places and we take the mean sea ice stress over a collection of floes to be given by the average

$$\boldsymbol{\sigma}^{EAP}(h) = P_r(h) \int_{\mathbb{S}} \vartheta(\mathbf{r}) \left[\boldsymbol{\sigma}_r^b(\mathbf{r}) + k\boldsymbol{\sigma}_s^b(\mathbf{r})\right] d\mathbf{r}, \tag{75}$$

where we have introduced the friction parameter $k = P_s/P_r$ and where we identify the ridging ice strength $P_r(h)$ with the strength P described in section 1 and used within the EVP framework.

As is the case for the EVP rheology, elasticity is included in the EAP description not to describe any physical effect, but to make use of the efficient, explicit numerical algorithm used to solve the full sea ice momentum balance. We use the analogous EAP stress equations,

$$\frac{\partial\sigma_1}{\partial t} + \frac{\sigma_1}{2T} = \frac{\sigma_1^{EAP}}{2T}, \tag{76}$$

$$\frac{\partial\sigma_2}{\partial t} + \frac{\sigma_2}{2T} = \frac{\sigma_2^{EAP}}{2T}, \tag{77}$$

$$\frac{\partial\sigma_{12}}{\partial t} + \frac{\sigma_{12}}{2T} = \frac{\sigma_{12}^{EAP}}{2T}, \tag{78}$$

where the anisotropic stress $\boldsymbol{\sigma}^{EAP}$ is trilinearly interpolated from a previously constructed look-up table (Eq. (76)) for the current values of strain rate and structure tensor. The look-up table is constructed by computing the stress (normalized by the strength) from Eq. (76) for discrete values of the largest eigenvalue of the structure tensor, $\frac{1}{2} \leq A_1 \leq 1$, the angle $0 \leq \theta \leq 2\pi$, and the angle $-\pi/2 \leq y \leq \pi/2$ between the major principal axis of the strain rate tensor and the structure tensor [56]. The look-up table is stored in the additional input file, *eap_stresses*, provided with the new release of CICE. The updated stress, after the elastic relaxation, is then passed to the momentum equation and the sea ice velocities are updated in the usual manner within the subcycling loop of the EVP rheology. The structure tensor evolution equations are solved implicitly at the same frequency, Δt_e, as the ice velocities and internal stresses. Finally, to be coherent with our new rheology we compute the area loss rate due to ridging as $|\dot{\boldsymbol{\epsilon}}|\alpha_r(\theta)$, with $\alpha_r(\theta)$ and $\alpha_s(\theta)$ given by [60],

$$\alpha_r(\theta) = \frac{\sigma_{ij}^r \dot{\epsilon}_{ij}}{P_r|\dot{\boldsymbol{\epsilon}}|}, \qquad \alpha_s(\theta) = \frac{\sigma_{ij}^s \dot{\epsilon}_{ij}}{P_s|\dot{\boldsymbol{\epsilon}}|}. \tag{79}$$

Both ridging rate and sea ice strength are computed in the outer loop of the dynamics.

3.5.3 Revised approach

A modification of the standard elastic-viscous-plastic (EVP) approach for sea ice dynamics has been proposed by [5], that generalizes the EVP elastic modulus E and the time stepping approach for both momentum and stress to use an under-relaxation technique. In general terms, the momentum and stress equations become

$$\begin{aligned}
\mathbf{u}^{k+1} &= \mathbf{u}^k + \left(\breve{\mathbf{u}}^k - \mathbf{u}^{k+1}\right)\frac{1}{\beta} \\
\sigma^{k+1} &= \sigma^k + \left(\breve{\sigma}^k - \sigma^{k+1}\right)\frac{1}{\alpha}
\end{aligned}$$

where $\breve{\mathbf{u}}$ and $\breve{\sigma}$ represent the converged VP solution and $\alpha, \beta < 1$.

Momentum
The momentum equations become

$$\begin{aligned}\beta\frac{m}{\Delta t}\left(u^{k+1}-u^{k}\right) &= \overline{u}+\texttt{vrel}\left(-u^{k+1}\cos\theta+v^{k+1}\sin\theta\right)+mfv^{k+1}-\frac{m}{\Delta t}u^{k+1} \\ \beta\frac{m}{\Delta t}\left(v^{k+1}-v^{k}\right) &= \overline{v}-\texttt{vrel}\left(u^{k+1}\sin\theta+v^{k+1}\cos\theta\right)-mfu^{k+1}-\frac{m}{\Delta t}v^{k+1}\end{aligned}$$

where

$$\overline{u} = F_u+\tau_{ax}-mg\frac{\partial H_\circ}{\partial x}+\texttt{vrel}\left(U_w\cos\theta-V_w\sin\theta\right)+\frac{m}{\Delta t}u^\circ \qquad (80)$$

$$\overline{v} = F_v+\tau_{ay}-mg\frac{\partial H_\circ}{\partial y}+\texttt{vrel}\left(U_w\sin\theta+V_w\cos\theta\right)+\frac{m}{\Delta t}v^\circ, \qquad (81)$$

$\mathbf{u}^\circ$ is the initial value of velocity at the beginning of the subcycling ($k = 0$), and we use $\mathbf{u}^{k+1}$ for the ice-ocean stress and Coriolis terms. Eqs. (80) and (81) differ from Eqs. (60) and (61) only in the last term.

Solving simultaneously for $\mathbf{u}^{k+1}$ as before, we have

$$\begin{aligned}u^{k+1} &= \frac{\tilde{a}\tilde{u}+b\tilde{v}}{\tilde{a}^2+b^2} \\ v^{k+1} &= \frac{\tilde{a}\tilde{v}-b\tilde{u}}{\tilde{a}^2+b^2},\end{aligned}$$

where

$$\tilde{a} = (1+\beta)\frac{m}{\Delta t}+\texttt{vrel}\cos\theta \qquad (82)$$

$$\tilde{\mathbf{u}} = \overline{\mathbf{u}}+\beta\frac{m}{\Delta t}\mathbf{u}^k, \qquad (83)$$

and b is the same as in Eq. (63).

Stress
In CICE's classic approach, the update to σ_1 at subcycle step $k+1$ is

$$\sigma_1^{k+1} = \left(\sigma_1^k+\frac{P}{\Delta}\frac{\Delta t_e}{2T}\left(\dot\epsilon-\Delta\right)\right)*\left(1+\frac{\Delta t_e}{2T}\right) \qquad (84)$$

If we set

$$\alpha_1 = \frac{2T}{\Delta t_e},$$

then Eq. (84) becomes

$$\sigma_1^{k+1}\left(1+\alpha_1\right) = \alpha_1\sigma_1^k+\frac{P}{\Delta}\left(\dot\epsilon-\Delta\right). \qquad (85)$$

This is equivalent to Eq. (71) in [5], but using σ at the current subcycle $k+1$ in the last term on the right-hand side. Likewise, setting

$$\alpha_2 = \frac{2T}{e^2\Delta t_e} = \frac{\alpha_1}{e^2}$$

produces equations equivalent to Eq. (71) in [5] for σ_2 and σ_{12}. Therefore the only change needed in the stress code is to use α_1 and α_2 instead of $2T/\Delta t_e$ and $2T/e^2\Delta t_e$.

However, [5] introduce another change to the EVP stress equations by altering the form of Young's modulus in the elastic term: the coefficient of $\partial\sigma_1/\partial t$ is $1/E$, but it is e^2/E in the σ_2 and σ_{12} equations.

This change does not affect the VP equations to which the EVP equations should converge, but it does affect the transient path taken during the subcycling. Since EVP subcycling is finite, the numerical solutions obtained using this method differ from the original EVP code.

To implement this second change, we need define only $\alpha_1 = 2T/\Delta t_e$ as above and incorporate the factor of e^2 from α_2 into the equations for σ_2 and σ_{12}:

$$\sigma_1^{k+1}(1+\alpha_1) = \sigma_1^k + \alpha_1 \frac{P}{\Delta} D_D, \tag{86}$$

$$\sigma_2^{k+1}(1+\alpha_1) = \sigma_2^k + \frac{\alpha_1}{e^2}\frac{P}{\Delta} D_T, \tag{87}$$

$$\sigma_{12}^{k+1}(1+\alpha_1) = \sigma_{12}^k + \frac{\alpha_1}{2e^2}\frac{P}{\Delta} D_S. \tag{88}$$

To minimize code changes and unify the two approaches, we define and apply $1/\alpha_1$ and β in the classic EVP code, and modify the elastic stress term. These under-relaxation parameters control the rate at which the iteration converges. Thus for classic EVP we set

$$\begin{aligned} \texttt{arlx1i} &= \frac{1}{\alpha_1} = \frac{\Delta t_e}{2T} \\ \texttt{brlx} &= \beta = \frac{\Delta t}{\Delta t_e}. \end{aligned}$$

Then

$$\begin{aligned} \texttt{denom1} &= \frac{1}{1+\texttt{arlx1i}} = \frac{1}{1+1/\alpha_1} = \frac{1}{1+\Delta t_e/2T} \\ \texttt{c1} &= \frac{P}{\Delta}\texttt{arlx1i} = \frac{P}{\Delta}\frac{\Delta t_e}{2T} \\ \texttt{c0} &= \frac{\texttt{c1}}{e^2} = \frac{P}{\Delta}\frac{\Delta t_e}{2Te^2}. \end{aligned}$$

The stress equations for `stressp` (σ_1) are unchanged; the modified equations for `stressm` (σ_2) and `stress12` (σ_{12}) take the form

$$\begin{aligned} \texttt{stressm} &= \texttt{stressm} + \texttt{c0}\, D_T\, \texttt{denom1} \\ \texttt{stress12} &= \texttt{stress12} + 0.5\,\texttt{c0}\, D_S\, \texttt{denom1}. \end{aligned}$$

For classic EVP,

$$\texttt{cca} = a = \texttt{brlx}\,\frac{m}{\Delta t} + \texttt{vrel}\cos\theta = \frac{m}{\Delta t_e} + \texttt{vrel}\cos\theta.$$

For revised EVP, `arlx1i` and `brlx` are defined separately from Δt, Δt_e, T and e, and

$$\texttt{cca} = \tilde{a} = (1+\texttt{brlx})\,\frac{m}{\Delta t} + \texttt{vrel}\cos\theta = (1+\beta)\,\frac{m}{\Delta t} + \texttt{vrel}\cos\theta.$$

$\tilde{\mathbf{u}}$ must also be defined for revised EVP as in Eq. (83). The extra terms in $\tilde{a}$ and $\tilde{\mathbf{u}}$ are multiplied by a flag (`revp`) that equals 1 for revised EVP and 0 for classic EVP. Revised EVP is activated by setting the namelist parameter `revised_evp = .true.` Note that in the current implementation, only the modified version of the elastic term is available for either the classic (`revised_evp = .false.`) or the revised EVP method.

3.6 Thermodynamics

The thermodynamic sea ice model is based on [39] and [4], and is described more fully in [32]. For each thickness category the model 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 heat to simulate the effect of brine pocket melting and freezing.

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 $0^\circ\mathrm{C}$. 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.

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

$$S_{ik} = \frac{1}{2} S_{\max}[1 - \cos(\pi z^{(\frac{a}{z+b})})], \tag{89}$$

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 [46]. 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 [39]. Equation (89) 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.

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 $0^\circ\mathrm{C}$. 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. Since the salinity is prescribed, there is a one-to-one relationship between temperature and enthalpy. We can also define a snow enthalpy q_s, which depends on temperature alone. Expressions for the enthalpy are derived in Section 3.6.4.

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 $m + 1$. The calculation proceeds in two steps. First we solve a set of equations for the new temperatures, as discussed in Section 3.6.3. 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 Section 3.6.4. We begin by describing the surface forcing parameterizations, which are closely related to the ice and snow surface temperatures.

3.6.1 Melt ponds

Three explicit melt pond parameterizations are available in CICE, 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:

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

where

$$r = r_{min} + (r_{max} - r_{min})\, a_i$$

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 Section 3.1 and Fig. 1.

CESM formulation (`tr_pond_cesm = .true.`)

Melt pond area and thickness tracers are carried on each ice thickness category as in Section 3.1. 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,

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

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 ($h_p \leq 0.9 h_i$). 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 [22].

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 [15, 16, 17]. Thus, the topography of the ice cover plays a crucial role in determining the melt pond cover. However, CICE 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.

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 $i = 1$ up to $i = k + 1$ are covered in melt water. Figure 6a depicts the areas covered in melt water and saturated snow on the surface height (rather than thickness) categories $h_{top,k}$. Note in the CICE code, we assume that $h_{top,n}/h_{in} = 0.6$ (an arbitrary choice). The fractional area of the ith 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

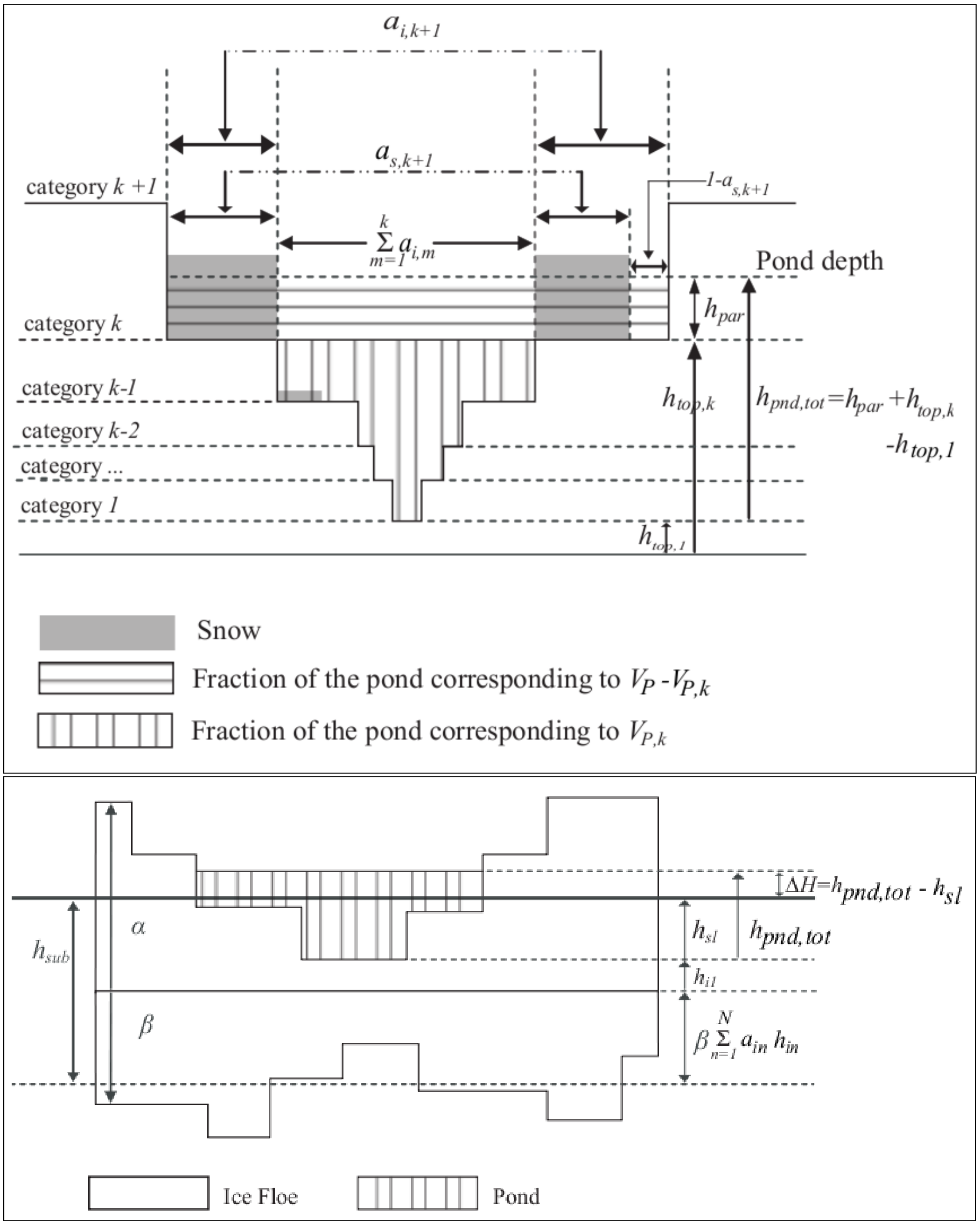

Figure 6: (a) Schematic illustration of the relationship between the height of the pond surface $h_{pnd,tot}$, the volume of water V_{Pk} required to completely fill up to category k, the volume of water $V_P - V_{Pk}$, and the depth to which this fills up category $k+1$. Ice and snow areas a_i and a_s 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.

that any extra melt water must occupy the second thickness category, and it is given by the expression

$$V_{P1} = a_{i1}(h_{top,2} - h_{top,1}) - a_{s1}a_{i1}h_{s1}(1 - V_{sw}), \tag{90}$$

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

$$V_{P2} = a_{i1}(h_{top,3} - h_{top,2}) + a_{i2}(h_{top,3} - h_{top,2}) - a_{s2}a_{i2}h_{s2}(1 - V_{sw}) + V_{P1}. \tag{91}$$

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

$$V_{Pk} = \sum_{m=0}^{k} a_{im}(h_{top,k+1} - h_{top,k}) - a_{sk}a_{ik}h_{sk}(1 - V_{sw}) + \sum_{m=0}^{k-1} V_{Pm}. \tag{92}$$

(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

$$h_{pnd,tot} = h_{par} + h_{top,k} - h_{top,1}, \tag{93}$$

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 6a 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 6a that for an incompletely to completely saturated snow cover on surface ice class $k+1$,

$$\begin{aligned} V_P - V_{Pk} &= h_{par}\left(\sum_{m=1}^{k} a_{ik} + a_{i,k+1}(1 - a_{s,k+1}) + a_{i,k+1}a_{s,k+1}V_{sw}\right) \\ &\quad \text{for} \quad h_{par} \leq h_{s,k+1}, \end{aligned} \tag{94}$$

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

$$\begin{aligned} V_P - V_{Pk} &= h_{par}\left(\sum_{m=1}^{k} a_{ik} + a_{i,k+1}(1 - a_{s,k+1})\right) + a_{i,k+1}a_{s,k+1}V_{sw}h_{s,k+1} \\ &\quad + a_{i,k+1}a_{s,k+1}(h_{par} - h_{s,k+1}) \\ &\quad \text{for} \quad h_{par} > h_{s,k+1}. \end{aligned} \tag{95}$$

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 [12]. 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. We model the vertical flushing rate using Darcy's law for flow through a porous medium

$$w = -\frac{\Pi_v}{\mu}\rho_o g \frac{\Delta H}{h_i}, \tag{96}$$

where w is the vertical mass flux per unit perpendicular crosssectional 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 [19] the vertical permeability of sea ice can be calculated from the liquid fraction ϕ:

$$\Pi_v = 3 \times 10^{-8} \phi^3 \mathrm{m}^2. \tag{97}$$

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 [13], the solid fraction is determined by:

$$\phi = \frac{c_i - c_{bulk}}{c_i - C(T)}, \tag{98}$$

where c_{bulk} is the bulk salinity of the ice (set to 3.2 ppt), $C(T)$ 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

$$h_{sl} = h_{sub} - 0.4 \sum_{n=1}^{N} a_{in} h_{in} - \beta h_{i1}, \tag{99}$$

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 6b depicts the relationship between the hydraulic head and the depths and heights that appear in equation (99). The depth of the submerged portion of the floe is determined from hydrostatic equilibrium to be

$$h_{sub} = \frac{\rho_m}{\rho_w} V_P + \frac{\rho_s}{\rho_w} V_s + \frac{\rho_i}{\rho_w} V_i, \tag{100}$$

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

$$\rho_i L \frac{dh_{ipnd}}{dt} = k_i \frac{\partial T_i}{\partial z} - k_p \frac{\partial T_p}{\partial z}, \tag{101}$$

where L 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 [53]. Approximating the temperature gradient in the ice lid as linear, the Stefan condition yields the classic Stefan solution for ice lid depth

$$h_{ipnd} = \sqrt{\frac{2k_i}{\rho_s L} \Delta T_i t}, \tag{102}$$

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 [16], the pond volume was carried as an ice area tracer, but in [17] and here, pond area and thickness are carried as separate tracers, as in Section 3.1.

Unlike the cesm and level-ice melt pond schemes, the liquid pond water in the topo parameterization is not virtual; it is withheld from being passed to the ocean model until the ponds drain. The refrozen pond lids are still virtual. Extra code needed to track and enforce conservation of water has been added to **ice_itd.F90** (subroutine *zap_small_areas*), **ice_mechred.F90** (subroutine *ridge_shift*), **ice_therm_itd.F90** (subroutines *linear_itd* and *lateral_melt*), and **ice_therm_vertical.F90** (subroutine *thermo_vertical*), along with global diagnostics in **ice_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 [28]. 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 meltwater 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

$$\frac{\partial}{\partial t}(a_{lvl}a_i) + \nabla \cdot (a_{lvl}a_i\mathbf{u}) = 0, \tag{103}$$

$$\frac{\partial}{\partial t}(a_{pnd}a_{lvl}a_i) + \nabla \cdot (a_{pnd}a_{lvl}a_i\mathbf{u}) = 0, \tag{104}$$

$$\frac{\partial}{\partial t}(h_{pnd}a_{pnd}a_{lvl}a_i) + \nabla \cdot (h_{pnd}a_{pnd}a_{lvl}a_i\mathbf{u}) = 0, \tag{105}$$

$$\frac{\partial}{\partial t}(h_{ipnd}a_{pnd}a_{lvl}a_i) + \nabla \cdot (h_{ipnd}a_{pnd}a_{lvl}a_i\mathbf{u}) = 0. \tag{106}$$

(We have dropped the category subscript here, for clarity.) Equations (105) and (106) 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 (Eq. 103), and pond fraction a_{pnd} is a tracer on level ice (Eq. 104).

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

$$\rho_i L \frac{\partial h_i}{\partial t} = k_i \frac{\partial T_i}{\partial z} \sim k_i \frac{\Delta T}{h_i} \tag{107}$$

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

$$\Delta h_i = \begin{cases} \sqrt{\beta \Delta t}/2 & \text{if } h_i = 0 \\ \beta \Delta t / 2h_i & \text{if } h_i > 0, \end{cases} \tag{108}$$

where

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

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

$$\Delta h_i = -\min\left(\frac{\max(F_\circ, 0)\Delta t}{\rho_i L}, h_i\right), \tag{109}$$

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

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

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 $h_{ps}^0 = F_{snow}\Delta t$ 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 CICE, but with its own parameter h_{s1}:

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

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 $a_{pnd}^{eff} = (1 - f_{frac})a_{pnd}a_{lvl}$, where (see Eq. 109)

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

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

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

then effectively there are no meltponds present, that is, $a_{pnd}^{eff} = h_{pnd}^{eff} = 0$. 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

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

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,

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

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

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

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

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

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,

$$\rho_i h_i + \rho_s h_s + \rho_0 h_p = \rho_w d \leq \rho_w h_i.$$

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 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 [41, 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 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

$$\Delta V_{perm} = -a_i d_p \min\left(h_{pnd}, \frac{pP\Delta t}{\mu h_i}\right),$$

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 **ice_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 **ice_therm_itd.F90** (subroutine *add_new_ice*) by maintaining the same mean pond area in a grid cell after the addition of new ice,

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

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

3.6.2 Thermodynamic surface forcing balance

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

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

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 default ("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 [6]. 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). See [6] for details; the CESM melt pond and Delta-Eddington parameterizations are further explained and validated in [22].

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 $T_{sf} < -1^\circ\mathrm{C}$ and $h_i >$ `ahmax`, the bare ice albedo is 0.78 for visible wavelengths ($<$ 700 nm) and 0.36 for near IR wavelengths ($>$ 700 nm). 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 $-1^\circ\mathrm{C}$ to $0^\circ\mathrm{C}$. The albedo of cold snow ($T_{sf} < -1^\circ\mathrm{C}$) 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 $-1^\circ\mathrm{C}$ to $0^\circ\mathrm{C}$. The total albedo is an area-weighted average of the ice and snow albedos, where the fractional snow-covered area is

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

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.

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:

$$I(z) = I_0 \exp(-\kappa_i z), \tag{110}$$

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 [11]. 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, $F_{L\uparrow} = \epsilon\sigma\left(T_{sf}^K\right)^4$, where $\epsilon = 0.95$ is the emissivity of snow or ice, σ is the Stefan-Boltzmann constant and T_{sf}^K 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 $F_{L\uparrow}$.)

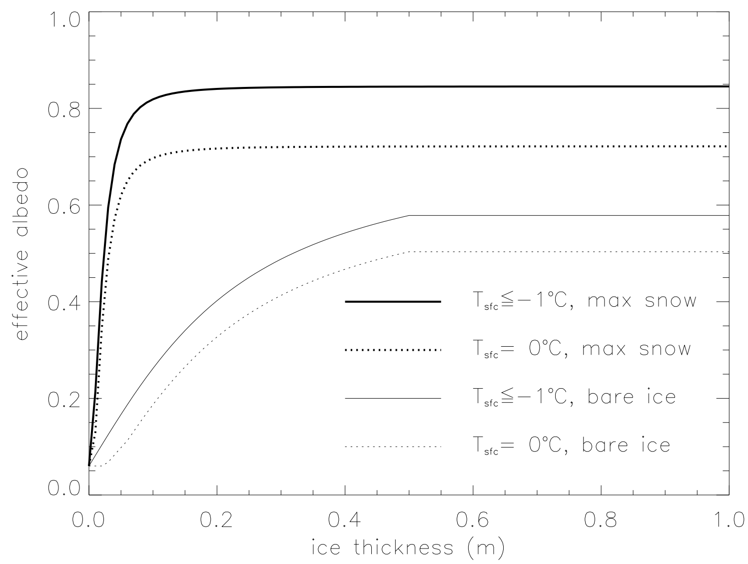

Figure 7: Albedo as a function of ice thickness and temperature, for the two extrema in snow depth, for the default (CCSM3) shortwave option. Maximum snow depth is computed based on Archimedes' Principle for the given ice thickness. These curves represent the envelope of possible albedo values.

The sensible heat flux is proportional to the difference between air potential temperature and the surface temperature of the snow or snow-free ice,

$$F_s = C_s \left(\Theta_a - T_{sf}^K \right).$$

C_s and C_l (below) are nonlinear turbulent heat transfer coefficients described in Section 2.1. Similarly, the latent heat flux is proportional to the difference between Q_a and the surface saturation specific humidity Q_{sf}:

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

where $q_1 = 1.16378 \times 10^7 \,\mathrm{kg/m^3}$, $q_2 = 5897.8\,\mathrm{K}$, T_{sf}^K 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 [37]:

$$F_{bot} = -\rho_w c_w c_h u_* (T_w - T_f), \tag{111}$$

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, $u_* = \sqrt{|\vec{\tau}_w| / \rho_w}$ is the friction velocity, and T_w is the sea surface temperature. A minimum value of u_* is available; we recommend $u_{*\,\mathrm{min}} = 5 \times 10^{-4}$ 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 [38] and [50]. 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).

3.6.3 New temperatures

An option for zero-layer thermodynamics [47] is available in this version of CICE by setting the namelist parameter `heat_capacity` to false and changing the number of ice layers, `nilyr`, in **ice_domain_size.F90** to 1. In the zero-layer case, the ice is fresh and the thermodynamic calculations are much simpler than in the default configuration, which we describe here.

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

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{112}$$

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) $T_{sf} < 0^\circ\mathrm{C}$, snow present; (2) $T_{sf} = 0^\circ\mathrm{C}$, snow present; (3) $T_{sf} < 0^\circ\mathrm{C}$, snow absent; and (4) $T_{sf} = 0^\circ\mathrm{C}$, 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 $0^\circ\mathrm{C}$, 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 [39]:

$$\rho_i c_i \frac{\partial T_i}{\partial t} = \frac{\partial}{\partial z}\left(K_i \frac{\partial T_i}{\partial z}\right) - \frac{\partial}{\partial z}[I_{pen}(z)], \tag{113}$$

where $\rho_i = 917\,\mathrm{kg/m^3}$ 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 [42]

$$c_i(T,S) = c_0 + \frac{L_0 \mu S}{T^2}, \tag{114}$$

where $c_0 = 2106$ J/kg/deg is the specific heat of fresh ice at $0^\circ\mathrm{C}$, $L_0 = 3.34 \times 10^5$ J/kg is the latent heat of fusion of fresh ice at $0^\circ\mathrm{C}$, and $\mu = 0.054$ deg/ppt is the ratio between the freezing temperature and salinity of brine.

Following [57] and [39], the standard thermal conductivity (`conduct`='MU71') is given by

$$K_i(T,S) = K_0 + \frac{\beta S}{T}, \tag{115}$$

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 [55] suggest that (115) 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 $0°$C, but does decrease near the top surface (regardless of temperature).

An alternative parameterization based on the "bubbly brine" model of [43] for conductivity is available (`conduct`='bubbly'):

$$K_i = \frac{\rho_i}{\rho_0}\left(2.11 - 0.011T + 0.09S/T\right), \tag{116}$$

where ρ_i and $\rho_0 = 917$ kg/m^3 are densities of sea ice and pure ice. Whereas the parameterization in (115) asymptotes to a constant conductivity of 2.03 W m^{-1} K^{-1} with decreasing T, K_i in (116) continues to increase with colder temperatures.

The equation for temperature changes in snow is analogous to (113), 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 **ice_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 (113) 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 (113). 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

$$c_{ik} = c_0 + \frac{L_0 \mu S_{ik}}{T_{ik}^m T_{ik}^{m+1}}, \tag{117}$$

which ensures that energy is conserved during a change in temperature. This can be shown by using (114) 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 (117). 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 (117) with each iteration until the solution converges.

Next consider the first term on the right-hand side of (113). The first term describes heat diffusion and is discretized for a given ice or snow layer k as

$$\frac{\partial}{\partial z}\left(K\frac{\partial T}{\partial z}\right) = \frac{1}{\Delta h}\left[K_k^*(T_{k-1}^{m+1} - T_k^{m+1}) - K_{k+1}^*(T_k^{m+1} - T_{k+1}^{m+1})\right], \tag{118}$$

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 (113) 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 $T_{sf} < 0^\circ$C we require

$$F_0 = F_{ct}, \tag{119}$$

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 (119) 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

$$\left[\left(\frac{dF_0}{dT_{sf}}\right)^* - K_1^*\right] T_{sf}^{m+1} + K_1^* T_1^{m+1} = \left(\frac{dF_0}{dT_{sf}}\right)^* T_{sf}^* - F_0^*, \tag{120}$$

the first equation in the set of equations (112). The temperature change in ice/snow layer k is

$$\rho_k c_k \frac{(T_k^{m+1} - T_k^m)}{\Delta t} = \frac{1}{\Delta h_k}[K_k^*(T_{k-1}^{m+1} - T_k^{m+1}) - K_{k+1}(T_k^{m+1} - T_{k+1}^{m+1})], \tag{121}$$

where $T_0 = T_{sf}$ in the equation for layer 1. In tridiagonal matrix form, (121) becomes

$$-\eta_k K_k T_{k-1}^{m+1} + \left[1 + \eta_k(K_k + K_{k+1})\right] T_k^{m+1} - \eta_k K_{k+1} T_{k+1}^{m+1} = T_k^m + \eta_k I_k, \tag{122}$$

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 $T_{sf} = 0^\circ$C, then there is no surface flux equation. In this case the first equation in (112) is similar to (122), 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 CICE (`calc_Tsfc` = false; see Section 2). In this case there is no surface flux equation. The top layer temperature is computed by an equation similar to (122) 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:

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

For thin layers and typical coupling intervals (~ 1 hr), K^* may need to be limited before being passed to the atmosphere via the coupler. Otherwise, the fluxes that are returned to CICE 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. $T_{sf} \leq 0°\mathrm{C}$.
2. The change in T_{sf} since the previous iteration is less than a prescribed limit, $\Delta T_{\max}$.
3. $F_0 \geq F_{ct}$. (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 $\Delta T_{\max} \approx 0.01°\mathrm{C}$ and $\Delta F_{max} \approx 0.01\ \mathrm{W/m^2}$—the calculation is complete.

3.6.4 Growth and melting

We first 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. The specific heat of sea ice, given by (114), 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 (114) 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 $\mathrm{J/kg}$ to $\mathrm{J/m^3}$ and adding a term for the energy needed to raise the meltwater temperature to $0°\mathrm{C}$, we obtain the sea ice enthalpy:

$$q_i(T, S) = -\rho_i\left[c_0(T_m - T) + L_0\left(1 - \frac{T_m}{T}\right) - c_w T_m.\right] \tag{123}$$

Note that (123) 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

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

The other root is unphysical.

Melting at the top surface is given by

$$q\,\delta h = \begin{cases} (F_0 - F_{ct})\,\Delta t & \text{if } F_0 > F_{ct} \\ 0 & \text{otherwise} \end{cases} \tag{124}$$

where q is the enthalpy of the surface ice or snow layer (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

$$q\,\delta h = (F_{cb} - F_{bot})\,\Delta t, \tag{125}$$

where F_{bot} is given by (111) and F_{cb} is the conductive heat flux at the bottom surface:

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

If ice is melting at the bottom surface, q in (125) 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}. This ice is added to the bottom layer.

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

$$(\rho L_v - q)\delta h = F_l \Delta t, \tag{126}$$

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 0°C. 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

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

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 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 (Section 3.3), which assumes that ice growth during a single time step is fairly small.

Lateral melting is accomplished by multiplying the state variables by $1-r_{side}$, where r_{side} is the fraction of ice melted laterally, and adjusting the ice energy and fluxes as appropriate.

4 Numerical implementation

CICE is written in FORTRAN90 and runs on platforms using UNIX, LINUX, and other operating systems. The code is parallelized via grid decomposition with MPI and includes some optimizations for vector architectures.

A second, "external" layer of parallelization involves message passing between CICE and the flux coupler, which may be running on different machines in a distributed system. The parallelization scheme for CICE was designed so that MPI could be used for the coupling along with either MPI or no parallelization internally. The internal parallelization method is set at compile time with the `NTASK` definition in the compile script. Message passing between the ice model and the CESM flux coupler is accomplished with MPI, regardless of the type of internal parallelization used for CICE, although the ice model may be coupled to another system without using MPI.

4.1 Directory structure

The present code distribution includes make files, several scripts and some input files. The main directory is **cice/**, and a run directory (**rundir/**) is created upon initial execution of the script **comp_ice**. One year of atmospheric forcing data is also available from the code distribution web site (see the **README** file for details).

cice/

README_v4.0 basic information

bld/ makefiles

Macros.⟨OS⟩.⟨SITE⟩.⟨machine⟩ macro definitions for the given operating system, used by **Makefile.⟨OS⟩**

Makefile.⟨OS⟩ primary makefile for the given operating system (⟨**std**⟩ works for most systems)

- **makedep.c** perl script that determines module dependencies

clean_ ice script that removes files from the compile directory

comp_ ice script that sets up the run directory and compiles the code

csm_ share/ modules based on “shared” code in CESM

- **shr_orb_mod.F90** orbital parameterizations

doc/ documentation

- **cicedoc.pdf** this document
- **PDF/** PDF documents of numerous publications related to CICE

drivers/ institution-specific modules

- **cice4/** official driver for CICE v.4 (LANL)
  - **CICE.F90** main program
  - **CICE.F90_debug** debugging version of **CICE.F90**
  - **CICE_FinalMod.F90** routines for finishing and exiting a run
  - **CICE_InitMod.F90** routines for initializing a run
  - **CICE_RunMod.F90** main driver routines for time stepping
  - **ice_constants.F90** physical and numerical constants and parameters
- **esmf/** Earth System Modeling Framework driver (www.esmf.ucar.edu)
  - **CICE.F90** main program
  - **CICE_ComponentMod.F90** subroutinized version of **CICE.F90** for direct coupling
  - **CICE_FinalMod.F90** routines for finishing and exiting a run
  - **CICE_InitMod.F90** routines for initializing a run
  - **CICE_RunMod.F90** main driver routines for time stepping
  - **ice_constants.F90** physical and numerical constants and parameters

ice.log.⟨OS⟩.⟨SITE⟩.⟨machine⟩ sample diagnostic output files

input_templates/ input files that may be modified for other CICE configurations

- **gx1/** ⟨1°⟩ displaced pole grid files
  - **global_gx1.grid** ⟨1°⟩ displaced pole grid (binary)
  - **global_gx1.kmt** ⟨1°⟩ land mask (binary)
  - **ice.restart_file** pointer for restart file name
  - **ice_in** namelist input data (data paths depend on particular system)
  - **iced_gx1_v4.0_kcatbound0** restart file used for initial condition
- **gx3/** ⟨3°⟩ displaced pole grid files
  - **global_gx3.grid** ⟨3°⟩ displaced pole grid (binary)
  - **global_gx3.kmt** ⟨3°⟩ land mask (binary)
  - **global_gx3.grid.nc** ⟨3°⟩ displaced pole grid (netCDF)
  - **global_gx3.kmt.nc** ⟨3°⟩ land mask (netCDF)

- **ice.restart_file** pointer for restart file name
- **ice_in** namelist input data (data paths depend on particular system)
- **iced_gx3_v4.0_kcatbound0** restart file used for initial condition

col/ column configuration files

- **ice_in** namelist input data (data paths depend on particular system)

run_ice.⟨OS⟩.⟨SITE⟩.⟨machine⟩ sample script for running on the given operating system

mpi/ modules that require MPI calls

- **ice_boundary.F90** boundary conditions
- **ice_broadcast.F90** routines for broadcasting data across processors
- **ice_communicate.F90** routines for communicating between processors
- **ice_exit.F90** aborts or exits the run
- **ice_gather_scatter.F90** gathers/scatters data to/from one processor from/to all processors
- **ice_global_reductions.F90** global sums, minvals, maxvals, etc., across processors
- **ice_timers.F90** timing routines

serial/ same modules as in **mpi/** but without MPI calls

source/ general CICE source code

- **ice_age.F90** handles most work associated with the age tracer
- **ice_atmo.F90** stability-based parameterization for calculation of turbulent ice-atmosphere fluxes
- **ice_blocks.F90** for decomposing global domain into blocks
- **ice_calendar.F90** keeps track of what time it is
- **ice_diagnostics.F90** miscellaneous diagnostic and debugging routines
- **ice_distribution.F90** for distributing blocks across processors
- **ice_domain.F90** decompositions, distributions and related parallel processing info
- **ice_domain_size.F90** domain and block sizes
- **ice_dyn_evp.F90** elastic-viscous-plastic dynamics component
- **ice_fileunits.F90** unit numbers for I/O
- **ice_flux.F90** fluxes needed/produced by the model
- **ice_forcing.F90** routines to read and interpolate forcing data for stand-alone ice model runs
- **ice_grid.F90** grid and land masks
- **ice_history.F90** netCDF or binary output routines
- **ice_init.F90** namelist and initializations
- **ice_itd.F90** utilities for managing ice thickness distribution
- **ice_kinds_mod.F90** basic definitions of reals, integers, etc.
- **ice_lvl.F90** handles most work associated with the level ice area and volume tracers
- **ice_mechred.F90** mechanical redistribution component (ridging)
- **ice_meltpond_cesm.F90** CESM melt pond parameterization

ice_meltpond_lvl.F90 level-ice melt pond parameterization

ice_meltpond_topo.F90 topo melt pond parameterization

ice_ocean.F90 mixed layer ocean model

ice_orbital.F90 orbital parameters for Delta-Eddington shortwave parameterization

ice_read_write.F90 utilities for reading and writing files

ice_restart.F90 read/write core restart file

ice_restoring.F90 basic restoring for open boundary conditions

ice_shortwave.F90 shortwave and albedo parameterizations

ice_spacecurve.F90 space-filling-curves distribution method

ice_state.F90 essential arrays to describe the state of the ice

ice_step_mod.F90 routines for time stepping the major code components

ice_therm_itd.F90 thermodynamic changes mostly related to ice thickness distribution

ice_therm_vertical.F90 vertical growth rates and fluxes

ice_transport_driver.F90 driver for horizontal advection

ice_transport_remap.F90 horizontal advection via incremental remapping

ice_work.F90 globally accessible work arrays

rundir/ execution or "run" directory created when the code is compiled using the **comp_ice** script (gx3)

cice code executable

compile/ directory containing object files, etc.

grid horizontal grid file from **cice/input_templates/gx3/**

ice.log.[ID] diagnostic output file

ice_in namelist input data from **cice/input_templates/gx3/**

hist/iceh.[timeID].nc output history file

kmt land mask file from **cice/input_templates/gx3/**

restart/ restart directory

iced_gx3_v4.0_kcatbound0 initial condition from **cice/input_templates/gx3/**

ice.restart_file restart pointer from **cice/input_templates/gx3/**

run_ice batch run script file from **cice/input_templates/**

4.2 Grid, boundary conditions and masks

The spatial discretization is specialized for a generalized orthogonal B-grid as in [40] or [49]. The ice and snow area, volume and energy are given at the center of the cell, velocity is defined at the corners, and the internal ice stress tensor takes four different values within a grid cell; bilinear approximations are used for the stress tensor and the ice velocity across the cell, as described in [26]. This tends to avoid the grid decoupling problems associated with the B-grid. EVP is available on the C-grid through the MITgcm code distribution, http://mitgcm.org/cgi-bin/viewcvs.cgi/MITgcm/pkg/seaice

Since ice thickness and thermodynamic variables such as temperature are given in the center of each cell, the grid cells are referred to as "T cells." We also occasionally refer to "U cells," which are centered

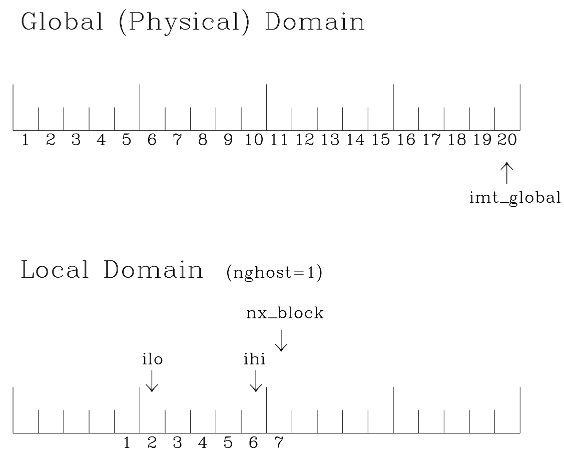

Figure 8: Grid parameters for a sample one-dimensional, 20-cell global domain decomposed into four local subdomains. Each local domain has one ghost cell on each side, and the physical portion of the local domains are labeled `ilo:ihi`. The parameter `nx_block` is the total number of cells in the local domain, including ghost cells, and the same numbering system is applied to each of the four subdomains.

on the northeast corner of the corresponding T cells and have velocity in the center of each. The velocity components are aligned along grid lines.

The user has several choices of grid routines: *popgrid* reads grid lengths and other parameters for a nonuniform grid (including tripole grids), and *rectgrid* creates a regular rectangular grid, including that used for the column configuration. The input files **global_gx3.grid** and **global_gx3.kmt** contain the ⟨3°⟩ POP grid and land mask; **global_gx1.grid** and **global_gx1.kmt** contain the ⟨1°⟩ grid and land mask. These are binary unformatted, direct access files produced on an SGI (Big Endian). If you are using an incompatible (Little Endian) architecture, choose `rectangular` instead of `displaced_pole` in **ice_in**, or follow procedures as for conejo (⟨**OS**⟩.⟨**SITE**⟩.⟨**machine**⟩ = `Linux.LANL.conejo`). There are netCDF versions of the gx3 grid files available.

4.2.1 Grid domains and blocks

In general, the global gridded domain is `nx_global`×`ny_global`, while the subdomains used in the block distribution are `nx_block`×`ny_block`. The physical portion of a subdomain is indexed as [`ilo:ihi, jlo:jhi`], with `nghost` "ghost" or "halo" cells outside the domain used for boundary conditions. These parameters are illustrated in Figure 8 in one dimension. The routines *global_scatter* and *global_gather* distribute information from the global domain to the local domains and back, respectively. If MPI is not being used for grid decomposition in the ice model, these routines simply adjust the indexing on the global domain to the single, local domain index coordinates. Although we recommend that the user choose the local domains so that the global domain is evenly divided, if this is not possible then the furthest east and/or north blocks will contain nonphysical points ("padding"). These points are excluded from the computation domain and have little effect on model performance.

The user chooses a block size `BLCKX`×`BLCKY` and the number of processors `NTASK` in **comp_ice**. Parameters in the *domain_nml* namelist in **ice_in** determine how the blocks are distributed across the processors, and how the processors are distributed across the grid domain. Recommended combinations of these parameters for best performance are given in Section 4.7. The script **comp_in** computes the maximum number of blocks on each processor for typical Cartesian distributions, but for non-Cartesian cases `MXBLCKS` may need to be set in the script. The code will print this information to the log file before aborting, and the user will need to adjust MXBLCKS in **comp_ice** and recompile. The code will also print a warning if the maximum number of blocks is too large. Although this is not fatal, it does require excess memory.

A loop at the end of routine *create_blocks* in module **ice_blocks.F90** will print the locations for all of the blocks on the global grid if dbug is set to be true. Likewise, a similar loop at the end of routine *create_local_block_ids* in module **ice_distribution.F90** will print the processor and local block number for each block. With this information, the grid decomposition into processors and blocks can be ascertained. The dbug flag must be manually set in the code in each case (independently of the dbug flag in **ice_in**), as there may be hundreds or thousands of blocks to print and this information should be needed only rarely. This information is much easier to look at using a debugger such as Totalview.

4.2.2 Tripole grids

The tripole grid is a device for constructing a global grid with a normal south pole and southern boundary condition, which avoids placing a physical boundary or grid singularity in the Arctic Ocean. Instead of a single north pole, it has two "poles" in the north, both located on land, with a line of grid points between them. This line of points is called the "fold," and it is the "top row" of the physical grid. One pole is at the left-hand end of the top row, and the other is in the middle of the row. The grid is constructed by "folding" the top row, so that the left-hand half and the right-hand half of it coincide. Two choices for constructing the tripole grid are available. The one first introduced to CICE is called "U-fold", which means that the poles and the grid cells between them are U cells on the grid. Alternatively the poles and the cells between them can be grid T cells, making a "T-fold." Both of these options are also supported by the OPA/NEMO ocean model, which calls the U-fold an "f-fold" (because it uses the Arakawa C-grid in which U cells are on T-rows). The choice of tripole grid is given by the namelist variable ns_boundary_type, "tripole" for the U-fold and "tripoleT" for the T-fold grid.

In the U-fold tripole grid, the poles have U-index $\mathtt{nx\_global}/2$ and nx_global on the top U-row of the physical grid, and points with U-index i and $\mathtt{nx\_global} - \mathtt{i}$ are coincident. Let the fold have U-row index n on the global grid; this will also be the T-row index of the T-row to the south of the fold. There are ghost T- and U-rows to the north, beyond the fold, on the logical grid. The point with index i along the ghost T-row of index $n+1$ physically coincides with point $\mathtt{nx\_global} - \mathtt{i} + 1$ on the T-row of index n. The ghost U-row of index $n+1$ physically coincides with the U-row of index $n-1$.

In the T-fold tripole grid, the poles have T-index 1 and and $\mathtt{nx\_global}/2 + 1$ on the top T-row of the physical grid, and points with T-index i and $\mathtt{nx\_global} - \mathtt{i} + 2$ are coincident. Let the fold have T-row index n on the global grid. It is usual for the northernmost row of the physical domain to be a U-row, but in the case of the T-fold, the U-row of index n is "beyond" the fold; although it is not a ghost row, it is not physically independent, because it coincides with U-row $n-1$, and it therefore has to be treated like a ghost row. Points i on U-row n coincides with $\mathtt{nx\_global} - \mathtt{i} + 1$ on U-row $n-1$. There are still ghost T- and U-rows $n+1$ to the north of U-row n. Ghost T-row $n+1$ coincides with T-row $n-1$, and ghost U-row $n+1$ coincides with U-row $n-2$.

The tripole grid thus requires two special kinds of treatment for certain rows, arranged by the halo-update routines. First, within rows along the fold, coincident points must always have the same value. This is achieved by averaging them in pairs. Second, values for ghost rows and the "quasi-ghost" U-row on the T-fold grid are reflected copies of the coincident physical rows. Both operations involve the tripole buffer, which is used to assemble the data for the affected rows. Special treatment is also required in the scattering routine, and when computing global sums one of each pair of coincident points has to be excluded.

4.2.3 Column configuration

A column modeling capability is available. Because of the boundary conditions and other spatial assumptions in the model, this is not a single column, but a small array of columns (minimum grid size is 5x5). However, the code is set up so that only the single, central column is used (all other columns are designated as land). 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. The grid type is rectangular, dynamics are turned off (`kdyn=0`) and one processor is used.

History variables available for column output are ice and snow temperature, `Tinz` and `Tsnz`. These variables also include thickness category as a fourth dimension.

4.2.4 Boundary conditions

Open boundary conditions are the default in CICE; the physical domain can still be closed using the land mask. In our bipolar, displaced-pole grids, one row of grid cells along the north and south boundaries is located on land, and along east/west domain boundaries not masked by land, periodic conditions wrap the domain around the globe. CICE can be run on regional grids with open boundary conditions. Except for variables describing grid lengths, non-land ghost cells along the grid edge must be filled by restoring them to specified values. The namelist variable `restore_ice` turns this functionality on and off and currently uses the restoring timescale `trestore` (also used for restoring ocean sea surface temperature in stand-alone ice runs). This implementation is only intended to provide the "hooks" for a more sophisticated treatment; the rectangular grid option can be used to test this configuration. For exact restarts using restoring, set `restart_ext` = true in namelist to use the extended-grid subroutines.

Much of the infrastructure used in CICE, including the boundary routines, is adopted from POP. The boundary routines perform boundary communications among processors when MPI is in use and among blocks whenever there is more than one block per processor.

4.2.5 Masks

A land mask `hm` (M_h) is specified in the cell centers, with 0 representing land and 1 representing ocean cells. A corresponding mask `uvm` (M_u) for velocity and other corner quantities is given by

$$M_u(i,j) = \min\{M_h(l),\, l = (i,j),\, (i+1,j),\, (i,j+1),\, (i+1,j+1)\}.$$

The logical masks `tmask` and `umask` (which correspond to the real masks `hm` and `uvm`, respectively) are useful in conditional statements.

In addition to the land masks, two other masks are implemented in *evp_prep* in order to reduce the dynamics component's work on a global grid. At each time step the logical masks `ice_tmask` and `ice_umask` are determined from the current ice extent, such that they have the value "true" wherever ice exists. They also include a border of cells around the ice pack for numerical purposes. These masks are used in the dynamics component to prevent unnecessary calculations on grid points where there is no ice. They are not used in the thermodynamics component, so that ice may form in previously ice-free cells. Like the land masks `hm` and `uvm`, the ice extent masks `ice_tmask` and `ice_umask` are for T cells and U cells, respectively.

Two additional masks are created for the user's convenience: `lmask_n` and `lmask_s` can be used to compute or write data only for the northern or southern hemispheres, respectively. Special constants (`spval` and `spval_dbl`, each equal to 10^{30}) are used to indicate land points in the history files and diagnostics.

`ice_ic`	`runtype/restart`		
	initial/false	initial/true	continue/true (or false[a])
none	no ice	no ice[b]	restart using **pointer_file**
default	SST/latitude dependent	SST/latitude dependent[b]	restart using **pointer_file**
filename	no ice[c]	start from **filename**	restart using **pointer_file**

Table 4: Ice initial state resulting from combinations of `ice_ic`, `runtype` and `restart`. [a]If false, `restart` is reset to true. [b]`restart` is reset to false. [c]`ice_ic` is reset to 'none.'

4.3 Initialization and coupling

The ice model's parameters and variables are initialized in several steps. Many constants and physical parameters are set in **ice_constants.F90**. Namelist variables (Table 7), whose values can be altered at run time, are handled in *input_data* and other initialization routines. These variables are given default values in the code, which may then be changed when the input file **ice_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, some variables are read and reinitialized in *restartfile*. Finally, albedo and other quantities dependent on the initial ice state are set. Some of these parameters will be described in more detail in Table 7.

Three namelist variables control model initialization, `ice_ic`, `runtype`, and `restart`, as described in Table 4. It is possible to do an initial run from a file **filename** in two ways: (1) set `runtype` = 'initial', `restart` = true and `ice_ic` = **filename**, or (2) `runtype` = 'continue' and `pointer_file` = **./restart/ice.restart_file** where **./restart/ice.restart_file** contains the line "./restart/filename". The first option is convenient when repeatedly starting from a given file when subsequent restart files have been written.

MPI is initialized in *init_communicate* for both coupled and stand-alone MPI runs. The ice component communicates with a flux coupler or other climate components via external routines that handle the variables listed in Table 1. For stand-alone runs, routines in **ice_forcing.F90** read and interpolate data from files, and are intended merely to provide guidance for the user to write his or her own routines. Whether the code is to be run in stand-alone or coupled mode is determined at compile time, as described below.

4.4 Choosing an appropriate time step

The time step is chosen based on stability of the transport component (both horizontal and in thickness space) and on resolution of the physical forcing. CICE allows the dynamics, advection and ridging portion of the code to be run with a shorter timestep, Δt_{dyn} (`dt_dyn`), than the thermodynamics timestep Δt (`dt`). In this case, `dt` and the integer `ndyn_dt` are specified, and `dt_dyn = dt/ndyn_dt`.

A conservative estimate of the horizontal transport time step bound, or CFL condition, under remapping yields

$$\Delta t_{dyn} < \frac{\min\left(\Delta x, \Delta y\right)}{2\max\left(u, v\right)}.$$

Numerical estimates for this bound for several POP grids, assuming $\max(u,v) = 0.5$ m/s, are as follows:

grid label	N pole singularity	dimensions	min $\sqrt{\Delta x \cdot \Delta y}$	max Δt_{dyn}
gx3	Greenland	100×116	39×10^3 m	10.8 hr
gx1	Greenland	320×384	18×10^3 m	5.0 hr
p4	Canada	900×600	6.5×10^3 m	1.8 hr

As discussed in section 3.4 and [35], the maximum time step in practice is usually determined by the time scale for large changes in the ice strength (which depends in part on wind strength). Using the strength parameterization of [44], as in eq. 55, limits the time step to $\sim$30 minutes for the old ridging scheme, and to $\sim$2 hours for the new scheme, assuming Δx = 10 km. Practical limits may be somewhat less, depending on the strength of the atmospheric winds.

Transport in thickness space imposes a similar 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.

The dynamics component is subcycled `ndte` (N) times per dynamics time step so that the elastic waves essentially disappear before the next time step. The subcycling time step (Δt_e) is thus

```
dte = dt_dyn/ndte.
```

A second parameter, $E_\circ$ (`eyc`), must be selected, which defines the elastic wave damping timescale T, described in Section 3.5, as `eyc*dt_dyn`. The forcing terms are not updated during the subcycling. Given the small step (`dte`) at which the EVP dynamics model is subcycled, the elastic parameter E is also limited by stability constraints, as discussed in [25]. Linear stability analysis for the dynamics component shows that the numerical method is stable as long as the subcycling time step Δt_e sufficiently resolves the damping timescale T. For the stability analysis we had to make several simplifications of the problem; hence the location of the boundary between stable and unstable regions is merely an estimate. In practice, the ratio $\Delta t_e \; : \; T \; : \; \Delta t$ = 1 : 40 : 120 provides both stability and acceptable efficiency for time steps (Δt) on the order of 1 hour.

Note that only T and Δt_e figure into the stability of the dynamics component; Δt does not. Although the time step may not be tightly limited by stability considerations, large time steps (*eg.,* $\Delta t = 1$ day, given daily forcing) do not produce accurate results in the dynamics component. The reasons for this error are discussed in [25]; see [29] for its practical effects. The thermodynamics component is stable for any time step, as long as the surface temperature T_{sfc} is computed internally.

4.5 Model output

4.5.1 History files

Model output data is averaged over the period(s) given by `histfreq` and `histfreq_n`, and written to binary or netCDF files prepended by `history_file` in **ice_in**. That is, if `history_file = 'iceh'` then the filenames will have the form **iceh.[timeID].nc** or **iceh.[timeID].da**, depending on the output file format (`history_format`) chosen. The netCDF history files are CF-compliant; header information for data contained in the netCDF files is displayed with the command `ncdump -h filename.nc`. With binary files, a separate header file is written with equivalent information. Standard fields are output according to settings in the **icefields_nml** namelist in **ice_in**.The user may add (or subtract) variables not already available in the namelist by following the instructions in section 4.8.2.

The history module allows output at different frequencies. Five output frequencies (`1`, `h`, `d`, `m`, `y`) are available simultaneously during a run. The same variable can be output at different frequencies (say daily and monthly) via its namelist flag, `f_`$\langle var \rangle$, which is now a character string corresponding to `histfreq` or '`x`' for none. (Grid variable flags are still logicals, since they are written to all files, no matter what the frequency is.) If there are no namelist flags with a given `histfreq` value, or if an element of `histfreq_n` is 0, then no file will be written at that frequency. The output period can be discerned from the filenames.

For example, in namelist:

```
histfreq = '1', 'h', 'd', 'm', 'y'
```

```
histfreq_n = 1, 6, 0, 1, 1
f_hi = '1'
f_hs = 'h'
f_Tsfc = 'd'
f_aice = 'm'
f_meltb = 'mh'
f_iage = 'x'
```

Here, `hi` will be written to a file on every timestep, `hs` will be written once every 6 hours, `aice` once a month, `meltb` once a month AND once every 6 hours, and `Tsfc` and `iage` will not be written.

From an efficiency standpoint, it is best to set unused frequencies in histfreq to '`x`'. Having output at all 5 frequencies takes nearly 5 times as long as for a single frequency. If you only want monthly output, the most efficient setting is `histfreq='m','x','x','x','x'`. The code counts the number of desired streams (`nstreams`) based on `histfreq`.

The history variable names must be unique for netcdf, so in cases where a variable is written at more than one frequency, the variable name is appended with the frequency in files after the first one. In the example above, `meltb` is called `meltb` in the monthly file (for backward compatibility with the default configuration) and `meltb_h` in the 6-hourly file.

Using the same frequency twice in `histfreq` will have unexpected consequences and currently will cause the code to abort. It is not possible at the moment to output averages once a month and also once every 3 months, for example.

If `write_ic` is set to T in **ice_in**, a snapshot of the same set of history fields at the start of the run will be written to the history directory in **iceh_ic.[timeID].nc(da)**. Nine history variables are hard-coded for instantaneous output regardless of the averaging flag, at the frequency given by their namelist flag.

The normalized principal components of internal ice stress are computed in *principal_stress* and written to the history file. This calculation is not necessary for the simulation; principal stresses are merely computed for diagnostic purposes and included here for the user's convenience.

Several history variables are available in two forms, a value representing an average over the sea ice fraction of the grid cell, and another that is multiplied by a_i, representing an average over the grid cell area. Our naming convention attaches the suffix "_ai" to the grid-cell-mean variable names.

4.5.2 Diagnostic files

Like `histfreq`, the parameters `diagfreq` and `diagfreq_n` can be used to regulate how often output is written to a log file. The log file unit to which diagnostic output is written is set in **ice_fileunits.F90**. If `diag_type = 'stdout'`, then it is written to standard out (or to **ice.log.[ID]** if you redirect standard out as in **run_ice**); otherwise it is written to the file given by `diag_file`. In addition to the standard diagnostic output (maximum area-averaged thickness, velocity, average albedo, total ice area, and total ice and snow volumes), the namelist options `print_points` and `print_global` cause additional diagnostic information to be computed and written. `print_global` outputs global sums that are useful for checking global conservation of mass and energy. `print_points` writes data for two specific grid points. Currently, one point is near the North Pole and the other is in the Weddell Sea; these may be changed in **ice_diagnostics.F90**.

Timers are declared and initialized in **ice_timers.F90**, and the code to be timed is wrapped with calls to *ice_timer_start* and *ice_timer_stop*. Finally, *ice_timer_print* writes the results to the log file. The optional "stats" argument (true/false) prints additional statistics. Calling *ice_timer_print_all* prints all of the timings at once, rather than having to call each individually. Currently, the timers are set up as in Table 5. Section 4.8.1 contains instructions for adding timers.

The timings provided by these timers are not mutually exclusive. For example, the column timer (5)

Timer Index	Label	
1	Total	the entire run
2	Step	total minus initialization and exit
3	Dynamics	EVP
4	Advection	horizontal transport
5	Column	all vertical (column) processes
6	Thermo	vertical thermodynamics
7	Shortwave	SW radiation and albedo
8	Meltponds	melt ponds
9	Ridging	mechanical redistribution
10	Cat Conv	transport in thickness space
11	Coupling	sending/receiving coupler messages
12	ReadWrite	reading/writing files
13	Diags	diagnostics (log file)
14	History	history output
15	Bound	boundary conditions and subdomain communications

Table 5: CICE timers.

includes the timings from 6, 7, 8 and 9, and subroutine *bound* (timer 14) is called from many different places in the code, including the dynamics and advection routines.

The timers use *MPI_WTIME* for parallel runs and the F90 intrinsic *system_clock* for single-processor runs.

4.5.3 Restart files

A binary unformatted file is created that contains all of the data that CICE needs for a full restart. The filename begins with the character string `dumpfile`, and the restart dump frequency is given by `dumpfreq` and `dumpfreq_n`. The pointer to the filename from which the restart data is to be read for a continuation run is set in `pointer_file`.

Routines for gathering, scattering and (unformatted) reading and writing of the "extended" global grid, including the physical domain and ghost cells around the outer edges, allow exact restarts on regional grids with open boundary conditions, and they will also simplify restarts on the various tripole grids. They are accessed by setting `restart_ext` = true in namelist.

4.6 Execution procedures

To compile and execute the code: in the source directory,

1. Download the forcing data used for testing from the CICE website, http://climate.lanl.gov/Models/CICE/.
2. Create **Macros.*** and **run_ice.*** files for your particular platform, if they do not already exist (type 'uname -s' at the prompt to get ⟨OS⟩).
3. Alter directories in the script **comp_ice**.
4. Run **comp_ice** to set up the run directory and make the executable '**cice**'.

variable	options	description
RES	col, gx3, gx1	grid resolution
NTASK	(integer)	total number of processors
BLCKX	(integer)	number of grid cells on each block in the x-direction†
BLCKY	(integer)	number of grid cells on each block in the y-direction†
MXBLCKS	(integer)	maximum number of blocks per processor
NICELYR	(integer)	number of vertical layers in the ice
NSNWLYR	(integer)	number of vertical layers in the snow
NICECAT	(integer)	number of ice thickness categories
TRAGE	0 or 1	set to 1 for ice age tracer
TRLVL	0 or 1	set to 1 for level and deformed ice tracers
TRPND	0 or 1	set to 1 for melt pond tracers
NTRAERO	(integer)	number of aerosol tracers
USE_ESMF	yes/no	for coupling with the Earth System Modeling Framework
CAM_ICE	yes/no	for single-column CAM runs
NETCDF	yes/no	use 'no' if netCDF library is unavailable
DITTO	yes/no	for reproducible diagnostics

† Does not include ghost cells.

Table 6: Configuration options available in **comp_ice**.

5. To clean the compile directory and start fresh, alter directories in the script **clean_ice** and execute the script.

In the run directory,

1. Alter `atm_data_dir` and `ocn_data_dir` in the namelist file **ice_in**.
2. Alter the script **run_ice** for your system.
3. Execute **run_ice**.

If this fails, see Section 5.1.

This procedure creates the output log file **ice.log.[ID]**, and if `npt` is long enough compared with `dumpfreq` and `histfreq`, dump files **iced.[timeID]** and netCDF (or binary) history output files **iceh_[timeID].nc (.da)**. Using the $\langle 3^\circ \rangle$ grid, the log file should be similar to **ice.log.⟨OS⟩**, provided for the user's convenience. These log files were created using MPI on 4 processors on the $\langle 3^\circ \rangle$ grid.

Several options are available in **comp_ice** for configuring the run, shown in Table 6. If `NTASK` = 1, then the **serial/** code is used, otherwise the code in **mpi/** is used. Note that the value of `NTASK` in **comp_ice** must equal the value of `nprocs` in **ice_in**. Generally the value of `MXBLCKS` computed by **comp_ice** is sufficient, but sometimes it will need to be set explicitly, as discussed in Section 4.7. To conserve memory, match the tracer requests in **comp_ice** with those in **ice_in**. CESM uses 3 aerosol tracers; the number given in **comp_ice** must be less than or equal to the maximum allowed in **ice_domain_size.F90**.

The scripts define a number of environment variables, mostly as directories that you will need to edit for your own environment. `$SYSTEM_USERDIR`, which on machines at Oak Ridge National Laboratory points automatically to scratch space, is intended to be a disk where the run directory resides. `SHRDIR` is a path to the CESM shared code.

The ‘reproducible’ option (DITTO) makes diagnostics bit-for-bit when varying the number of processors. (The simulation results are bit-for-bit regardless, because they do not require global sums or max/mins as do the diagnostics.) This was done mainly by increasing the precision for the global reduction calculations, except for regular double-precision (r8) calculations involving MPI; MPI can not handle MPI_REAL16 on some architectures. Instead, these cases perform sums or max/min calculations across the global block structure, so that the results are bit-for-bit as long as the block distribution is the same (the number of processors can be different).

CICE namelist variables available for changes after compile time appear in **ice.log.*** with values read from the file **ice_in**; their definitions are given in Section 5.6. For example, to run for a different length of time, say three days, set `npt = 72` in **ice_in**. At present, the user supplies the time step `dt`, the number of dynamics/advection/ridging subcycles `ndyn_dt`, and the number of evp subcycles `ndte`, and `dte` is then calculated in subroutine *init_evp*. The primary reason for doing it this way is to ensure that `ndte` is an integer.

To restart from a previous run, set `restart = .true.` in **ice_in**. There are two ways of restarting from a given file. The restart pointer file **ice.restart_file** (created by the previous run) contains the name of the last written data file (**iced.[timeID]**). Alternatively, a filename can be assigned to `ice_ic` in **ice_in**. Consult Section 4.3 for more details. Restarts are exact for MPI or single processor runs.

4.7 Performance

Namelist options (*domain_nml*) provide considerable flexibility for finding the most efficient processor and block configuration. Some of these choices are illustration in Figure 9. `processor_shape` chooses between tall, thin processor domains (`slenderX1` or `slenderX2`, often better for sea ice simulations on global grids where nearly all of the work is at the top and bottom of the grid with little to do in between) and close-to-square domains, which maximize the volume to surface ratio (and therefore on-processor computations to message passing, if there were ice in every grid cell). In cases where the number of processors is not a perfect square (4, 9, 16...), the `processor_shape` namelist variable allows the user to choose how the processors are arranged. Here again, it is better in the sea ice model to have more processors in x than in y, for example, 8 processors arranged 4x2 (`square-ice`) rather than 2x4 (`square-pop`). The latter option is offered for direct-communication compatibility with POP, in which this is the default.

The `distribution_type` options allow standard Cartesian distribution of blocks, redistribution via a ‘rake’ algorithm for improved load balancing across processors, and redistribution based on space-filling curves. The rake and space-filling curve algorithms are primarily helpful when using squarish processor domains where some processors (located near the equator) would otherwise have little work to do. Processor domains need not be rectangular, however.

`distribution_wght` chooses how the work-per-block estimates are weighted. The ‘block’ option is the default in POP, which uses a lot of array syntax requiring calculations over entire blocks (whether or not land is present), and is provided here for direct-communication compatibility with POP. The ‘latitude’ option weights the blocks based on latitude and the number of ocean grid cells they contain.

The rake distribution type is initialized as a standard, Cartesian distribution. Using the work-per-block estimates, blocks are “raked” onto neighboring processors as needed to improve load balancing characteristics among processors, first in the x direction and then in y.

Space-filling curves reduce a multi-dimensional space (2D, in our case) to one dimension. The curve is composed of a string of blocks that is snipped into sections, again based on the work per processor, and each piece is placed on a processor for optimal load balancing. This option requires that the block size be chosen such that the number of blocks in the x direction equals the number of blocks in the y direction, and that number must be factorable as $2^n 3^m 5^p$ where n, m, p are integers. For example, a 16x16 array of blocks, each containing 20x24 grid cells, fills the gx1 grid ($n = 4, m = p = 0$). If either of these conditions is not

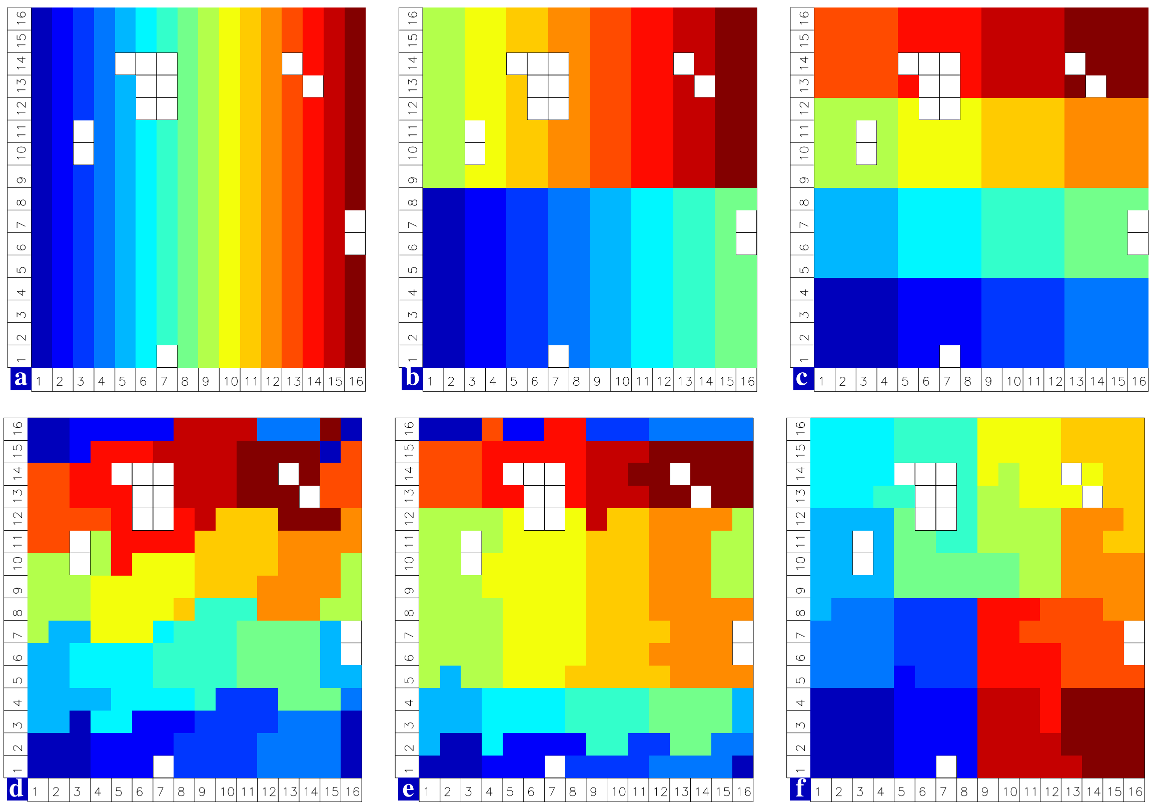

Figure 9: Distribution of 256 blocks across 16 processors, represented by colors, on the gx1 grid: (a) cartesian, slenderX1, (b) cartesian, slenderX2, (c) cartesian, square-ice (square-pop is equivalent here), (d) rake with block weighting, (e) rake with latitude weighting, (f) spacecurve. Each block consists of 20x24 grid cells, and white blocks consist entirely of land cells.

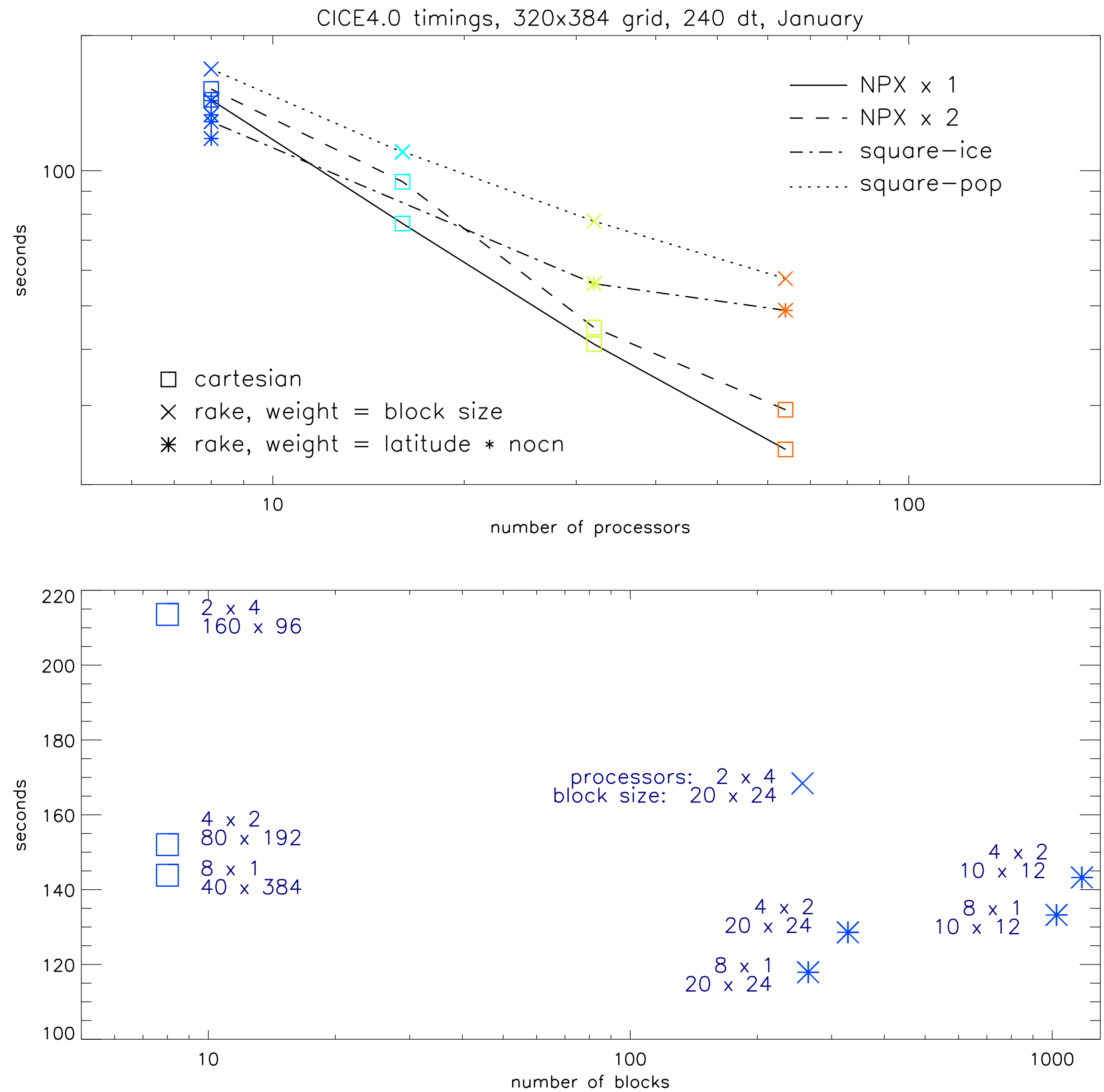

Figure 10: (a) CICE 4.0 timings for 8, 16, 32, and 64 processors of the Linux cluster "coyote," after 240 1-hr time steps on a 320x384 displaced pole grid. (b) 8-processor timings for various processor shapes (indicated by the upper pair of numbers; eg., 8x1 = 'slenderX1') and block sizes.

met, a Cartesian distribution is used instead.

The user provides the total number of processors and the block dimensions in the setup script (**comp_ice**). When moving toward smaller, more numerous blocks, there is a point where the code becomes less efficient; blocks should not have fewer than about 20 grid cells in each direction. Squarish blocks are probably better, again to optimize the volume-to-surface ratio for communications.

In general, the following rules-of-thumb seem to work best:

- For large numbers of processors: `processor_shape` = 'slenderX1' with one block per processor (`distribution_type` and `distribution_wght` do not matter—they default to cartesian). 'slenderX2' is not quite as efficient for the same number of processors but is better than either square option, and is sometimes necessary to keep the processor domain width sufficiently large.
- For small numbers of processors: `distribution_type` = 'rake' or 'spacecurve' with `distribution_wght` = 'latitude' and `processor_shape` = 'square-ice'.
- The cross-over point, i.e. the number of processors where these choices result in about the same timings, will vary depending on the grid and the processor/architecture.

Figure 10 shows the model's scaling characteristics for up to 64 processors on the Linux cluster "coyote." These runs were made on the gx1 global grid (320x384, approximately 1 degree resolution), from the restart file included with the code distribution. On the gx1 grid, the scaling is nearly linear up to 64 processors for "slender" (x1 or x2) decompositions. More than 64 processors do not have enough ice work to do on this grid, and the timings do not scale as well. The lower plot shows timings for 8-processor block distributions versus the total number of blocks, illustrating the performance gains that are available by redistributing smaller blocks across processors.

Throughout the code, (i, j) loops have been combined into a single loop, often over just ocean cells or those containing sea ice. This was done to reduce unnecessary operations and to improve vector performance.

4.8 Adding things

4.8.1 Timers

Timing any section of code, or multiple sections, consists of defining the timer and then wrapping the code with start and stop commands for that timer. Printing of the timer output is done simultaneously for all timers. To add a timer, first declare it (`timer_[tmr]`) at the top of **ice_timers.F90** (we recommend doing this in both the **mpi/** and **serial/** directories), then add a call to *get_ice_timer* in the subroutine *init_ice_timers*. In the module containing the code to be timed, call `ice_timer_start(timer_[tmr])` at the beginning of the section to be timed, and a similar call to `ice_timer_stop` at the end. Be careful not to have one command outside of a loop and the other command inside. Timers can be run for individual blocks, if desired, by including the block ID in the timer calls.

4.8.2 History fields

To add a variable to be printed in the history output, search for 'example' in **ice_history.F90**:

1. add a frequency flag for the new field
2. add the flag to the namelist (here and also in **ice_in**)
3. add an index number

4. broadcast the flag
5. add a call to `define_hist_field`
6. add a call to `accum_hist_field`

To add an output frequency for an existing variable, see section 4.5.1.

4.8.3 Tracers

For backward compatibility of restart files, each new tracer has its own restart file, generated in its own module, **ice_[tracer].F90**, which also contains as much of the additional tracer code as possible. 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, melt pond volume, aerosols and level ice area and volume (from which ridged ice quantities are derived). Age, level-ice area and melt pond quantities are volume-weighted tracers, while aerosols and level-ice volume are volume-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. **ice_domain_size.F90**: increase `max_ntrcr`
2. **ice_state.F90**: declare `nt_[tracer]`
3. **ice_[tracer].F90**: create initialization, physics, restart routines
4. **ice_fileunits.F90**: add new dump and restart file units
5. **ice_init.F90**:
   - add new module and `tr_[tracer]` to list of used modules and variables
   - add logical namelist variable `tr_[tracer]`
   - initialize namelist variable
   - broadcast 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)
6. **CICE_InitMod.F90**: initialize tracer (includes reading restart file)
7. **CICE_RunMod.F90**, **ice_step_mod.F90**:
   - call routine to write tracer restart file

- call physics routines in **ice_[tracer].F90**

8. **ice_history.F90**: add history variables (Section 4.8.2)
9. **ice_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 3.6.1.

5 Troubleshooting

5.1 Initial setup

The script **comp_ice** is configured so that the files **grid**, **kmt**, **ice_in**, **run_ice**, **iced_gx3_v4.0_kcatbound0** and **ice.restart_file** are NOT overwritten after the first setup. If you wish to make changes to the original files in **input_templates/** rather than those in the run directory, either remove the files from the run directory before executing **comp_ice** or edit the script.

The code may abort during the setup phase for any number of reasons, and often the buffer containing the diagnostic output fails to print before the executable exits. The quickest way to get the diagnostic information is to run the code in an interactive shell with just the command `cice` for serial runs or "`mpirun -np N cice`" for MPI runs, where N is the appropriate number of processors (or a command appropriate for your computer's software).

If the code fails to compile or run, or if the model configuration is changed, try the following:

- create **Macros.***. **Makefile.*** and **run_ice.*** files for your particular platform, if they do not already exist (type 'uname -s' at the prompt and compare the result with the file suffixes; we rename `UNICOS/mp` as `UNICOS` for simplicity).
- modify the `INCLUDE` directory path and other settings for your system in the scripts, **Macros.*** and **Makefile.*** files.
- alter directory paths, file names and the execution command as needed in **run_ice** and **ice_in**.
- ensure that `nprocs` in **ice_in** is equal to `NTASK` in **comp_ice**.
- ensure that the block size `NXBLOCK`, `NYBLOCK` in **comp_ice** is compatible with the `processor_shape` and other domain options in **ice_in**
- if using the rake or space-filling curve algorithms for block distribution (`distribution_type` in **ice_in**) the code will abort if `MXBLCKS` is not large enough. The correct value is provided in the diagnostic output.
- if starting from a restart file, ensure that kcatbound is the same as that used to create the file (`kcatbound` = 0 for the files included in this code distribution). Other configuration parameters, such as `NICECAT`, must also be consistent between runs.
- for stand-alone runs, check that `-Dcoupled` is *not* set in the **Macros.*** file.
- for coupled runs, check that `-Dcoupled` and other coupled-model-specific (eg., CESM, popcice or hadgem) preprocessing options are set in the **Macros.*** file.
- edit the grid size and other parameters in **comp_ice**.
- remove the **compile/** directory completely and recompile.

5.2 Slow execution

On some architectures, underflows (10^{-300} for example) are not flushed to zero automatically. Usually a compiler flag is available to do this, but if not, try uncommenting the block of code at the end of subroutine *stress* in **ice_dyn_evp.F90**. You will take a hit for the extra computations, but it will not be as bad as running with the underflows.

5.3 Debugging hints

Several utilities are available 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_ice **(CICE.F90)** A wrapper for *print_state* that is easily called from numerous points during the timestepping loop (see **CICE.F90_debug**, which can be substituted for **CICE.F90**).

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

`dbug = .true.` **(ice_in)** Print numerous diagnostic quantities.

`print_global` **(ice_in)** If true, compute and print numerous global sums for energy and mass balance analysis. This option can significantly degrade code efficiency.

`print_points` **(ice_in)** If true, print numerous diagnostic quantities for two grid cells, one near the north pole and one in the Weddell Sea. This utility also provides the local grid indices and block and processor numbers (`ip, jp, iblkp, mtask`) for these points, which can be used in conjunction with `check_step`, to call *print_state*. These flags are set in **ice_diagnostics.F90**. This option can be fairly slow, due to gathering data from processors.

global_minval, global_maxval, global_sum **(ice_global_reductions.F90)** Compute and print the minimum and maximum values for an individual real array, or its global sum.

5.4 Known bugs

1. Fluxes sent to the CESM coupler may have incorrect values in grid cells that change from an ice-free state to having ice during the given time step, or vice versa, due to scaling by the ice area. The authors of the CESM flux coupler insist on the area scaling so that the ice and land models are treated consistently in the coupler (but note that the land area does not suddenly become zero in a grid cell, as does the ice area).

2. With the standard CESM radiative scheme (`shortwave` = 'default'), 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.

3. The date-of-onset diagnostic variables, `melt_onset` and `frz_onset`, are not included in the restart file, and therefore may be incorrect for the current year if the run is restarted after Jan 1. Also, these variables were implemented with the Arctic in mind and may be incorrect for the Antarctic.

4. The single-processor *system_clock* time may give erratic results on some architectures.

5. History files that contain time averaged data (`hist_avg = .true.` in **ice_in**) will be incorrect if restarting from midway through an averaging period.

6. In stand-alone runs, restarts from the end of `ycycle` will not be exact.

7. Using the same frequency twice in `histfreq` will have unexpected consequences and causes the code to abort.

5.5 Multi-dimensional output

An issue that has already arisen in the column configuration of the model is the format of the multi-dimensional history variables. Even though each history file includes only a single time slice or average, all netcdf history variables include the time dimension for use with external post-processing software such as NCO. When the time dimension is included for the four-dimensional variables (x, y, z, and categories; strictly speaking, time makes them 5D), the Ferret software package misinterprets the data. In order to look at these variables using Ferret, uncomment the lines indicated by the tag "ferret" in **ice_history.F90** and comment out the lines that they replace.

5.6 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.

Acknowledgments and Copyright

This work has been supported through the Department of Energy Computer Hardware Applied Mathematics and Model Physics (CHAMMP) program, Climate Change Prediction Program (CCPP), and Scientific Discovery through Advanced Computing (SCIDAC) program, with additional support from the T-3 Fluid Dynamics Group at Los Alamos National Laboratory. Special thanks are due to the following people:

- members of the CESM Polar Climate Working Group, including David Bailey, Cecilia Bitz, Bruce Briegleb, Tony Craig, Marika Holland, John Dennis, Julie Schramm, Bonnie Light and Phil Jones,

- Jonathan Gregory of the University of Reading and the U.K. MetOffice for supplying tripole T-fold code and documentation,

- Alison McLaren, Ann Keen and others working with the Hadley Centre GCM for testing non-standard model configurations and providing their code to us,

- the many researchers who tested beta versions of CICE 4.0 and waited patiently for the official release.

Table of namelist options

variable	options/format	description	recommended value
setup_nml		*Time*	
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	
istep0	integer	initial time step number	0
dt	seconds	thermodynamics time step length	3600.
npt	integer	total number of time steps to take	
ndyn_dt	integer	number of dynamics/advection/ridging steps per thermo timestep	1
		Initialization/Restarting	
runtype	initial	start from ice_ic	
	continue	restart using pointer_file	
ice_ic	default	latitude and sst dependent	default
	none	no ice	
	path/file	restart file name	
restart	true/false	initialize using restart file	.true.
restart_dir	path/	path to restart directory	
restart_file	filename prefix	output file for restart dump	'iced'
pointer_file	pointer filename	contains restart filename	
dumpfreq	y	write restart every dumpfreq_n years	y
	m	write restart every dumpfreq_n months	
	d	write restart every dumpfreq_n days	
dumpfreq_n	integer	frequency restart data is written	1
dump_last	true/false	if true, write restart on last time step of simulation	
		Model Output	
diagfreq	integer	frequency of diagnostic output in dt	24
	eg., 10	once every 10 time steps	
diag_type	stdout	write diagnostic output to stdout	
	file	write diagnostic output to file	
diag_file	filename	diagnostic output file (script may reset)	
print_global	true/false	print diagnostic data, global sums	.false.
print_points	true/false	print diagnostic data for two grid points	.false.
latpnt	real	latitude of (2) diagnostic points	
lonpnt	real	longitude of (2) diagnostic points	
dbug	true/false	if true, write extra diagnostics	.false.

Table 7: Namelist options (continued next page).

variable	options/format	description	recommended value
		Model Output, continued	
histfreq	string array	defines output frequencies	
	y	write history every histfreq_n years	
	m	write history every histfreq_n months	
	d	write history every histfreq_n days	
	h	write history every histfreq_n hours	
	1	write history every time step	
	x	unused frequency stream (i.e., do not write)	
histfreq_n	integer array	frequency history output is written	
	0	do not write to history	
hist_avg	true	write time-averaged data	.true.
	false	write snapshots of data	
history_dir	path/	path to history output directory	
history_file	filename prefix	output file for history	'iceh'
history_format	nc	write netCDF history file	nc
	bin	write direct access, binary file	
write_ic	true/false	write initial condition	
incond_dir	path/	path to initial condition directory	
incond_file	filename prefix	output file for initial condition	'iceh'
runid	string	label for run (currently CESM only)	
grid_nml		*Grid*	
grid_format	nc	read netCDF grid and kmt files	bin'
	bin	read direct access, binary file	
grid_type	rectangular	defined in *rectgrid*	displaced_pole
	displaced_pole	read from file in *popgrid*	
	tripole	read from file in *popgrid*	
grid_file	filename	name of grid file to be read	'grid'
kmt_file	filename	name of land mask file to be read	'kmt'
kcatbound	0	original category boundary formula	0
	1	new formula with round numbers	
	2	WMO standard categories	
	-1	one category	
domain_nml		*Domain*	
nprocs	integer	number of processors to use	
processor_shape	slenderX1	1 processor in the y direction (tall, thin)	
	slenderX2	2 processors in the y direction (thin)	
	square-ice	more processors in x than y, ∼square	
	square-pop	more processors in y than x, ∼square	
distribution_type	cartesian	distribute blocks in 2D Cartesian array	
	rake	redistribute blocks among neighbors	
	spacecurve	distribute blocks via space-filling curves	
distribution_weight	block	full block size sets work_per_block	
	latitude	latitude/ocean sets work_per_block	

Table 7: Namelist options (continued).

variable	options/format	description	recommended value
		Domain, continued	
`ew_boundary_type`	`cyclic`	periodic boundary conditions in x-direction	
	`open`	Neumann boundary conditions in x	
	`closed`	Dirichlet boundary conditions in x	
`ns_boundary_type`	`cyclic`	periodic boundary conditions in y-direction	
	`open`	Neumann boundary conditions in y	
	`closed`	Dirichlet boundary conditions in y	
	`tripole`	U-fold tripole boundary conditions in y	
	`tripoleT`	T-fold tripole boundary conditions in y	
tracer_nml		*Tracers*	
`tr_iage`	true/false	ice age	
`restart_age`	true/false	restart tracer values from file	
`tr_FY`	true/false	first-year ice area	
`restart_FY`	true/false	restart tracer values from file	
`tr_lvl`	true/false	level ice area and volume	
`restart_lvl`	true/false	restart tracer values from file	
`tr_pond_cesm`	true/false	CESM melt ponds	
`restart_pond_cesm`	true/false	restart tracer values from file	
`tr_pond_topo`	true/false	topo melt ponds	
`restart_pond_topo`	true/false	restart tracer values from file	
`tr_pond_lvl`	true/false	level-ice melt ponds	
`restart_pond_lvl`	true/false	restart tracer values from file	
`tr_pond_aero`	true/false	aerosols	
`restart_pond_aero`	true/false	restart tracer values from file	
ice_nml		*Physical Parameterizations*	
`kitd`	`0`	delta function ITD approximation	`1`
	`1`	linear remapping ITD approximation	
`kdyn`	`0`	dynamics OFF	`1`
	`1`	EVP dynamics	
`ndte`	integer	number of EVP subcycles	120
`kstrength`	`0`	ice strength formulation [20]	`1`
	`1`	ice strength formulation [44]	
`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	
`advection`	`remap`	linear remapping advection	`remap`
	`upwind`	donor cell advection	
`heat_capacity`	true	salinity-dependent thermodynamics	`.true.`
	false	zero-layer thermodynamic model	
`conduct`	`MU71`	conductivity [39]	
	`bubbly`	conductivity [43]	
`shortwave`	`default`	NCAR CCSM3 distribution method	`default`
	`dEdd`	Delta-Eddington method	

Table 7: Namelist options (continued).

variable	options/format	description	recommended value
albedo_type	default	NCAR CCSM3 albedos	default
	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
		Physical Parameterizations, continued	
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) ...	
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	
snowinfil	true/false	snow infiltration on/off	.true.
rfracmin	$0 \leq r_{min} \leq 1$	minimum melt water added to ponds	0.15
rfracmax	$0 \leq r_{max} \leq 1$	maximum melt water added to ponds	1.0
pndaspect	real	aspect ratio of pond changes (depth:area)	0.8
		Forcing	
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_format	nc	read netCDF atmo forcing files	bin
	bin	read direct access, binary files	
atm_data_type	default	constant values defined in the code	
	ecmwf	ECMWF forcing data	
	ncar	NCAR bulk forcing data	
	LYq	AOMIP/Large-Yeager forcing data	
	monthly	monthly forcing data	
atm_data_dir	path/	path to atmospheric forcing data directory	
calc_strair	true	calculate wind stress and speed	
	false	read wind stress and speed from files	
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)	
Tfrzpt	constant	freezing temperature = -1.8°C	
	linear_S	linear function of salinity profile	
ustar_min	real	minimum value of ocean friction velocity	0.0005 m/s

Table 7: Namelist options (continued).

variable	options/format	description	recommended value
update_ocn_f	true	include frazil water/salt fluxes in ocn fluxes	
	false	do not include (when coupling with POP)	
oceanmixed_ice	true/false	active ocean mixed layer calculation	.true. (if uncoupled)
ocn_data_format	nc	read netCDF ocean forcing files	bin
	bin	read direct access, binary files	
sss_data_type	default	constant values defined in the code	
	clim	climatological data	
	ncar	POP ocean forcing data	
sst_data_type	default	constant values defined in the code	
	clim	climatological data	
	ncar	POP ocean forcing data	
ocn_data_dir	path/	path to oceanic forcing data directory	
oceanmixed_file	filename	data file containing ocean forcing data	
restore_sst	true/false	restore sst to data	
trestore	integer	sst restoring time scale (days)	
restore_ice	true/false	restore ice state along lateral boundaries	
icefields_nml		*History Fields*	
f_$\langle var \rangle$	string	frequency units for writing $\langle var \rangle$ to history	
	y	write history every histfreq_n years	
	m	write history every histfreq_n months	
	d	write history every histfreq_n days	
	h	write history every histfreq_n hours	
	1	write history every time step	
	x	do not write $\langle var \rangle$ to history	
	md	*e.g.,* write both monthly and daily files	

Table 7: Namelist options (continued).

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 (•) with their default values. For other namelist options, see Table 7. All quantities in the code are expressed in MKS units (temperatures may take either Celsius or Kelvin units).

A

a_min	minimum area concentration for computing velocity	0.001
advection	• type of advection algorithm used (‘remap’ or ‘upwind’)	remap
ahmax	• thickness above which ice albedo is constant	0.3 m
aice_extmin	minimum value for ice extent diagnostic	0.15
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’)	
albicei	• near infrared ice albedo for thicker ice	
albicev	• visible ice albedo for thicker ice	
albocn	ocean albedo	0.06
albsnowi	• near infrared, cold snow albedo	
albsnowv	• visible, cold snow albedo	
alpha	floe shape constant for lateral melt	0.66
alv(n)dr(f)	albedo: visible (near IR), direct (diffuse)	
alv(n)dr(f)_gbm	grid-box-mean value of alv(n)dr(f)	
ANGLE	for conversions between the POP grid and latitude-longitude grids	radians
ANGLET	ANGLE converted to T-cells	radians
apondn	area concentration of melt ponds	
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’)	
awtidf	weighting factor for near-ir, diffuse albedo	0.36218
awtidr	weighting factor for near-ir, direct albedo	0.63282
awtvdf	weighting factor for visible, diffuse albedo	0.00182
awtvdr	weighting factor for visible, direct albedo	0.00318
avgsiz	number of cell-averaged fields that can be written to history file	81

B

bignum	a large number	10^{30}
block	data type for blocks	
block_id	global block number	
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	

C

c$\langle n \rangle$	real(n)	
calc_strair	• if true, calculate wind stress	T
calc_Tsfc	• if true, calculate surface temperature	T
Cf	ratio of ridging work to PE change in ridging	17.
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	1.81×10^3 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.

D

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
dardg1dt	rate of fractional area loss by ridging ice	1/s
dardg2dt	rate of fractional area gain by new ridges	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 (see Section 3.5)	1/s
depressT	ratio of freezing temperature to salinity of brine	0.054 deg/ppt
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_info	block distribution information	
distribution_type	• 'cartesian' or 'rake' or 'spacecurve'	
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
dpscale	• time scale for flushing in permeable ice	1×10^{-3}
dragio	drag coefficient for water on ice	0.00536
dragw	drag coefficient for water on ice*ρ_w	kg/m^3
dt	• thermodynamics time step	3600. s
dT_mlt	[see **ice_shortwave.F90**]	1. deg
dte	subcycling time step for EVP dynamics (Δt_e)	s
dtei	1/dte, where dte is the EVP subcycling time step	1/s
dump_file	• output file for restart dump	
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	
dxt	width of T cell (Δx) through the middle	m
dxu	width of U cell (Δx) through the middle	m
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
dvirdgdt	ice volume ridging rate	m/s

E

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
eps11	a small number	10^{-11}
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
evp_damping	• if true, use evp damping procedure [23]	F
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
fcondtop(n)(_f)	conductive heat flux	W/m^2
fcor_blk	Coriolis parameter	1/s
ferrmax	max allowed energy flux error (thermodynamics)	$1.\times10^{-3}$ W/m^2
fhocn	net heat flux to ocean	W/m^2
fhocn_gbm	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
flat	latent heat flux	W/m^2
floediam	effective floe diameter for lateral melt	300. m
flw	incoming longwave radiation	W/m^2
flwout	outgoing longwave radiation	W/m^2
fm	Coriolis parameter * mass in U cell	kg/s
frain	rainfall rate	kg/m^2/s
frazil	frazil ice growth	m
fresh	fresh water flux to ocean	kg/m^2/s
fresh_gbm	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_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_gbm	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	absorbed shortwave radiation	W/m^2
fswfac	scaling factor to adjust ice quantities for updated data	
fswthru	shortwave penetrating to ocean	W/m^2
fswthru_gbm	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' or 'displaced_pole' or 'column'	
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
hicen	ice thickness in category n	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	
history_format	• format of history files	
hm	land/boundary mask, thickness (T-cell)	
hmix	ocean mixed layer depth	20. m
hour	hour of the year	
hp0	pond depth at which transition to bare ice occurs	0.2 m
hpmin	minimum melt pond depth	0.005 m
hpondn	melt pond depth	m
hs_min	minimum thickness for which T_s is computed	$1.\times 10^{-4}$ m
hsmin	minimum snow depth for Delta-Eddington	$1.\times 10^{-4}$ 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
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_ic	• choice of initial conditions (see Table 4)	
ice_ref_salinity	reference salinity for ice-ocean exchanges	4. ppt
icells	number of grid cells with specified property (for vectorization)	
iceruf	ice surface roughness	$5.\times 10^{-4}$ 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	
i(j)hi	last i(j) index of physical domain (local)	
i(j)lo	first i(j) index of physical domain (local)	
ilyr1	index of the top layer in each cat (for eicen)	
ilyrn	index of the bottom layer in each cat (for eicen)	
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

K

kappan	visible extinction coefficient in ice, wavelength>700nm	$17.6\ m^{-1}$
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	2.03 W/m/deg
kimin	minimum conductivity of saline ice	0.10 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
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= Rothrock 1975, 0= Hibler 1979)	1

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	
latpt	• latitude of diagnostic points	degrees N
latt(u)_bounds	latitude of T(U) grid cell corners	degrees N
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.)

lonpt	• 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}
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	
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	
ncat	number of ice categories	5
ndte	• number of subcycles	120
ndyn_dt	• 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
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	
nt_$\langle trcr\rangle$	tracer index	
ntilyr	sum of number of ice layers in all categories	
ntrace	number of fields being transported	
ntrcr	number of tracers transported in remapping	
ntslyr	sum of number of snow layers in all categories	
nu_diag	unit number for diagnostics output file	
nu_dump	unit number for dump file for restarting	
nu_dump_age	unit number for age dump file for restarting	
nu_dump_pond	unit number for melt pond 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_age	unit number for age restart input file	
nu_restart_pond	unit number for melt pond restart input file	
nu_rst_pointer	unit number for pointer to latest restart file	
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

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	
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	
pi	π	
pi2	2π	
pih	$\pi/2$	
pndaspect	• aspect ratio of pond changes (depth:area)	0.8
pointer_file	• input file for restarting	
potT	atmospheric potential temperature	K
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
Pstar	ice strength parameter	2.75×10^4N/m
puny	a small positive number	1×10^{-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	1.16378×10^7kg/m^3
qqqocn	for saturated specific humidity over ocean	6.275724×10^6kg/m^3
Qref	2m atmospheric reference specific humidity	kg/kg

R

R_ice	• parameter for Delta-Eddington ice albedo	
R_pnd	• parameter for Delta-Eddington pond albedo	
R_snw	• parameter for Delta-Eddington snow albedo	
r16_kind	definition of quad precision	selected_real_kind(26)
rad_to_deg	degree-radian conversion	$180/\pi$
radius	earth radius	6.37×10^6 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
restart	• if true, initialize using restart file instead of defaults	T
restart_age	• if true, read age restart file	
restart_dir	• path to restart/dump files	
restart_file	• restart file prefix	
restart_pond_cesm	• if true, read CESM melt pond restart file	
restart_pond_topo	• if true, read topo melt pond restart file	
restart_pond_lvl	• if true, read level-ice melt pond restart file	
restore_ice	• if true, restore ice state along lateral boundaries	

restore_sst	• restore sst to data	
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
rhos	density of snow	330. 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
runtype	• type of initialization used	

S

salin	ice salinity	ppt
saltmax	max salinity, at ice base	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)	
sinw	sine of the turning angle in water	0.
snoice	snow-ice formation	m
snowinfil	• snow infiltration on/off	T
snowpatch	length scale for parameterizing nonuniform snow coverage	0.02 m
spval	special value (single precision)	10^{30}
spval_dbl	special value (double precision)	10^{30}
ss_tltx(y)	sea surface slope 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$\times$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
tday	absolute day number	
Tf	freezing temperature	C
Tffresh	freezing temp of fresh ice	273.15 K
Tfrzpt	• type of freezing temperature ('constant' or 'linear_S')	
time	total elapsed time	s
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
Tmin	minimum allowed internal temperature	-100. C
Tmlt	melting temperature of ice	
Tocnfrz	temperature of constant freezing point parameterization	-1.8 C
tr_FY	• if true, use first-year area tracer	
tr_iage	• if true, use ice age tracer	
tr_pond	• if true, use explicit melt ponds	
trcr	ice tracers	
trcr_depend	tracer dependency on basic state variables	
Tref	2m atmospheric reference temperature	K
trestore	• sst 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_{sfc} error (thermodynamics)	$5.\times 10^{-4}$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 threashold (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	
ustar_min	• minimum friction velocity under ice	
uvel	x-component of ice velocity	m/s
uvm	land/boundary mask, velocity (U-cell)	

V

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
vocn	ocean current in the y-direction	m/s
vonkar	von Karman constant	0.4
vsno(n)	volume per unit area of snow (in category n)	m
vvel	y-component of ice velocity	m/s

W

warmice	value for constant albedo parameterization	0.68
warmsno	value for constant albedo parameterization	0.77
wind	wind speed	m/s
work_g1(23)	allocatable, 2D dbl_kind work array	
work_gi4	allocatable, 2D integer work array	
work_gi8	allocatable, 2D long integer work array	
work_gr	allocatable, 2D real_kind work array	
work_gr3	allocatable, 3D real_kind work array	
work1(2)	(nx_block, ny_block, max_blocks) work array	
worka(bcd)	(nx_block, ny_block) work array	
write_history	if true, write history now	
write_ic	• if true, write initial conditions	
write_restart	if 1, write restart now	

Y

ycycle	• number of years in forcing data cycle	
yday	day of the year	
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}	2. m
zvir	gas constant (water vapor)/gas constant (air) - 1 . . .	0.606

References

[1] T. L. Amundrud, H. Malling, and R. G. Ingram. Geometrical constraints on the evolution of ridged sea ice. *J. Geophys. Res.*, 109, 2004. C06005, doi:10.1029/2003JC002251.

[2] K. C. Armour, L. Thompson C. M. Bitz, and E. C. Hunke. Controls on Arctic sea ice from rst-year and multi-year ice survivability. *J. Climate*, 24:2378–2390, 2011.

[3] C. M. Bitz, M. M. Holland, A. J. Weaver, and M. Eby. Simulating the ice-thickness distribution in a coupled climate model. *J. Geophys. Res.–Oceans*, 106:2441–2463, 2001.

[4] C. M. Bitz and W. H. Lipscomb. An energy-conserving thermodynamic sea ice model for climate study. *J. Geophys. Res.–Oceans*, 104:15669–15677, 1999.

[5] S. Bouillon, T. Fichefet, V. Legat, and G. Madec. The revised elastic-viscous-plastic method. *Ocean Modelling*, page submitted, 2013.

[6] 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 Tech. Note NCAR/TN-472+STR, National Center for Atmospheric Research, 2007.

[7] W. M. Connolley, J. M. Gregory, E. C. Hunke, and A. J. McLaren. On the consistent scaling of terms in the sea ice dynamics equation. *J. Phys. Oceanogr.*, 34:1776–1780, 2004.

[8] J. K. Dukowicz and J. R. Baumgardner. Incremental remapping as a transport/advection algorithm. *J. Comput. Phys.*, 160:318–335, 2000.

[9] J. K. Dukowicz, R. D. Smith, and R. C. Malone. A reformulation and implementation of the Bryan-Cox-Semtner ocean model on the connection machine. *J. Atmos. Oceanic Technol.*, 10:195–208, 1993.

[10] J. K. Dukowicz, R. D. Smith, and R. C. Malone. Implicit free-surface method for the Bryan-Cox-Semtner ocean model. *J. Geophys. Res.–Oceans*, 99:7991–8014, 1994.

[11] 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*, 100:15,965–15,975, 1995.

[12] 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.*, 109:C08007, doi:10.1029/2003JC001989, 2004.

[13] D. L. Feltham, N. Untersteiner, J. S. Wettlaufer, and M. G. Worster. Sea ice is a mushy layer. *Geophys. Res. Lett.*, 33:L14501, doi:10.1029/2006GL026290, 2006.

[14] 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.

[15] D. Flocco and D. L. Feltham. A continuum model of melt pond evolution on Arctic sea ice. *J. Geophys. Res.*, 112:C08016, doi:10.1029/2006JC003836, 2007.

[16] 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.*, 115:C08012, doi:10.1029/2009JC005568, 2010.

[17] 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.*, 2012. submitted.

[18] C. A. Geiger, W. D. Hibler, and S. F. Ackley. Large-scale sea ice drift and deformation: Comparison between models and observations in the western Weddell Sea during 1992. *J. Geophys. Res.–Oceans*, 103:21893–21913, 1998.

[19] K. M. Golden, H. Eicken, A. L. Heaton, J. Miner, D. J. Pringle, and J. Zhu. Thin and thinner: Sea ice mass balance measurements during SHEBA. *Geophys. Res. Lett.*, 34:L16501, doi:10.1029/2007GL030447, 2007.

[20] W. D. Hibler. A dynamic thermodynamic sea ice model. *J. Phys. Oceanogr.*, 9:817–846, 1979.

[21] W. D. Hibler. Modeling a variable thickness sea ice cover. *Mon. Wea. Rev.*, 108:1943–1973, 1980.

[22] 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. Clim.*, 25:14131430, 2012.

[23] E. C. Hunke. Viscous-plastic sea ice dynamics with the EVP model: Linearization issues. *J. Comput. Phys.*, 170:18–38, 2001.

[24] E. C. Hunke and C. M. Bitz. Age characteristics in a multidecadal arctic sea ice simulation. *J. Geophys. Res.*, 114:C08013, doi:10.1029/2008JC005186, 2009.

[25] E. C. Hunke and J. K. Dukowicz. An elastic-viscous-plastic model for sea ice dynamics. *J. Phys. Oceanogr.*, 27:1849–1867, 1997.

[26] E. C. Hunke and J. K. Dukowicz. The Elastic-Viscous-Plastic sea ice dynamics model in general orthogonal curvilinear coordinates on a sphere—Effect of metric terms. *Mon. Wea. Rev.*, 130:1848–1865, 2002.

[27] E. C. Hunke and J. K. Dukowicz. The sea ice momentum equation in the free drift regime. Technical Report LA-UR-03-2219, Los Alamos National Laboratory, 2003.

[28] E. C. Hunke, D. A. Hebert, and O. Lecomte. Level-ice melt ponds in the los alamos sea ice model, cice. *Ocean Modelling*, page in press, 2012.

[29] E. C. Hunke and Y. Zhang. A comparison of sea ice dynamics models at high resolution. *Mon. Wea. Rev.*, 127:396–408, 1999.

[30] 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*, 104:7785–7806, 1999.

[31] B. G. Kauffman and W. G. Large. The CCSM coupler, version 5.0.1. Technical note, National Center for Atmospheric Research, August 2002. http://www.ccsm.ucar.edu/models/.

[32] W. H. Lipscomb. *Modeling the Thickness Distribution of Arctic Sea Ice*. PhD thesis, Dept. of Atmospheric Sciences, Univ. of Washington, Seattle, 1998.

[33] W. H. Lipscomb. Remapping the thickness distribution in sea ice models. *J. Geophys. Res.–Oceans*, 106:13,989–14,000, 2001.

[34] W. H. Lipscomb and E. C. Hunke. Modeling sea ice transport using incremental remapping. *Mon. Wea. Rev.*, 132:1341–1354, 2004.

[35] W. H. Lipscomb, E. C. Hunke, W. Maslowski, and J. Jakacki. Improving ridging schemes for high-resolution sea ice models. *J. Geophys. Res.–Oceans*, 112:C03S91, doi:10.1029/2005JC003355, 2007.

[36] G. A. Maykut. Large-scale heat exchange and ice production in the central Arctic. *J. Geophys. Res.–Oceans*, 87:7971–7984, 1982.

[37] G. A. Maykut and M. G. McPhee. Solar heating of the Arctic mixed layer. *J. Geophys. Res.–Oceans*, 100:24691–24703, 1995.

[38] G. A. Maykut and D. K. Perovich. The role of shortwave radiation in the summer decay of a sea ice cover. *J. Geophys. Res.*, 92(C7):7032–7044, 1987.

[39] G. A. Maykut and N. Untersteiner. Some results from a time dependent thermodynamic model of sea ice. *J. Geophys. Res.*, 76:1550–1575, 1971.

[40] R. J. Murray. Explicit generation of orthogonal grids for ocean models. *J. Comput. Phys.*, 126:251–273, 1996.

[41] Dirk Notz. *Thermodynamic and Fluid-Dynamical Processes in Sea Ice*. PhD thesis, University of Cambridge, UK, 2005.

[42] N. Ono. Specific heat and heat of fusion of sea ice. In H. Oura, editor, *Physics of Snow and Ice*, volume 1, pages 599–610. Institute of Low Temperature Science, Hokkaido, Japan, 1967.

[43] 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.*, 112:C04017, doi:10.1029/2006JC003641, 2007.

[44] D. A. Rothrock. The energetics of the plastic deformation of pack ice by ridging. *J. Geophys. Res.*, 80:4514–4519, 1975.

[45] E. M. Schulson. Brittle failure of ice. *Eng. Fract. Mech.*, 68:1839–1887, 2001.

[46] W. Schwarzacher. Pack ice studies in the Arctic Ocean. *J. Geophys. Res.*, 64:2357–2367, 1959.

[47] A. J. Semtner. A model for the thermodynamic growth of sea ice in numerical investigations of climate. *J. Phys. Oceanogr.*, 6:379–389, 1976.

[48] R. D. Smith, J. K. Dukowicz, and R. C. Malone. Parallel ocean general circulation modeling. *Physica D*, 60:38–61, 1992.

[49] R. D. Smith, S. Kortas, and B. Meltz. Curvilinear coordinates for global ocean models. Technical Report LA-UR-95-1146, Los Alamos National Laboratory, 1995.

[50] M. Steele. Sea ice melting and floe geometry in a simple ice-ocean model. *J. Geophys. Res.*, 97(C11):17729–17738, 1992.

[51] M. Steele, J. Zhang, D. Rothrock, and H. Stern. The force balance of sea ice in a numerical model of the Arctic Ocean. *J. Geophys. Res.–Oceans*, 102:21061–21079, 1997.

[52] A. H. Stroud. *Approximate Calculation of Multiple Integrals*. Prentice-Hall, Englewood Cliffs, New Jersey, 1971. 431 pp.

[53] P. D. Taylor and D. L. Feltham. A model of melt pond evolution on sea ice. *J. Geophys. Res.*, 109:C12007, doi:10.1029/2004JC002361, 2004.

[54] 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.

[55] H. J. Trodahl, S. O. F. Wilkinson, M. J. McGuiness, and T. G. Haskell. Thermal conductivity of sea ice: dependence on temperature and depth. *Geophys. Res. Lett.*, 28:1279–1282, 2001.

[56] M. Tsamados, D. L. Feltham, and A. V. Wilchinsky. Impact of a new anisotropic rheology on simulations of arctic sea ice. *J. Geophys. Res. Oceans*, 118:91–107, 2013.

[57] N. Untersteiner. Calculations of temperature regime and heat budget of sea ice in the Central Arctic. *J. Geophys. Res.*, 69:4755–4766, 1964.

[58] J. Weiss and E. M. Schulson. Coulombic faulting from the grain scale to the geophysical scale: lessons from ice. *J. of Physics D: Applied Physics*, 42:doi 10.1088/0022–3727/42/21/214, 2009.

[59] A. V. Wilchinsky and D. Feltham. Modelling the rheology of sea ice as a collection of diamond-shaped oes. *J. Non-Newtonian Fluid Mech.*, 138:22–32, 2006.

[60] A. V. Wilchinsky and D. L. Feltham. Dependence of sea ice yield-curve shape on ice thickness. *J. Phys. Oceanogr.*, 34(12):2852–2856, 2004.