
CICE-Consortium Documentation

Release 0.0.1

Alice DuVivier

Nov 16, 2017

CONTENTS

1	Table of Contents:	1
1.1	Introduction - CICE5	1
1.1.1	Quick Start guide	1
1.1.1.1	Get the model	1
1.1.1.2	Running the model	1
1.1.1.3	More Details	2
1.1.1.4	Forcing data	3
1.1.1.5	Online resources	3
1.1.2	Major updates	3
1.1.2.1	CICE V5.1	3
1.1.2.2	CICE V5.0	4
1.1.3	Acknowledgements	5
1.1.4	Copyright	5
1.2	Science Guide	5
1.2.1	Coupling with other climate model components	5
1.2.1.1	Atmosphere	7
1.2.1.2	Ocean	9
1.2.1.3	Variable exchange coefficients	10
1.2.2	Model components	11
1.2.2.1	Tracers	13
1.2.2.1.1	Tracers that depend on other tracers (e.g., melt ponds)	14
1.2.2.1.2	Ice age	15
1.2.2.1.3	Sea ice biogeochemistry	15
1.2.2.1.4	Aerosols	18
1.2.2.1.5	Brine height	18
1.2.2.2	Horizontal transport	20
1.2.2.2.1	Reconstructing area and tracer fields	21
1.2.2.2.2	Locating departure triangles	23
1.2.2.2.3	Integrating fields	27
1.2.2.2.4	Updating state variables	28
1.2.2.3	Dynamics	28
1.2.2.3.1	Momentum	29
1.2.2.3.2	Internal stress	30
1.2.2.3.3	Revised approach	33
1.2.2.4	Thickness changes	36
1.2.2.4.1	Transport in thickness space	36
1.2.2.4.2	Mechanical redistribution	39
1.2.2.4.3	Thermodynamics	42
1.3	User Guide	63
1.3.1	Numerical implementation	63

1.3.1.1	Directory structure	63
1.3.1.2	Grid, boundary conditions and masks	67
1.3.1.2.1	Grid domains and blocks	67
1.3.1.2.2	Tripole grids	69
1.3.1.2.3	Bio-grid	70
1.3.1.2.4	Column configuration	70
1.3.1.2.5	Boundary conditions	70
1.3.1.2.6	Masks	71
1.3.1.3	Test configurations	71
1.3.1.4	Initialization and coupling	71
1.3.1.5	Choosing an appropriate time step	72
1.3.1.6	Model output	73
1.3.1.6.1	History files	73
1.3.1.6.2	Diagnostics files	74
1.3.1.6.3	Restart files	75
1.3.2	Execution procedures	76
1.3.2.1	Scripts	78
1.3.2.2	Directories	78
1.3.2.3	Local modifications	78
1.3.2.4	Forcing data	78
1.3.3	Performance	78
1.3.4	Adding things	81
1.3.4.1	Timers	81
1.3.4.2	History fields	81
1.3.4.3	Tracers	82
1.3.5	Troubleshooting	83
1.3.5.1	Initial setup	83
1.3.5.2	Restarts	84
1.3.5.3	Slow execution	84
1.3.5.4	Debugging hints	84
1.3.5.5	Known bugs	85
1.3.5.6	Interpretation of albedos	85
1.3.5.7	Proliferating subprocess parameterizations	86
1.3.6	Testing CICE	86
1.3.6.1	Individual tests and test suites	86
1.3.6.2	Additional testing options	86
1.3.6.3	Test suite format	87
1.3.6.4	Example Tests (Quickstart)	88
1.3.6.4.1	To generate a baseline dataset for a test case	88
1.3.6.4.2	To run a test case and compare to a baseline dataset	88
1.3.6.4.3	To run a test suite to generate baseline data	88
1.3.6.4.4	To run a test suite to compare to baseline data	88
1.3.6.4.5	To compare to another test	89
1.3.6.4.6	Additional Details	89
1.3.6.5	Code compliance test	89
1.3.6.5.1	Method	90
1.3.6.5.2	Practical Testing Procedure	91
1.3.7	Table of namelist options	91
1.4	Index of primary variables and parameters	97
1.4.1	Comprehensive Alphabetical Index	97
1.5	References	112
1.5.0.0.3	References	112
1.6	Useful tools	112

TABLE OF CONTENTS:

1.1 Introduction - CICE5

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–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 [62][13][14]. 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. External routines would 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 5.1, available from <http://oceans11.lanl.gov/trac/CICE/wiki>. It updates CICE5.0, which was released in September 2013. With so many new parameterizations, we must admit that all combinations have not been tested. Also, different parameterizations for various sub-processes (e.g., snow infiltration by melt or sea water) have been introduced as part of the new code options, which need to be unified or pulled out as separate options.

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

1.1.1 Quick Start guide

1.1.1.1 Get the model

Checkout the model from the CICE-Consortium repository,

`github.com/CICE-Consortium`

For more details about how to work in github with CICE, a document can be found [here](#).

1.1.1.2 Running the model

```
> cd consortium
```

```
> ./create.case -c ~/mycase1 -g gx3 -m thunder -s diag1,thread -p 8x1
```

```
> cd ~/mycase1
> ./cice.build
> ./cice.submit/Users/duvivier/Documents/Research/github/CICE-Consortium/CICE/doc/source/all_orig/cice_2_quick_start.rst
```

1.1.1.3 More Details

create.case generates a case, use “create.case -h” for help with the tool. -c is the case name and location (required)

-m is the machine name (required). Currently, there are working ports for NCAR yellowstone and cheyenne, AFRL thunder, NavyDSRC gordon and conrad, and LANL’s wolf machines.

-g is the resolution (default is gx3)

-p is the task x thread/task values (default is 4x1)

-s are comma separated optional env or namelist settings (default is “null”)

-t is the test name and location (cannot be used with -c).

-bd is used to specify the location of the baseline datasets (only used with -t)

-bg is used to specify the cice version name for generating baseline datasets (only used with -t)

-bc is used to specify the cice version name for comparison. I.e., the version name for the baseline dataset (only used with -t)

-testid is used to specify a test ID (used only with -t or -ts)

-ts is used to generate all test cases for a given test suite.

Several files are placed in the case directory

- env.\${machine} defines the environment
- cice.settings defines many variables associated with building and running the model
- makdep.c is a tool that will automatically generate the make dependencies
- Macros.\${machine} defines the Makefile Macros
- Makefile is the makefile used to build the model
- cice.build is a script that build the model
- ice_in is the namelist file
- cice.run is a batch run script
- cice.submit is a simple script that submits the cice.run script

Once the case is created, all scripts and namelist are fully resolved. Users can edit any of the files in the case directory manually to change the model configuration. The file dependency is indicated in the above list. For instance, if any of the files before cice.build in the list are edited, cice.build should be rerun.

The casescripts directory holds scripts used to create the case and can largely be ignored.

In general, when cice.build is executed, the model will build from scratch due to the large dependence on cpps. To change this behavior, edit the env variable ICE_CLEANBUILD in cice.settings.

The cice.submit script just submits the cice.run script. You can use cice.submit or just submit the cice.run script on the command line.

The model will run in the directory defined by the env variable CICE_RUNDIR in cice.settings. Build and run logs will be copied into the case logs directory when complete.

To port, an `env.machine` and `Macros.machine` file have to be added to `scripts/machines` and the `cice.run.setup.csh` file needs to be

- `cd` to `consortium/scripts/machines`
- Copy an existing `env` and `Macros` file to new names for your new machine
- Edit the `env` and `Macros` file
- `cd` to `consortium/scripts`
- Edit the `cice.run.setup.csh` script to add a section for your machine for the batch settings and for the job launch settings
- Download and untar the 1997 dataset to the location defined by `ICE_MACHINE_INPUTDATA` in the `env` file
- Create a file in your home directory called `.cice_proj` and add your preferred account name to the first line.
- You can now create a case and test. If there are problems, you can manually edit the `env`, `Macros`, and `cice.run` files in the case directory until things are working properly. Then you can copy the `env` and `Macros` files back to `consortium/scripts/machines`. You will have to manually modify the `cice.run.setup.csh` script if there are any changes needed there.

1.1.1.4 Forcing data

The code is currently configured to run in standalone mode on a 3 degree grid using atmospheric data from 1997, available as detailed on the [wiki](#). These data files are designed only for testing the code, not for use in production runs or as observational data. Please do not publish results based on these data sets. Module `cicecore/dynamics/cicedynB/ice_forcing.F90` can be modified to change the forcing data.

As currently configured, the model runs on 4 processors. MPI is used for message passing between processors, and OpenMP threading is available. The grid provided here is too small for the code to scale well beyond about 8 processors. A 1 degree grid is provided also, and details about this grid can be found on the [wiki](#).

1.1.1.5 Online resources

DO WE WANT TO KEEP THESE?

primary wiki page:

FAQ:

instructions for code developers:

ongoing or planned development projects:

list of users and publications:

Please send references to your publications using the CICE model to ...

Please report any bugs to Elizabeth Hunke (eclare@lanl.gov)

Good luck!

1.1.2 Major updates

1.1.2.1 CICE V5.1

- include ice velocity in atm-ice coupling updates (e.g. stress) for high-frequency coupling

- allow a variable coefficient for the ice-ocean heat flux
- **several new namelist options improve flexibility, especially for coupled model configurations:**
 - ice-ocean heat flux
 - ‘virtual’ or ‘real’ topo melt pond water
 - ocean freezing temperature
 - high-frequency coupling
 - coupling and computational grids may be different
 - and more
- additional software enhancements improve flexibility and compatibility with CESM, Hadley Centre, and U.S. Navy coupled models
- new diagnostics and updated documentation
- various bug fixes

1.1.2.2 CICE V5.0

- A method for prognosing sea ice salinity, including improved snow-ice formation
- Two new explicit melt pond parameterizations (topo and level-ice)
- Sea ice biogeochemistry
- Elastic-Anisotropic-Plastic rheology
- Improved parameterization of form drag
- The “revised EVP” under-damping approach
- Gracefully handles the case when an internal layer melts completely
- Gregorian calendar with leap years
- Reduced memory and floating-point operations for tracer calculations
- Ice and snow enthalpy defined as tracers
- New history variables for melt ponds, ridging diagnostics, biogeochemistry and more
- Read/write variables on the extended grid, including ghost (halo) cells
- Parallelism option via OpenMP threads
- Improved parallel efficiency through halo masks and new block distributions
- Parallel I/O option via the PIO library
- Restarts in binary or netCDF formats
- CPP options for categories, layers and tracers
- Corrected bugs, particularly for nonstandard configurations.

1.1.3 Acknowledgements

This work has been supported under the Department of Energy’s Climate, Ocean and Sea Ice Modeling project through the Computer Hardware Applied Mathematics and Model Physics (CHAMMP) program, Climate Change Prediction Program (CCPP), Improving the Characterization of Clouds, Aerosols and the Cryosphere in Climate Models (Cloud-Cryo) program and Scientific Discovery through Advanced Computing (SCIDAC) program, with additional support from the T-3 Fluid Dynamics and Solid Mechanics 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, Alice DuVivier, Cecilia Bitz, Bruce Briegleb, Tony Craig, Marika Holland, John Dennis, Julie Schramm, Bonnie Light and Phil Jones.
- Andrew Roberts of the Naval Postgraduate School,
- David Hebert and Olivier Lecomte for their melt pond work,
- 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,
- Daniel Feltham and his research group for several new parameterizations and documentation,
- Sylvain Bouillon for the revised EVP approach,
- the many researchers who tested beta versions of CICE 5 and waited patiently for the official release.

1.1.4 Copyright

© Copyright 2013, LANS LLC. All rights reserved. Unless otherwise indicated, this information has been authored by an employee or employees of the Los Alamos National Security, LLC (LANS), operator of the Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396 with the U.S. Department of Energy. The U.S. Government has rights to use, reproduce, and distribute this information. The public may copy and use this information without charge, provided that this Notice and any statement of authorship are reproduced on all copies. Neither the Government nor LANS makes any warranty, express or implied, or assumes any liability or responsibility for the use of this information. Beginning with version 4.0, the CICE code carries Los Alamos Software Release number LA-CC-06-012.

1.2 Science Guide

1.2.1 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 [39] 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 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. In CESM, the sea ice model may exchange coupling fluxes using a different grid than the computational grid. This functionality is activated using the namelist variable `gridcpl_file`. Another namelist variable `highfreq`, allows the high-frequency coupling procedure implemented in the Regional Arctic System Model (RASIM). In particular, the relative atmosphere-ice velocity ($\vec{U}_a - \vec{u}$) is used instead of the full atmospheric velocity for computing turbulent fluxes in the atmospheric boundary layer.

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

Table 1.1: Table 1

Variable	Description	Interaction with flux coupler
z_o	Atmosphere level height	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
\vec{U}_a	Wind velocity	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
Q_a	Specific humidity	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
ρ_a	Air density	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
Θ_a	Air potential temperature	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
T_a	Air temperature	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
$F_{sw\downarrow}$	Incoming shortwave radiation (4 bands)	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
$F_{L\downarrow}$	Incoming longwave radiation	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
F_{rain}	Rainfall rate	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
F_{snow}	Snowfall rate	From <i>atmosphere model</i> via flux coupler to <i>sea ice model</i>
F_{frzmlt}	Freezing/melting potential	From <i>ocean model</i> via flux coupler to <i>sea ice model</i>
T_w	Sea surface temperature	From <i>ocean model</i> via flux coupler to <i>sea ice model</i>
S	Sea surface salinity	From <i>ocean model</i> via flux coupler to <i>sea ice model</i>
∇H_o	Sea surface slope	From <i>ocean model</i> via flux coupler to <i>sea ice model</i>
\vec{U}_w	Surface ocean currents	From <i>ocean model</i> via flux coupler to <i>sea ice model</i>
$\vec{\tau}_a$	Wind stress	From <i>sea ice model</i> via flux coupler to <i>atmosphere model</i>
F_s	Sensible heat flux	From <i>sea ice model</i> via flux coupler to <i>atmosphere model</i>
F_l	Latent heat flux	From <i>sea ice model</i> via flux coupler to <i>atmosphere model</i>
$F_{L\uparrow}$	Outgoing longwave radiation	From <i>sea ice model</i> via flux coupler to <i>atmosphere model</i>
F_{evap}	Evaporated water	From <i>sea ice model</i> via flux coupler to <i>atmosphere model</i>
α	Surface albedo (4 bands)	From <i>sea ice model</i> via flux coupler to <i>atmosphere model</i>
T_{sfc}	Surface temperature	From <i>sea ice model</i> via flux coupler to <i>atmosphere model</i>
$F_{sw\downarrow}$	Penetrating shortwave radiation	From <i>sea ice model</i> via flux coupler to <i>ocean model</i>
F_{water}	Fresh water flux	From <i>sea ice model</i> via flux coupler to <i>ocean model</i>
F_{hocn}	Net heat flux to ocean	From <i>sea ice model</i> via flux coupler to <i>ocean model</i>
F_{salt}	Salt flux	From <i>sea ice model</i> via flux coupler to <i>ocean model</i>
$\vec{\tau}_w$	Ice-ocean stress	From <i>sea ice model</i> via flux coupler to <i>ocean model</i>
F_{bio}	Biogeochemical fluxes	From <i>sea ice model</i> via flux coupler to <i>ocean model</i>
a_i	Ice fraction	From <i>sea ice model</i> via flux coupler to both <i>ocean and atmosphere models</i>
T_a^{ref}	2m reference temperature (diagnostic)	From <i>sea ice model</i> via flux coupler to both <i>ocean and atmosphere models</i>
Q_a^{ref}	2m reference humidity (diagnostic)	From <i>sea ice model</i> via flux coupler to both <i>ocean and atmosphere models</i>
F_{swabs}	Absorbed shortwave (diagnostic)	From <i>sea ice model</i> via flux coupler to both <i>ocean and atmosphere models</i>

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

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

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.

1.2.1.1 Atmosphere

The wind velocity, specific humidity, air density and potential temperature at the given level height z_o 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 [65]. 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 *Thermodynamics*, these fluxes depend nonlinearly on the ice surface temperature T_{sfc} . The balance equation is iterated until convergence, and the resulting fluxes and T_{sfc} are then passed to the flux coupler.

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

Wind stress and transfer coefficients for the turbulent heat fluxes are computed in subroutine `atmo_boundary_layer` following [39]. 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”

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

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

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

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

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

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

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

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

In a departure from the parameterization used in [39], we use profiles for the stable case following [38],

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

The coefficients are then updated as

$$\begin{aligned}c'_u &= \frac{c_u}{1 + c_u(\lambda - \psi_m)/\kappa} \\ c'_\theta &= \frac{c_\theta}{1 + c_\theta(\lambda - \psi_s)/\kappa} \\ c'_q &= c'_\theta\end{aligned}\tag{1.6}$$

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

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

where L_{vap} and L_{ice} are latent heats of vaporization and fusion, ρ_a is the density of air and c_p is its specific heat. Again following [38], 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 [Thermodynamic surface forcing balance](#).

For CICE run in stand-alone mode (i.e., uncoupled), the AOMIP shortwave and longwave radiation formulas are available in **ice_forcing.F90**. In function *longwave_rosati_miyakoda*, downwelling longwave is computed as

$$F_{lw\downarrow} = \epsilon \sigma T_s^4 - \epsilon \sigma T_a^4 (0.39 - 0.05 e_a^{1/2}) (1 - 0.8 f_{cld}) - 4 \epsilon \sigma T_a^3 (T_s - T_a)\tag{1.8}$$

where the atmospheric vapor pressure (mb) is $e_a = 1000 Q_a / (0.622 + 0.378 Q_a)$, $\epsilon = 0.97$ is the ocean emissivity, σ is the Stephan-Boltzman constant, f_{cld} is the cloud cover fraction, and T_a is the surface air temperature (K). The first term on the right is upwelling longwave due to the mean (merged) ice and ocean surface temperature, T_s (K), and the other terms on the right represent the net longwave radiation patterned after [56]. The downwelling longwave formula of [54] is also available in function *longwave_parkinson_washington*:

$$F_{lw\downarrow} = \epsilon \sigma T_a^4 (1 - 0.261 \exp(-7.77 \times 10^{-4} T_a^2)) (1 + 0.275 f_{cld})\tag{1.9}$$

The value of $F_{lw\uparrow}$ is different for each ice thickness category, while $F_{lw\downarrow}$ depends on the mean value of surface temperature averaged over all of the thickness categories and open water.

The AOMIP shortwave forcing formula (in subroutine *compute_shortwave*) incorporates the cloud fraction and humidity through the atmospheric vapor pressure:

$$F_{sw\downarrow} = \frac{1353 \cos^2 Z}{10^{-3}(\cos Z + 2.7)e_a + 1.085 \cos Z + 0.1} (1 - 0.6f_{cl}^3) > 0 \quad (1.10)$$

where $\cos Z$ is the cosine of the solar zenith angle.

1.2.1.2 Ocean

New sea ice forms when the ocean temperature drops below its freezing temperature. In the Bitz and Lipscomb thermodynamics, [7] $T_f = -\mu S$, where S is the seawater salinity and $\mu = 0.054$ °/ppt is the ratio of the freezing temperature of brine to its salinity (linear liquidus approximation). For the mushy thermodynamics, T_f is given by a piecewise linear liquidus relation. The ocean model calculates the new ice formation; if the freezing/melting potential F_{frzmlt} is positive, its value represents a certain amount of frazil ice that has formed in one or more layers of the ocean and floated to the surface. (The ocean model assumes that the amount of new ice implied by the freezing potential actually forms.)

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

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

There is a flux of salt into the ocean under melting conditions, and a (negative) flux when sea water is freezing. However, melting sea ice ultimately freshens the top ocean layer, since the ocean is much more saline than the ice. The ice model passes the net flux of salt F_{salt} to the flux coupler, based on the net change in salt for ice in all categories. In the present configuration, `ice_ref_salinity` is used for computing the salt flux, although the ice salinity used in the thermodynamic calculation has differing values in the ice layers.

A fraction of the incoming shortwave $F_{sw\downarrow}$ penetrates the snow and ice layers and passes into the ocean, as described in Section [Thermodynamic surface forcing balance](#).

Many ice models compute the sea surface slope ∇H_o from geostrophic ocean currents provided by an ocean model or other data source. In our case, the sea surface height H_o 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,

$$\vec{\tau}_w = c_w \rho_w |\vec{U}_w - \vec{u}| \left[(\vec{U}_w - \vec{u}) \cos \theta + \hat{k} \times (\vec{U}_w - \vec{u}) \sin \theta \right] \quad (1.11)$$

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, 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.

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

1.2.1.3 Variable exchange coefficients

In the default CICE setup, atmospheric and oceanic neutral drag coefficients (c_u and c_w) are assumed constant in time and space. These constants are chosen to reflect friction associated with an effective sea ice surface roughness at the ice–atmosphere and ice–ocean interfaces. Sea ice (in both Arctic and Antarctic) contains pressure ridges as well as floe and melt pond edges that act as discrete obstructions to the flow of air or water past the ice, and are a source of form drag. Following [71] and based on recent theoretical developments [46][45], the neutral drag coefficients can now be estimated from properties of the ice cover such as ice concentration, vertical extent and area of the ridges, freeboard and floe draft, and size of floes and melt ponds. The new parameterization allows the drag coefficients to be coupled to the sea ice state and therefore to evolve spatially and temporally. This parameterization is contained in the subroutine *neutral_drag_coeffs* and is accessed by setting *formdrag* = true in the namelist.

Following [71], consider the general case of fluid flow obstructed by N randomly oriented obstacles of height H and transverse length L_y , distributed on a domain surface area S_T . Under the assumption of a logarithmic fluid velocity profile, the general formulation of the form drag coefficient can be expressed as

$$C_d = \frac{NcS_c^2\gamma L_y H}{2S_T} \left[\frac{\ln(H/z_0)}{\ln(z_{ref}/z_0)} \right]^2, \quad (1.12)$$

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

$$S_c = (1 - \exp(-s_l D/H))^{1/2}, \quad (1.13)$$

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

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

$$\text{Cdn_atm} = \text{Cdn_atm_rdg} + \text{Cdn_atm_floe} + \text{Cdn_atm_skin} + \text{Cdn_atm_pond}, \quad (1.14)$$

$$\text{Cdn_ocn} = \text{Cdn_ocn_rdg} + \text{Cdn_ocn_floe} + \text{Cdn_ocn_skin}. \quad (1.15)$$

The contributions to form drag from ridges (and keels underneath the ice), floe edges and melt pond edges can be expressed using the general formulation of equation (1.12) (see [71] for details). Individual terms in equation (1.15) are fully described in [71]. Following [4] the skin drag coefficient is parametrized as

$$\text{Cdn_atm/ocn_skin} = a_i \left(1 - m_{(s/k)} \frac{H_{(s/k)}}{D_{(s/k)}} \right) c_{s(s/k)}, \text{ if } \frac{H_{(s/k)}}{D_{(s/k)}} \geq \frac{1}{m_{(s/k)}}, \quad (1.16)$$

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

Calculation of equations (1.12) – (1.16) requires that small-scale geometrical properties of the ice cover be related to average grid cell quantities already computed in the sea ice model. These intermediate quantities are briefly presented here and described in more detail in [71]. The sail height is given by

$$H_s = 2 \frac{v_{rdg}}{a_{rdg}} \left(\frac{\alpha \tan \alpha_k R_d + \beta \tan \alpha_s R_h}{\phi_r \tan \alpha_k R_d + \phi_k \tan \alpha_s R_h^2} \right), \quad (1.17)$$

and the distance between sails

$$D_s = 2H_s \frac{a_i}{a_{rdg}} \left(\frac{\alpha}{\tan \alpha_s} + \frac{\beta}{\tan \alpha_k} \frac{R_h}{R_d} \right), \quad (1.18)$$

where $0 < \alpha < 1$ and $0 < \beta < 1$ are weight functions, α_s and α_k are the sail and keel slope, ϕ_s and ϕ_k are constant porosities for the sails and keels, and we assume constant ratios for the average keel depth and sail height ($H_k/H_s = R_h$) and for the average distances between keels and between sails ($D_k/D_s = R_d$). With the assumption of hydrostatic equilibrium, the effective ice plus snow freeboard is $H_f = \bar{h}_i(1 - \rho_i/\rho_w) + \bar{h}_s(1 - \rho_s/\rho_w)$, where ρ_i , ρ_w and ρ_s are respectively the densities of sea ice, water and snow, \bar{h}_i is the mean ice thickness and \bar{h}_s is the mean snow thickness (means taken over the ice covered regions). For the melt pond edge elevation we assume that the melt pond surface is at the same level as the ocean surface surrounding the floes [20][21][22] and use the simplification $H_p = H_f$. Finally to estimate the typical floe size L_A , distance between floes, D_F , and melt pond size, L_P we use the parameterizations of [46] to relate these quantities to the ice and pond concentrations. All of these intermediate quantities are available as history output, along with *Cdn_atm*, *Cdn_ocn* and the ratio *Cdn_atm_ratio_n* between the total atmospheric drag and the atmospheric neutral drag coefficient.

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

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

1.2.2 Model components

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

The fundamental equation solved by CICE is [69]:

$$\frac{\partial g}{\partial t} = -\nabla \cdot (g\mathbf{u}) - \frac{\partial}{\partial h}(fg) + \psi, \quad (1.19)$$

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 (1.19) 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 [6][42].) 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 [47]. The continuous function $g(h)$ is replaced by the discrete variable a_{in} , defined as the fractional area covered by ice in the open water by a_{i0} , giving $\sum_{n=0}^{N_C} a_{in} = 1$ by definition.

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

*Table 2 : Data exchanged between the CESM flux coupler and the sea ice model Lower boundary values for thickness categories, in meters, for the three distribution options (*kcatbound*) and linear remapping (*kitd* = 1). In the WMO*

case, the distribution used depends on the number of categories used.

Table 1.2: Table 2

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

In addition to the fractional ice area, a_{in} , we define the following state variables for each category n . In a change from previous CICE versions, we no longer carry snow and ice energy as separate variables; instead they and sea ice salinity are carried as tracers on snow and ice volume.

- 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; it is discussed in Section [Thermodynamics](#). (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.) CICE allows multiple snow layers, but the default value is $N_s = 1$.
- S_i , the bulk sea ice salt content in layer k , equal to the product of the ice layer volume and the sea ice salinity tracer.
- 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 Equation (1.19) 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 [12] as adapted for sea ice by [43]; this scheme is discussed in Section [Horizontal transport](#). Ice is transported in thickness space using the remapping scheme of [42], as described in Section [Transport in thickness space](#). The mechanical redistribution scheme, based on [69], [57], [27], [19], and [44] is outlined in Section [Mechanical redistribution](#). 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 ice growth rate f in each thickness category. We use the elastic-viscous-plastic (EVP) ice dynamics scheme of [32], as modified by [10], [30], [33] and [34], or a new elastic-anisotropic-plastic model [78][76][72] to find the velocity, as described in Section [Dynamics](#). Finally, we use a thermodynamic model to compute f (Section [Thermodynamics](#)). 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.

1.2.2.1 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

$$\frac{\partial}{\partial t}(a_{in}) + \nabla \cdot (a_{in} \mathbf{u}) = 0, \quad (1.20)$$

$$\frac{\partial v_{in}}{\partial t} + \nabla \cdot (v_{in} \mathbf{u}) = 0, \quad (1.21)$$

$$\frac{\partial v_{sn}}{\partial t} + \nabla \cdot (v_{sn} \mathbf{u}) = 0. \quad (1.22)$$

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

$$\frac{\partial h_{in} a_{in}}{\partial t} + \nabla \cdot (h_{in} a_{in} \mathbf{u}) = 0, \quad (1.23)$$

$$\frac{\partial h_{sn} a_{in}}{\partial t} + \nabla \cdot (h_{sn} a_{in} \mathbf{u}) = 0. \quad (1.24)$$

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 [Horizontal transport](#)). 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

$$\frac{\partial (a_{in} T_n)}{\partial t} + \nabla \cdot (a_{in} T_n \mathbf{u}) = 0, \quad (1.25)$$

$$\frac{\partial (v_{in} T_n)}{\partial t} + \nabla \cdot (v_{in} T_n \mathbf{u}) = 0, \quad (1.26)$$

$$\frac{\partial (v_{sn} T_n)}{\partial t} + \nabla \cdot (v_{sn} T_n \mathbf{u}) = 0. \quad (1.27)$$

Equation (1.25) describes the transport of surface temperature, whereas Equation (1.26) and Equation (1.27) describe the transport of ice and snow enthalpy, salt, and 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 Equation (1.25), Equation (1.26), or Equation (1.27), 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 (Sections [Tracers that depend on other tracers \(e.g., melt ponds\)](#) and [Melt ponds](#)), level ice area and level ice volume (used to diagnose ridged ice area and volume, Section [Mechanical redistribution](#)) are also available. Users may add any number of additional tracers that are transported conservatively provided that *trcr_depend* is defined appropriately. See Section [Tracers](#) for guidance on adding tracers.

1.2.2.1.1 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 Equations (1.116)–(1.119). 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

$$\frac{\partial}{\partial t}(a_{pnd}a_i) + \nabla \cdot (a_{pnd}a_i\mathbf{u}) = 0, \quad (1.28)$$

$$\frac{\partial}{\partial t}(h_{pnd}a_{pnd}a_i) + \nabla \cdot (h_{pnd}a_{pnd}a_i\mathbf{u}) = 0. \quad (1.29)$$

(These equations represent quantities within one thickness category; all melt pond calculations are performed for each category, separately.) Equation (1.29) expresses conservation of melt pond volume, but in this form highlights that the quantity tracked in the code is the pond depth tracer h_{pnd} , which depends on the pond area tracer a_{pnd} . Likewise, a_{pnd} is a tracer on ice area (Equation (1.28)), 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](#).

Mean thickness over			grid cell	
grid cell	sea ice	pond	0	1
			open water	ice area fraction
$h_i a_i$	h_i		$1 - a_i$	a_i
			CESM and topo ponds	
			unponded ice ponded ice fraction	
$h_{pnd} a_{pnd} a_i$	$h_{pnd} a_{pnd}$	h_{pnd}	$(1 - a_{pnd}) a_i$	$a_{pnd} a_i$
			Level-ice ponds	
			deformed ice level ice fraction	
			$(1 - a_{lvl}) a_i$	$a_{lvl} a_i$
			unponded ice ponded ice	
$h_{pnd} a_{pnd} a_{lvl} a_i$	$h_{pnd} a_{pnd} a_{lvl}$	h_{pnd}	$(1 - a_{pnd}) a_{lvl} a_i$	$a_{pnd} a_{lvl} a_i$

Fig. 1.1: Figure 1

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

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

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

$$a_{pnd}^{t+\Delta t} = \frac{a_{pnd}^t a_i^t}{a_i^{t+\Delta t}}. \quad (1.30)$$

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

$$h_{pnd}^{t+\Delta t} = \frac{h_{pnd}^t a_{pnd}^t a_i^t}{a_{pnd}^{t+\Delta t} a_i^{t+\Delta t}}. \quad (1.31)$$

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 ,

$$\sum_n a_{pnd}^{t+\Delta t}(n) a_i^{t+\Delta t}(n) = \sum_n a_{pnd}^t(n) a_i^t(n). \quad (1.32)$$

Thus,

$$\begin{aligned} a_{pnd}^{t+\Delta t}(m) &= \frac{\sum_n a_{pnd}^t(n) a_i^t(n) - \sum_{n \neq m} a_{pnd}^{t+\Delta t}(n) a_i^{t+\Delta t}(n)}{a_i^{t+\Delta t}(m)} \\ &= \frac{a_{pnd}^t(m) a_i^t(m) + \sum_{n \neq m} \Delta (a_{pnd} a_i)^{t+\Delta t}}{a_i^{t+\Delta t}(m)} \end{aligned} \quad (1.33)$$

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 [Melt ponds](#).

1.2.2.1.2 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 [\[31\]](#).

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 [\[2\]](#).

1.2.2.1.3 Sea ice biogeochemistry

Ice algal photosynthesis leads to carbon fixation and pigment buildup throughout much of the pack ice in springtime, including warm layers in contact with seawater and the central ice matrix. Turbulence moves seed organisms and trace elements upward across the ocean interface from the mixed layer [\[3\]](#), and convection allows them to penetrate deep into the brine channel network. Gravity drainage is strongly coupled to carbonate and alkalinity relationships; bio-active gases including molecular oxygen, dimethyl sulfide (DMS) and methane exhibit dynamic cycling within and around the ice matrix. All may ultimately be transferred across the upper interface, into snow cover and toward the atmosphere. Chlorophyll is often concentrated in thin layers, capable of absorbing downwelling radiation and redistributing the implied heat locally.

Although biogeochemical activity occurs throughout the sea ice column, in the present release we restrict ourselves to simulations of activity taking place in the lowest few vertical centimeters. A thin intermediate zone located in this vicinity can be considered distinct because it is relatively warm, extremely porous and necessarily in close contact with the seawater nutrient supply. The ecosystems supported are therefore especially intense, so that they constitute a logical starting point for code development. The resulting band of biological activity will be referred to most often in what follows as the 'bottom zone.' It constitutes the dominant habitat for ice algae across most of the noncoastal Arctic and among land fast ice systems of the Southern Hemisphere. In the literature and some portions of the code, however,

the alternate term ‘skeletal layer’ is used. For many purposes, the two are interchangeable. But in fact, the latter is a reference to dendritic structures more typically observable in the wintertime; ‘bottom zone’ is the more general term.

Ecological succession in the bottom zone interacts intimately with chemistry of the upper ocean–atmosphere system. Photosynthesis is constrained initially by light limitation but soon becomes a matter of resource availability [3][11]. Radiation intensity dictates the timing of bloom inception. The direct inflow of salt laden water can then be filtered immediately for the dissolved nutrient forms, which are converted locally into biomass. In association with the carbon storage, considerable DMS and molecular oxygen are generated [17]. Melt and flush periods cause bottom ice organisms to be rejected from the matrix mechanically. The dominant primary producers among them are pennate diatoms, a class subject to rapid sinking [40]. Hence the amount of bottom layer biological activity dictates high latitude nutrient and trace gas levels in the early spring, then carbon drawdown moving into the period of breakup.

Light transmission passes into the bottom zone through overlying layers of snow and ice. Attenuation by the ice algae is integrated and averaged over the bottom three centimeters in which they reside. Our current CICE release treats inflow of the primary Arctic nutrients nitrate and silicate, along with the major pathway for nitrogen recycling via ammonia. Both fertilizers and byproducts are permitted to exchange back out of the porous bottom volume on a time scale of order hours [3][40]. Biomass multiplies exponentially as light and nutrient restrictions are lifted, following the rising polar sun. Uptake accelerates until there is a transition to flux limitation [17]. At variable times and locations, a kinetic balance is often reached in which dissolved nitrogen is sequestered faster than it can leave the porous volume. Computational terms involved in the ecodynamic simulation resemble Michaelis-Menten enzyme kinetics, and have been adapted here from a series of pioneering ice biogeochemistry models [3][37][40]. CICE biogeochemical applications include a series of Pan-Arctic simulations, conceptually extending from primary production and carbon cycling to the release of trace gases into the ice domain [11][17].

The bottom ice biogeochemistry module described here is designed for ready attachment of extra-nutrient cycling, including byproduct and detrital processing. Our own mechanism for the generation of DMS is included in the release as an example. For full details please see descriptions and schematics in [17]. Generally speaking, the primary nutrient flow as nitrogen simultaneously supports formation of silicate frustules by the ice diatoms, along with carbon biomass production and the implied chlorophyll buildup. Synthesis of organosulfur metabolites is handled simultaneously and in direct proportion. DMS spins off of this subsystem at much higher levels than those familiar from open water studies, since the ice algae are especially abundant and suffer strong cryological, osmotic and oxidant stress in an extreme environmental regime [66]. The sulfur transformations are governed by a typical pattern of routing kinetics within and below the ice [17], which reduces the net yield of volatile gas but still permits considerable buildup and release during the spring thaw.

Biogeochemical processing is modelled as a single layer of reactive tracers attached to the sea ice bottom. Optional settings are available via the `zbgc_nml` namelist in **ice_in**. The prefix `tr_bgc` signifies a biogeochemical tracer, while the postscript `_sk` indicates that a particular quantity is restricted to the band of bottom or skeletal ice material. History fields are controlled in the `icefields_bgc_nml` namelist. As with other CICE history fields, the suffix `_ai` indicates that the field is multiplied by ice area and is therefore a grid-cell-mean quantity.

Setting the flag `skl_bgc` to true turns on a reduced ‘NP’ (Nitrogen-Plankton) biogeochemistry consisting of nitrate and one algal tracer. Ammonium (`tr_bgc_Am_sk`), silicate (`tr_bgc_Sil_sk`), dimethyl sulfoniopropionate (DMSP) in particulate form (`tr_bgc_DMSPp_sk`), DMSP in dissolved form (`tr_bgc_DMSPd_sk`), and volatile dimethyl sulfide (`tr_bgc_DMS_sk`) may also be included by setting the respective flags to true in **ice_in**.

All biogeochemical tracers (T_b) are brine concentrations multiplied by the skeletal layer thickness (h_{sk}). Bulk tracer concentrations are written to diagnostic files and are found by the expression $T_b \phi_{sk} / h_{sk}$ where ϕ_{sk} is the skeletal layer porosity. Both h_{sk} and ϕ_{sk} are defined in **ice_zbgc_shared.F90**.

Tracers T_b are area conserved and follow the horizontal transport equation ([eq:transport_aT]). The initial concentrations of tracers are specified in subroutine **init_bgc** in **ice_zbgc.F90**. Silicate and nitrate may be initialized and forced with climatology. Parameters `sil_data_type` and `nit_data_type` are set in **ice_in** and take the values ‘clim’ (climatology) and ‘default’ (constant). `nit_data_type` may also take the value ‘sss’ which sets nitrate to the ocean salinity value. For climatology, the data directory `bgc_data_dir` must also be specified. If `restore_bgc` is true, then values are linearly restored to climatology according to the restoring timescale `trestore`.

For each horizontal grid point, local biogeochemical tracer equations are solved in **ice_algae.F90**. There are two types

of ice–ocean tracer flux formulations: ‘Jin2006’ modelled after the growth rate dependent piston velocity of [37], and ‘constant’ modelled after the constant piston velocity of [17]. The formulation is specified in **ice_in** by setting *bgc_flux_type* equal to ‘Jin2006’ or ‘constant’.

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

$$\frac{dT_b}{dt} = w_b \frac{\Delta T_b}{\Delta z} + R_b(T_j : j = 1, \dots, N_b) \quad (1.34)$$

where R_b represents the nonlinear biochemical reaction terms detailed in [17] and Δz is a length scale representing the molecular sublayer of the ice–ocean interface. Its value is absorbed in the piston velocity parameters. The piston velocity w_b depends on the particular tracer and the flux formulation.

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

$$w_b = -p_g \left| m_1 + m_2 \frac{dh}{dt} - m_3 \left(\frac{dh}{dt} \right)^2 \right| \quad (1.35)$$

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

$$w_b = p_m \left| m_2 \frac{dh}{dt} - m_3 \left(\frac{dh}{dt} \right)^2 \right| \quad (1.36)$$

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

In ‘Jin2006’, the algal tracer (N_a) responds to ice melt in the same manner as the other tracers from Equation (1.36). However, this is not the case for ice growth. Unlike dissolved nutrients, algae are able to cling to the ice matrix and resist expulsion during desalination. For this reason, algal tracers do not flux between ice and ocean during ice growth unless the ice algal brine concentration is less than the ocean algal concentration (N_o). Then the ocean seeds the sea ice concentration according to

$$w_b \frac{\Delta N_a}{\Delta z} = \frac{N_o h_{sk} / \phi_{sk} - N_a}{\Delta t} \quad (1.37)$$

The ‘constant’ formulation uses a fixed piston velocity (PVC) for positive ice growth rates for all tracers except N_a . As in ‘Jin2006’, congelation ice growth seeds the sea ice algal population according to Equation (1.37) when $N_a < N_o h_{sk} / \phi_{sk}$. For bottom ice melt, all tracers follow the prescription

$$w_b \frac{\Delta T_b}{\Delta z} = \begin{cases} T_b |dh_i/dt| / h_{sk} & \text{if } |dh_i/dt| \Delta t / h_{sk} < 1 \\ T_b / \Delta t & \text{otherwise.} \end{cases} \quad (1.38)$$

All of the necessary fluxes required for coupling to an ocean model component are contained in a single array, *flux_bio*.

This bottom-specific biogeochemistry module necessarily simplifies the full sea ice ecosystem. For the moment, zooplankton comprise an imaginary tracer of indefinite mass that make their presence felt mainly through the recycling of ammonia [17][37]. Consumer organisms are essentially siphoned off of the standard primary production pathway as a fixed fraction of overall growth [3]. Transfer velocities relating the upper water column to ice fluids and solutes are now parameterized as a function of off-line congelation rates, based on laboratory brine expulsion data [3][17][37]. Thus far we have tested removal from the active bottom zone as a single adjustable time constant, but results have not been formalized or adequately evaluated against data. Internally consistent connections with gravity drainage will be implemented in the vertical biogeochemistry module being developed for future release in CICE.

1.2.2.1.4 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 (Equations (1.26) and (1.27)), 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*, *melth*, *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 [29]. 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.

1.2.2.1.5 Brine height

The brine height, h_b , is the distance from the ice–ocean interface to the brine surface. When *tr_brine* is set true in **ice_in** and *TRBRI* is set equal to 1 in **comp_ice**, the brine surface can move relative to the ice surface. Physically, this occurs when the ice is permeable and there is a nonzero pressure head: the difference between the brine height and the equilibrium sea surface. Brine height motion is computed in **ice_brine.F90** from thermodynamic variables (*melts*, *melth*, *melth*, etc) and the ice microstructural state deduced from internal bulk salinities and temperature. In the current release, this tracer is for diagnostic purposes only; it is driven by the prognostic salinity parameterization but is not used for computing salinity. In future releases it will be used for transporting biogeochemical tracers vertically through the ice.

Brine height is transported horizontally as the fraction $f_{bri} = h_b/h_i$, a volume conserved tracer (Equation (1.26)). Note that unlike the sea ice porosity, brine height fraction may be greater than 1 when $h_b > h_i$.

Vertical transport processes are, generally, a result of the brine motion. Therefore the vertical transport equations for biogeochemical tracers will be defined only where brine is present. This region, from the ice–ocean interface to the brine height, defines the domain of the vertical bio-grid, whose resolution is independent of the sea ice vertical grid and is set at compile time (see Section *Bio-grid*). The ice microstructural state, determined in **ice_brine.F90**, is computed from sea ice salinities and temperatures linearly interpolated to the bio-grid. When $h_b > h_i$, the upper surface brine is assumed to have the same microstructural properties as the ice surface.

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

$$h'_b = h_b(t) + \Delta h_b|_{h_i, h_s}. \quad (1.39)$$

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

$$w = -\frac{\Pi^* \bar{\rho} g}{\mu} \frac{h_p}{h_i}. \quad (1.40)$$

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

$$\Pi^* = \left(\frac{1}{h} \sum_{i=1}^N \frac{\Delta z_i}{\Pi_i} \right)^{-1} \quad (1.41)$$

where the sea ice is composed of N vertical layers with i th layer thickness Δz_i and permeability Π_i ([eq:topo_permea]). The average sea ice density is $\bar{\rho}$ specified in **ice_zbgc_shared.F90**. The hydraulic head is $h_p = h_b - h_{sl}$ where h_{sl} is the sea level given by

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

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

$$\frac{\partial h_b}{\partial t} = -w h_p \quad (1.42)$$

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

Since h_{sl} remains relatively unchanged during Darcy flow, Equation (1.42) has the approximate solution

$$h_b(t + \Delta t) \approx h_{sl}(t + \Delta t) + [h'_b - h_{sl}(t + \Delta t)] \exp \{-w \Delta t\}. \quad (1.43)$$

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

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

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

$$\Delta h_b|_{h_i, h_s} = (\Delta h_b)_{top} - (\Delta h_i)_{bot} - \frac{\rho_s}{\rho_o} \Delta h_s. \quad (1.44)$$

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

$$h_b(t + \Delta t) = h_b(t) + \Delta h_i \quad (1.45)$$

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

Both the volume fraction f_{bri} and the area-weighted brine height h_b can be written to the history file. The history variable $fbri$ contains the volume-averaged brine fraction tracer

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

while $hbri$ is comparable to h_i (h_i)

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

where the sums are taken over thickness categories.

1.2.2.2 Horizontal transport

We wish to solve the continuity or transport equation (Equation (1.20)) for the fractional ice area in each thickness category n . Equation (1.20) describes the conservation of ice area under horizontal transport. It is obtained from Equation (1.19) by discretizing g and neglecting the second and third terms on the right-hand side, which are treated separately (Sections *Transport in thickness space* and *Mechanical redistribution*).

There are similar conservation equations for ice volume (Equation (1.21)), snow volume (Equation (1.22)), ice energy and snow energy:

$$\frac{\partial e_{ink}}{\partial t} + \nabla \cdot (e_{ink} \mathbf{u}) = 0, \quad (1.46)$$

$$\frac{\partial e_{snk}}{\partial t} + \nabla \cdot (e_{snk} \mathbf{u}) = 0. \quad (1.47)$$

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 [12] as modified for sea ice by [43]. 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.

1.2.2.2.1 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, \bar{h} for thickness, and \bar{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 [12]. 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

$$\int_A a dA = \bar{a} A, \quad (1.48)$$

where $A = \int_A dA$ is the grid cell area. Equation (1.48) is satisfied if $a(\mathbf{r})$ has the form

$$a(\mathbf{r}) = \bar{a} + \alpha_a \langle \nabla a \rangle \cdot (\mathbf{r} - \bar{\mathbf{r}}), \quad (1.49)$$

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 Equation (1.49) that the ice area at the cell center ($\mathbf{r} = 0$) is

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

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 $\bar{\mathbf{r}}$, $\bar{x} = \int_A x dA / A$ and $\bar{y} = \int_A y dA / A$, are evaluated using the triangle integration formulas described in Section *Integrating fields*. 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 [12], but there is no analog in [12] to the enthalpy. The reconstructed ice or snow thickness $h(\mathbf{r})$ and enthalpy $q(\mathbf{r})$ must satisfy

$$\int_A a h dA = \bar{a} \tilde{h} A, \quad (1.50)$$

$$\int_A a h q dA = \bar{a} \tilde{h} \hat{q} A, \quad (1.51)$$

where $\tilde{h} = h(\bar{\mathbf{r}})$ 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 ([eq:mean_thickness]) and ([eq:mean_enthalpy]) are satisfied when $h(\mathbf{r})$ and $q(\mathbf{r})$ are given by

$$h(\mathbf{r}) = \tilde{h} + \alpha_h \langle \nabla h \rangle \cdot (\mathbf{r} - \bar{\mathbf{r}}), \quad (1.52)$$

$$q(\mathbf{r}) = \hat{q} + \alpha_q \langle \nabla q \rangle \cdot (\mathbf{r} - \hat{\mathbf{r}}), \quad (1.53)$$

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

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

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

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

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

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

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

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

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

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 Equation (1.52) and Equation (1.53), 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 Equation (1.26) and Equation (1.27) 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 ϕ_{\max} and ϕ_{\min} be the maximum and minimum values of the reconstructed ϕ within the cell. Since the reconstruction is linear, ϕ_{\max} and ϕ_{\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. Starting 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.

1.2.2.2.2 Locating departure triangles

The method for locating departure triangles is discussed in detail by [12]. 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 [Integrating fields](#)) in the coordinate system where they have been reconstructed (Section [Reconstructing area and tracer fields](#)). 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 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](#), reproduced from [12], 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.

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

[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 $\tilde{y}_2 = y_b$.

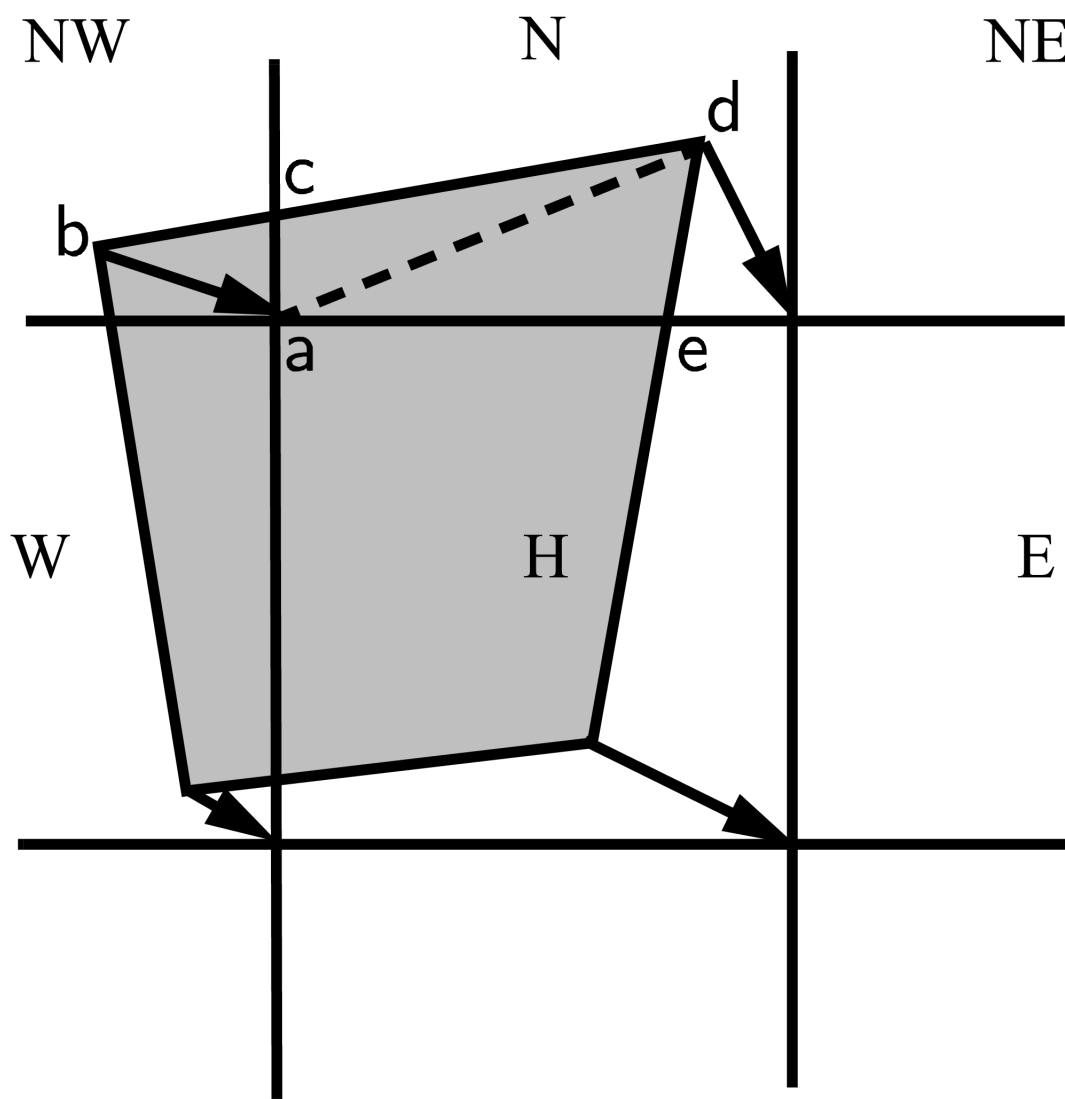


Fig. 1.2: Figure 2

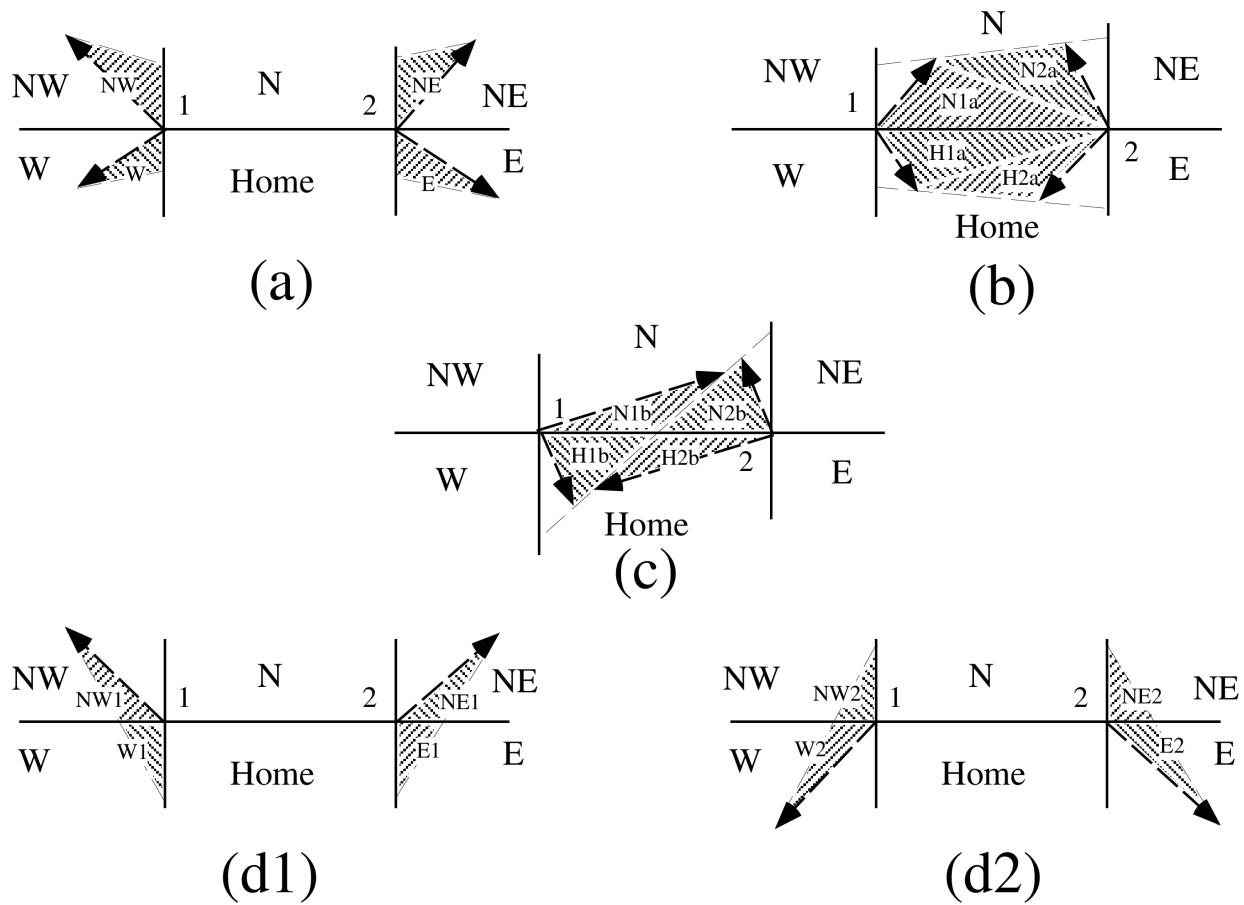


Fig. 1.3: Figure 3

Table 1.3: Table 3

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 $\tilde{y}_2 > 0$

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 maps each grid cell to a 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:

$$\nabla \cdot \mathbf{u} = \frac{1}{A} \left[\left(\frac{u_{NE} + u_{SE}}{2} \right) L_E + \left(\frac{u_{NW} + u_{SW}}{2} \right) L_W + \left(\frac{u_{NE} + u_{NW}}{2} \right) L_N + \left(\frac{u_{SE} + u_{SW}}{2} \right) L_S \right], \quad (1.54)$$

where L is an edge length and the indices N, S, E, W denote compass directions. Equation (1.54) is equivalent to the divergence computed in the EVP dynamics (Section [Dynamics](#)). 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 [\[12\]](#) for locating triangles. In their paper, departure points are defined by projecting cell corner velocities directly backward. That is,

$$\mathbf{x}_D = -\mathbf{u} \Delta t, \quad (1.55)$$

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.

1.2.2.2.3 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

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

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

$$F_a = A_T f(\mathbf{x}_0), \quad (1.56)$$

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

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

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

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

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

Equation (1.56) 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,

$$F_h = \frac{A_T}{3} \sum_{i=1}^3 f(\mathbf{x}'_i), \quad (1.57)$$

where $\mathbf{x}'_i = (\mathbf{x}_0 + \mathbf{x}_i)/2$ are points lying halfway between the midpoint and the three vertices. [\[12\]](#) use this formula to compute transports of the product ρT , which is analogous to ice volume. Equation (1.57) 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 [67]:

$$F_q = A_T \left[-\frac{9}{16} f(\mathbf{x}_0) + \frac{25}{48} \sum_{i=1}^3 f(\mathbf{x}_i'') \right] \quad (1.58)$$

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.

1.2.2.2.4 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

$$a'_{in}(i, j) = a_{in}(i, j) + \frac{F_{aE}(i-1, j) - F_{aE}(i, j) + F_{aN}(i, j-1) - F_{aN}(i, j)}{A(i, j)} \quad (1.59)$$

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 Equation (1.59) 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.

1.2.2.3 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 [26]. The elastic-anisotropic-plastic (EAP) model, on the other hand, explicitly accounts for the observed sub-continuum anisotropy of the sea ice cover [78][76]. 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 [78][72]. 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 [32], [30], [33] and [34] and the EAP dynamics in [72]. 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 [36] and [72]. 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 [33] and [34], with revisions to the numerical solver as in [8]. The implementation of the EAP sea ice dynamics into CICE is described in detail in [72].

1.2.2.3.1 Momentum

The force balance per unit area in the ice pack is given by a two-dimensional momentum equation [26], obtained by integrating the 3D equation through the thickness of the ice in the vertical direction:

$$m \frac{\partial \mathbf{u}}{\partial t} = \nabla \cdot \boldsymbol{\sigma} + \vec{\tau}_a + \vec{\tau}_w - \hat{k} \times m f \mathbf{u} - m g \nabla H_o, \quad (1.60)$$

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 [34] and :cite:CGHM04‘.

The momentum equation is discretized in time as follows, for the classic EVP approach. First, for clarity, the two components of Equation (1.60) are

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

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

$$\underbrace{\left(\frac{m}{\Delta t_e} + \text{vrel} \cos \theta \right)}_{cca} u^{k+1} - \underbrace{(m f + \text{vrel} \sin \theta)}_{ccb} v^{k+1} = \underbrace{\frac{\partial \sigma_{1j}^{k+1}}{\partial x_j}}_{\text{strintx}} + \underbrace{\tau_{ax} - m g \frac{\partial H_o}{\partial x}}_{\text{forcex}} + \underbrace{\text{vrel} (U_w \cos \theta - V_w \sin \theta)}_{\text{waterx}} + \frac{m}{\Delta t_e} u^k, \quad (1.61)$$

$$\underbrace{(m f + \text{vrel} \sin \theta)}_{ccb} u^{k+1} + \underbrace{\left(\frac{m}{\Delta t_e} + \text{vrel} \cos \theta \right)}_{cca} v^{k+1} = \underbrace{\frac{\partial \sigma_{2j}^{k+1}}{\partial x_j}}_{\text{strinty}} + \underbrace{\tau_{ay} - m g \frac{\partial H_o}{\partial y}}_{\text{forcey}} + \underbrace{\text{vrel} (U_w \sin \theta + V_w \cos \theta)}_{\text{watery}} + \frac{m}{\Delta t_e} v^k, \quad (1.62)$$

and $\text{vrel} \cdot \text{waterx}(y) = \text{taux}(y)$.

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

$$\hat{u} = F_u + \tau_{ax} - m g \frac{\partial H_o}{\partial x} + \text{vrel} (U_w \cos \theta - V_w \sin \theta) + \frac{m}{\Delta t_e} u^k \quad (1.63)$$

$$\hat{v} = F_v + \tau_{ay} - m g \frac{\partial H_o}{\partial y} + \text{vrel} (U_w \sin \theta + V_w \cos \theta) + \frac{m}{\Delta t_e} v^k, \quad (1.64)$$

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

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

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

$$a = \frac{m}{\Delta t_e} + \text{vrel} \cos \theta \quad (1.65)$$

$$b = mf + \text{vrel} \sin \theta. \quad (1.66)$$

When the subcycling is finished for each (thermodynamic) time step, the ice–ocean stress must be constructed from $\text{taux}(y)$ and the terms containing vrel on the left hand side of the equations.

The Hibler-Bryan form for the ice-ocean stress [28] is included in **ice_dyn_shared.F90** but is currently commented out, pending further testing.

1.2.2.3.2 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,

$$\frac{1}{E} \frac{\partial \sigma_1}{\partial t} + \frac{\sigma_1}{2\zeta} + \frac{P}{2\zeta} = D_D, \quad (1.67)$$

$$\frac{1}{E} \frac{\partial \sigma_2}{\partial t} + \frac{\sigma_2}{2\eta} = D_T, \quad (1.68)$$

$$\frac{1}{E} \frac{\partial \sigma_{12}}{\partial t} + \frac{\sigma_{12}}{2\eta} = \frac{1}{2} D_S, \quad (1.69)$$

where

$$D_D = \dot{\epsilon}_{11} + \dot{\epsilon}_{22},$$

$$D_T = \dot{\epsilon}_{11} - \dot{\epsilon}_{22},$$

$$D_S = 2\dot{\epsilon}_{12},$$

$$\dot{\epsilon}_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right),$$

$$\zeta = \frac{P}{2\Delta},$$

$$\eta = \frac{P}{2\Delta e^2},$$

$$\Delta = \left[D_D^2 + \frac{1}{e^2} (D_T^2 + D_S^2) \right]^{1/2},$$

and P is a function of the ice thickness and concentration, described in Section [Mechanical redistribution](#). The dynamics component employs a “replacement pressure” (see [\[24\]](#), 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 $T = E_o \Delta t$ and E_o (eyc) is a tunable parameter less than one. The stress equations (1.67)–(1.69) 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 Sections [Revised approach](#) and [Choosing an appropriate time step](#).

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 [\[33\]](#).

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 [Figure 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 [Figure 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

$$\mathbf{A} = \int_{\mathbb{S}} \vartheta(\mathbf{r}) \mathbf{r} \mathbf{r} d\mathbf{r},$$

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 [Figure 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} ,

$$\frac{D\mathbf{A}}{Dt} = \mathbf{F}_{iso}(\mathbf{A}) + \mathbf{F}_{frac}(\mathbf{A}, \boldsymbol{\sigma}), \quad (1.70)$$

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 [\[78\]](#), based on laboratory experiments by [\[58\]](#) 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.

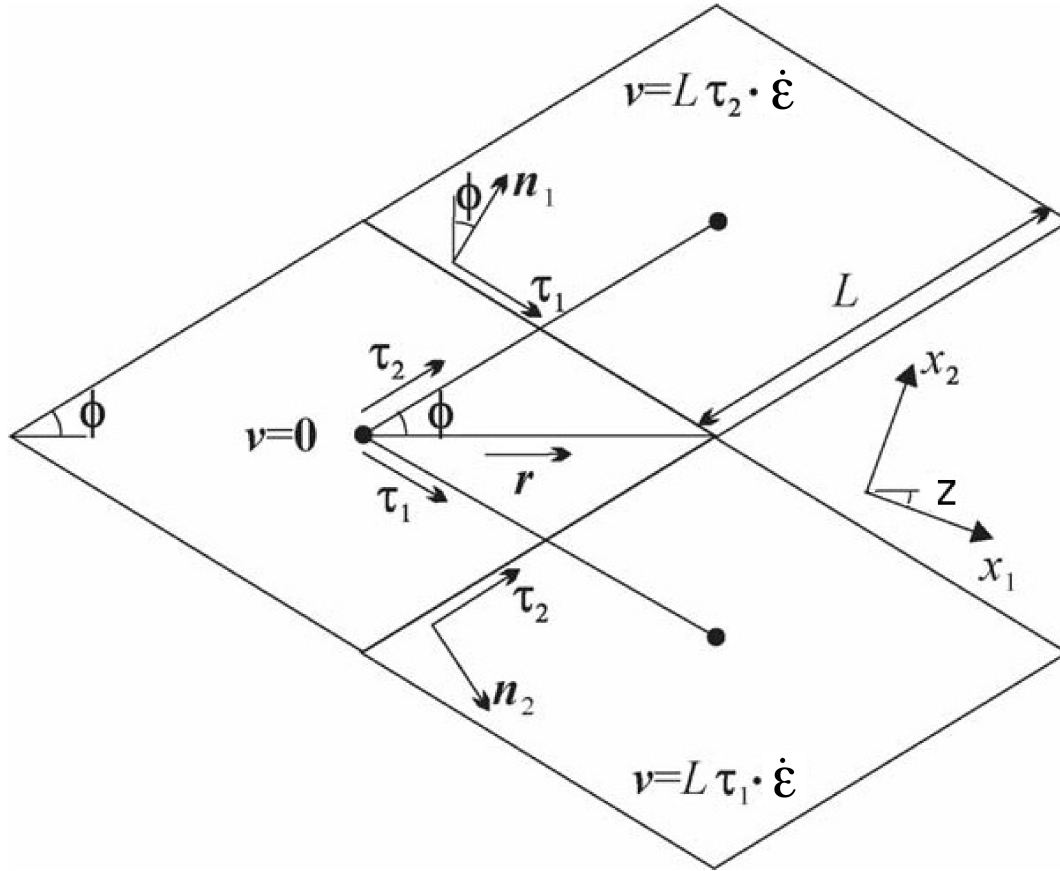


Fig. 1.4: Figure 5

Figure 5 : Geometry of interlocking diamond-shaped floes (taken from [78]). ϕ is half of the acute angle of the diamonds. L is the edge length. \mathbf{n}_1 , \mathbf{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 .

The new anisotropic rheology requires solving the evolution Equation (1.70) 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. Equation (1.70) then reduces to the system of two equations:

$$\begin{aligned} \frac{\partial A_{11}}{\partial t} &= -k_t \left(A_{11} - \frac{1}{2} \right) + M_{11}, \\ \frac{\partial A_{12}}{\partial t} &= -k_t A_{12} + M_{12}, \end{aligned}$$

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 Equation (1.70) that are given in [78] and [72]. 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 [78], 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

$$\boldsymbol{\sigma}^b(\mathbf{r}, h) = P_r(h)\boldsymbol{\sigma}_r^b(\mathbf{r}) + P_s(h)\boldsymbol{\sigma}_s^b(\mathbf{r}), \quad (1.71)$$

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 Figure 5, and the angle z defined in Figure 5. In the stress expressions above the underlying floes are assumed parallel, but in a continuum-scale sea ice region the floes can possess different orientations in different places and we take the mean sea ice stress over a collection of floes to be given by the average

$$\boldsymbol{\sigma}^{EAP}(h) = P_r(h) \int_{\mathcal{S}} \vartheta(\mathbf{r}) [\boldsymbol{\sigma}_r^b(\mathbf{r}) + k\boldsymbol{\sigma}_s^b(\mathbf{r})] d\mathbf{r} \quad (1.72)$$

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,

$$\frac{\partial \sigma_1}{\partial t} + \frac{\sigma_1}{2T} = \frac{\sigma_1^{EAP}}{2T}, \quad (1.73)$$

$$\frac{\partial \sigma_2}{\partial t} + \frac{\sigma_2}{2T} = \frac{\sigma_2^{EAP}}{2T}, \quad (1.74)$$

$$\frac{\partial \sigma_{12}}{\partial t} + \frac{\sigma_{12}}{2T} = \frac{\sigma_{12}^{EAP}}{2T}, \quad (1.75)$$

where the anisotropic stress $\boldsymbol{\sigma}^{EAP}$ is defined in a look-up table 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 Equations (1.73)–(1.75) 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 [72]. 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 $|\dot{\epsilon}|\alpha_r(\theta)$, with $\alpha_r(\theta)$ and $\alpha_s(\theta)$ given by [77],

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

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

1.2.2.3.3 Revised approach

A modification of the standard elastic-viscous-plastic (EVP) approach for sea ice dynamics has been proposed by [8], 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

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

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

Momentum

The momentum equations become

$$\begin{aligned}\beta \frac{m}{\Delta t} (u^{k+1} - u^k) &= \bar{u} + \mathbf{vrel} (-u^{k+1} \cos \theta + v^{k+1} \sin \theta) + mf v^{k+1} - \frac{m}{\Delta t} u^{k+1} \\ \beta \frac{m}{\Delta t} (v^{k+1} - v^k) &= \bar{v} - \mathbf{vrel} (u^{k+1} \sin \theta + v^{k+1} \cos \theta) - mf u^{k+1} - \frac{m}{\Delta t} v^{k+1}\end{aligned}$$

where

$$\bar{u} = F_u + \tau_{ax} - mg \frac{\partial H_o}{\partial x} + \mathbf{vrel} (U_w \cos \theta - V_w \sin \theta) + \frac{m}{\Delta t} u^\circ \quad (1.76)$$

$$\bar{v} = F_v + \tau_{ay} - mg \frac{\partial H_o}{\partial y} + \mathbf{vrel} (U_w \sin \theta + V_w \cos \theta) + \frac{m}{\Delta t} v^\circ, \quad (1.77)$$

\mathbf{u}° 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. Equations (1.76) and (1.77) differ from Equations (1.63) and (1.64) 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

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

$$\tilde{\mathbf{u}} = \bar{\mathbf{u}} + \beta \frac{m}{\Delta t} \mathbf{u}^k, \quad (1.78)$$

and b is the same as in Equation (1.66).

Stress

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

$$\sigma_1^{k+1} = \left(\sigma_1^k + \frac{P}{\Delta} \frac{\Delta t_e}{2T} (\dot{\epsilon} - \Delta) \right) * \left(1 + \frac{\Delta t_e}{2T} \right) \quad (1.79)$$

If we set

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

then Equation (1.79) becomes

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

This is equivalent to Eq. (23) in [8], 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. (23) in [8] 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, [8] 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} :

$$\begin{aligned}\sigma_1^{k+1} (1 + \alpha_1) &= \sigma_1^k + \alpha_1 \frac{P}{\Delta} D_D, \\ \sigma_2^{k+1} (1 + \alpha_1) &= \sigma_2^k + \frac{\alpha_1}{e^2} \frac{P}{\Delta} D_T, \\ \sigma_{12}^{k+1} (1 + \alpha_1) &= \sigma_{12}^k + \frac{\alpha_1}{2e^2} \frac{P}{\Delta} D_S.\end{aligned}$$

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

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

Then

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

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

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

For classic EVP,

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

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

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

\tilde{u} must also be defined for revised EVP as in Equation (1.78). The extra terms in \tilde{a} and \tilde{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. A final difference is that the revised approach initializes the stresses to 0 at the beginning of each time step, while the classic EVP approach uses the previous time step value.

1.2.2.4 Thickness changes

1.2.2.4.1 Transport in thickness space

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

$$\frac{\partial g}{\partial t} + \frac{\partial}{\partial h}(fg) = 0, \quad (1.80)$$

which is obtained from Equation (1.19) by neglecting the first and third terms on the right-hand side. We use the remapping method of [42], 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 *Thermodynamics*) to compute the new mean thicknesses h_n^{m+1} at time $m + 1$. The time step must be small enough that trajectories do not cross; i.e., $h_n^{m+1} < h_{n+1}^{m+1}$ for each pair of adjacent categories. The growth rate at $h = h_n$ is given by $f_n = (h_n^{m+1} - h_n^m)/\Delta t$. By linear interpolation we estimate the growth rate F_n at the upper category boundary H_n :

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

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

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

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:

$$\int_{H_L}^{H_R} g \, dh = a_n, \quad (1.81)$$

$$\int_{H_L}^{H_R} h g \, dh = v_n. \quad (1.82)$$

The simplest polynomial that can satisfy both equations is a line. It is convenient to change coordinates, writing $g(\eta) = g_1 \eta + g_0$, where $\eta = h - H_L$ and the coefficients g_0 and g_1 are to be determined. Then Equations (1.81) and (1.82) can be written as

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

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

$$g_0 = \frac{6a_n}{\eta_R^2} \left(\frac{2\eta_R}{3} - \eta_n \right), \quad (1.83)$$

$$g_1 = \frac{12a_n}{\eta_R^3} \left(\eta_n - \frac{\eta_R}{2} \right). \quad (1.84)$$

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

$$g(0) = \frac{6a_n}{\eta_R^2} \left(\frac{2\eta_R}{3} - \eta_n \right) = g_0, \quad (1.85)$$

$$g(\eta_R) = \frac{6a_n}{\eta_R^2} \left(\eta_n - \frac{\eta_R}{3} \right). \quad (1.86)$$

Equation (1.85) implies that $g(0) < 0$ when $\eta_n > 2\eta_R/3$, i.e., when h_n lies in the right third of the thickness range (H_L, H_R) . Similarly, Equation (1.86) implies that $g(\eta_R) < 0$ when $\eta_n < \eta_R/3$, i.e., when h_n is in the left third of the range. Since negative values of g are unphysical, a different solution is needed when h_n lies outside the central third of the thickness range. If h_n is in the left third of the range, we define a cutoff thickness, $H_C = 3h_n - 2H_L$, and set $g = 0$ between H_C and H_R . Equations (1.83) and (1.84) are then valid with η_R redefined as $H_C - H_L$. And if h_n is in the right third of the range, we define $H_C = 3h_n - 2H_R$ and set $g = 0$ between H_L and H_C . In this case, (1.83) and (1.84) apply with $\eta_R = H_R - H_C$ and $\eta_n = h_n - H_C$.

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

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

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

$$\Delta a_n = \int_{H_n}^{H_n^*} g \, dh, \quad (1.87)$$

$$\Delta v_n = \int_{H_n}^{H_n^*} h \, g \, dh. \quad (1.88)$$

If $H_n^* < H_n$, ice area and volume are transferred from category $n + 1$ to n using Equations (1.87) and (1.88) with the limits of integration reversed. To evaluate the integrals we change coordinates from h to $\eta = h - H_L$, where H_L is the left limit of the range over which $g > 0$, and write $g(\eta)$ using Equations (1.83) and (1.84). In this way we obtain the new areas a_n and volumes v_n between the original boundaries H_{n-1} and H_n in each category. The new thicknesses, $h_n = v_n/a_n$, are guaranteed to lie in the range (H_{n-1}, H_n) . If $g = 0$ in the part of a category that is remapped to a neighboring category, no ice is transferred.

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

The left and right boundaries of the domain require special treatment. If ice is growing in open water at a rate F_0 , the left boundary H_0 is shifted to the right by $F_0 \Delta t$ before g is constructed in category 1, then reset to zero after the remapping is complete. New ice is then added to the grid cell, conserving area, volume, and energy. If ice cannot grow in open water (because the ocean is too warm or the net surface energy flux is downward), H_0 is fixed at zero, and the

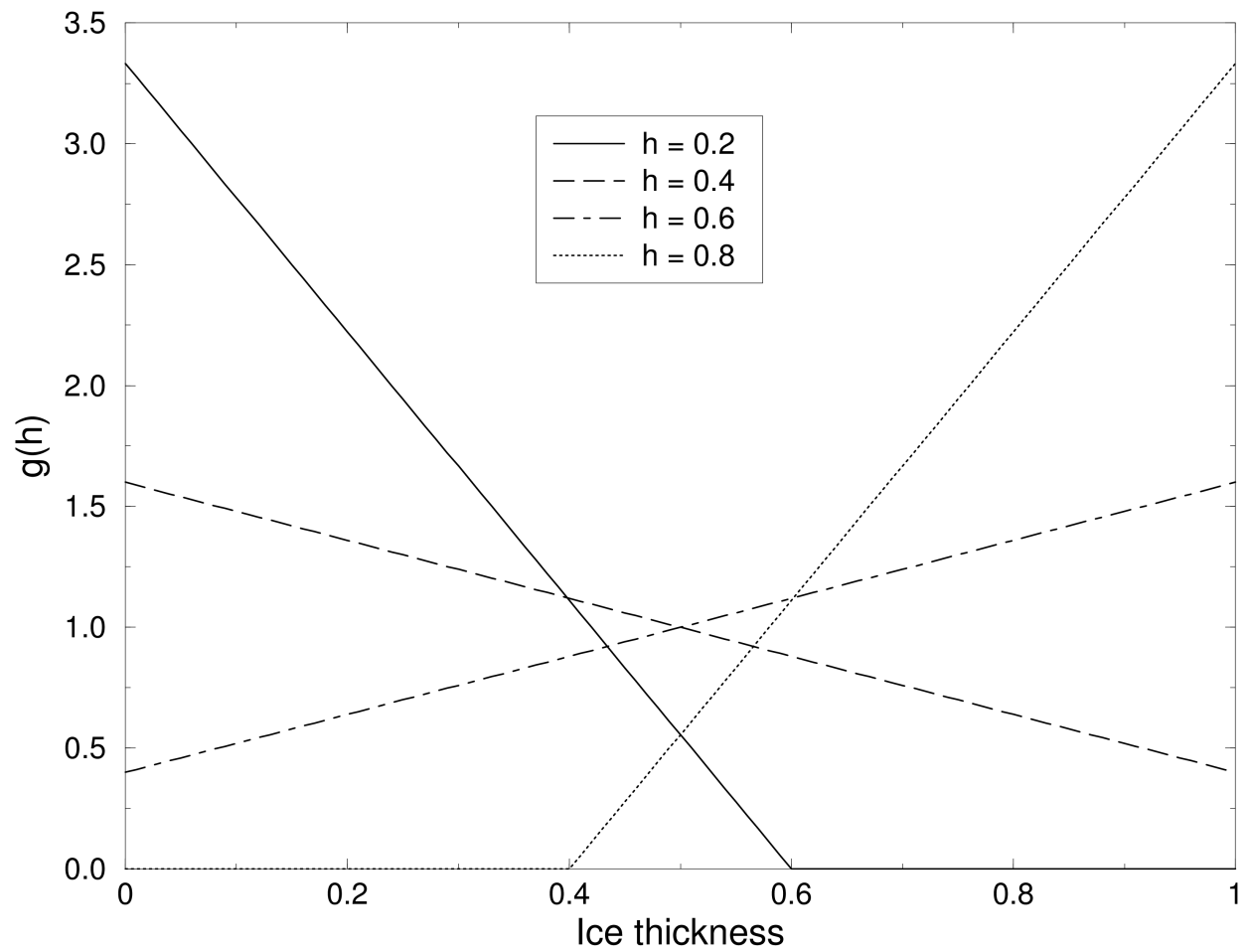


Fig. 1.5: Figure 4

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 [6], and categories outside their prescribed boundaries are merged with neighboring categories as needed. For time steps of less than a day and category thickness ranges of 10 cm or more, this simplification is needed rarely, if ever.

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

1.2.2.4.2 Mechanical redistribution

The last term on the right-hand side of Equation (1.19) 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 [69], [57], [27], [19], and [44]. 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 [69] and set

$$b(h) = \begin{cases} \frac{2}{G^*} \left(1 - \frac{G(h)}{G^*}\right) & \text{if } G(h) < G^* \\ 0 & \text{otherwise} \end{cases} \quad (1.89)$$

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 :

$$a_{Pn} = \frac{2}{G^*} (G_n - G_{n-1}) \left(1 - \frac{G_{n-1} + G_n}{2G^*}\right), \quad (1.90)$$

where a_{Pn} is the ratio of the ice area ridging (or open water area closing) in category n to the total area ridging and closing, and G_n is the total fractional ice area in categories 0 to n . Equation (1.90) applies to categories with $G_n < G^*$. If $G_{n-1} < G^* < G_n$, then Equation (1.90) 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 [69] we set $G^* = 0.15$.

If the spatial resolution is too fine for a given time step Δt , the weighting function Equation (1.89) 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 [44]:

$$b(h) = \frac{\exp[-G(h)/a^*]}{a^*[1 - \exp(-1/a^*)]} \quad (1.91)$$

When integrated between category boundaries, Equation (1.91) implies

$$a_{Pn} = \frac{\exp(-G_{n-1}/a^*) - \exp(-G_n/a^*)}{1 - \exp(-1/a^*)} \quad (1.92)$$

This weighting function is used if $krdg_partic = 1$ in the namelist. From Equation (1.91), the mean value of G for ice participating in ridging is a^* , as compared to $G^*/3$ for Equation (1.89). 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 [44] 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 [27]. 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 [19]. 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

$$f_m^{\text{area}} = \frac{H_R - H_L}{H_{\max} - H_{\min}}, \quad (1.93)$$

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

$$f_m^{\text{vol}} = \frac{(H_R)^2 - (H_L)^2}{(H_{\max})^2 - (H_{\min})^2}. \quad (1.94)$$

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 [44]:

$$g_R(h) \propto \exp[-(h - H_{\min})/\lambda] \quad (1.95)$$

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 (μ_{rdg}) is a tunable parameter with units . Thus the mean ridge thickness increases in proportion to $h_n^{1/2}$, as in [27]. The value $\mu = 4.0$ gives λ in the range 1–4 m for most ridged ice. Ice strengths with $\mu = 4.0$ and $krdg_redist = 1$ are roughly comparable to the strengths with $H^* = 50$ m and $krdg_redist = 0$.

From Equation (1.95) it can be shown that the fractional area going to category m as a result of ridging is

$$f_m^{\text{area}} = \exp[-(H_{m-1} - H_{\min})/\lambda] - \exp[-(H_m - H_{\min})/\lambda]. \quad (1.96)$$

The fractional volume going to category m is

$$f_m^{\text{vol}} = \frac{(H_{m-1} + \lambda) \exp[-(H_{m-1} - H_{\min})/\lambda] - (H_m + \lambda) \exp[-(H_m - H_{\min})/\lambda]}{H_{\min} + \lambda}. \quad (1.97)$$

Equations (1.96) and (1.97) replace Equations (1.93) and (1.94) 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 [19], R_{net} is given by

$$R_{\text{net}} = \frac{C_s}{2}(\Delta - |D_D|) - \min(D_D, 0), \quad (1.98)$$

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

$$R_{\text{net}} = \left[a_{P0} + \sum_{n=1}^N a_{Pn} \left(1 - \frac{1}{k_n} \right) \right] R_{\text{tot}}. \quad (1.99)$$

Given R_{net} from Equation (1.98), we use Equation (1.99) to compute R_{tot} . Then the area ridged in category n is given by $a_{rn} = r_n \Delta t$, where $r_n = a_{Pn} R_{\text{tot}}$. The area of new ridges is a_{rn}/k_n , and the volume of new ridges is $a_{rn} h_n$ (since volume is conserved during ridging). We remove the ridging ice from category n and use Equations (1.93) and (1.94): (or (1.96) and (1.97)) 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_{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_{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 [26]:

$$P = P^* h \exp[-C(1 - a_i)], \quad (1.100)$$

where $P^* = 27,500 \text{ N/m}$ 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 [57], 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

$$P = C_f C_p \beta \sum_{n=1}^{N_C} \left[-a_{Pn} h_n^2 + \frac{a_{Pn}}{k_n} \left(\frac{(H_n^{\text{max}})^3 - (H_n^{\text{min}})^3}{3(H_n^{\text{max}} - H_n^{\text{min}})} \right) \right], \quad (1.101)$$

where $C_P = (g/2)(\rho_i/\rho_w)(\rho_w - \rho_i)$, $\beta = R_{\text{tot}}/R_{\text{net}} > 1$ from Equation (1.99), and C_f is an empirical parameter that accounts for frictional energy dissipation. Following [19], 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:

$$P = C_f C_p \beta \sum_{n=1}^{N_C} \left[-a_{Pn} h_n^2 + \frac{a_{Pn}}{k_n} (H_{\min}^2 + 2H_{\min}\lambda + 2\lambda^2) \right] \quad (1.102)$$

The energy-based ice strength given by Equations (1.101) or (1.102) is more physically realistic than the strength given by Equation (1.100). However, use of Equation (1.100) is less likely to allow numerical instability at a given resolution and time step. See [44] for more details.

1.2.2.4.3 Thermodynamics

The current CICE version includes three thermodynamics options, the “zero-layer” thermodynamics of [60] ($ktherm = 0$), the Bitz and Lipscomb model [7] ($ktherm = 1$) that assumes a fixed salinity profile, and a new “mushy” formulation ($ktherm = 2$) in which salinity evolves [73]. For each thickness category, CICE 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, for $ktherm = 1$ and 2.

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

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

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

Melt ponds

Three explicit melt pond parameterizations are available in 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:

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

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 *Tracers* and *Figure 1*.

CESM formulation (*tr_pond_cesm* = true)

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

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

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 ($h_p \leq 0.9h_i$). The pond shape (*pondaspect*) $\delta_p = 0.8$ 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 [29].

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 [20][21][22]. 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.

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

Once a volume of water is produced from ice and snow melting, we calculate the number of ice categories covered by water. At each time step, we construct a list of volumes of water $\{V_{P1}, V_{P2}, \dots, V_{P,k-1}, V_{Pk}, V_{P,k+1}, \dots\}$, where V_{Pk} is the volume of water required to completely cover the ice and snow in the surface height categories from $i = 1$ up to $i = k$. The volume V_{Pk} is defined so that if the volume of water V_P is such that $V_{Pk} < V_P < V_{P,k+1}$ then the snow and ice in categories $n = 1$ up to $n = k + 1$ are covered in melt water. *Figure 6* (a) depicts the areas covered in melt

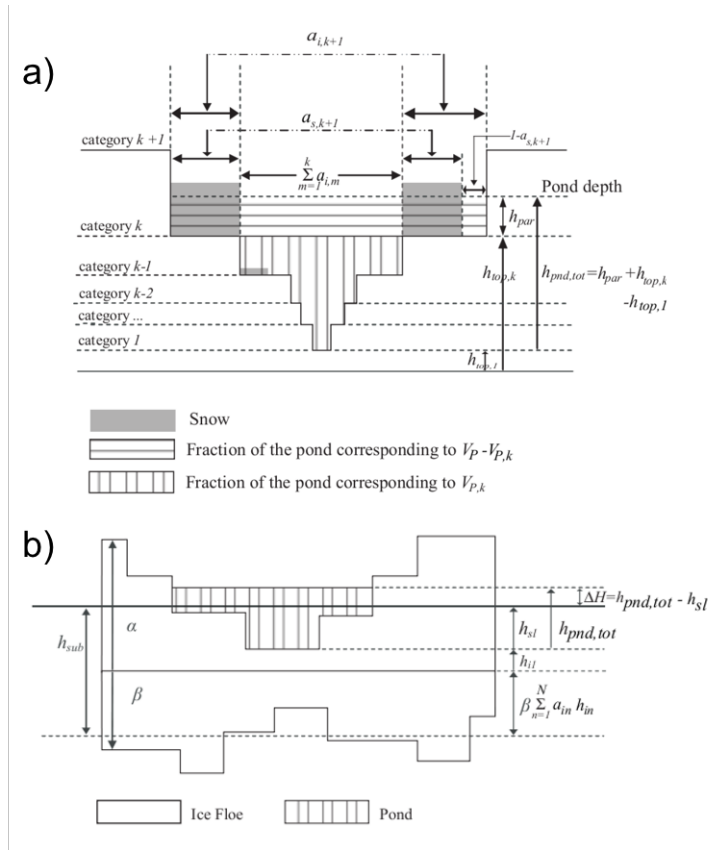


Fig. 1.6: Figure 6

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 n th category covered in snow is a_{sn} . The volume V_{P1} , which is the region with vertical hatching, is the volume of water required to completely fill up the first thickness category, so that any extra melt water must occupy the second thickness category, and it is given by the expression

$$V_{P1} = a_{i1}(h_{top,2} - h_{top,1}) - a_{s1}a_{i1}h_{s1}(1 - V_{sw}), \quad (1.103)$$

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

$$V_{P2} = a_{i1}(h_{top,3} - h_{top,2}) + a_{i2}(h_{top,3} - h_{top,2}) - a_{s2}a_{i2}h_{s2}(1 - V_{sw}) + V_{P1}. \quad (1.104)$$

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

$$V_{Pk} = \sum_{m=0}^k a_{im}(h_{top,k+1} - h_{top,k}) - a_{sk}a_{ik}h_{sk}(1 - V_{sw}) + \sum_{m=0}^{k-1} V_{Pm}. \quad (1.105)$$

(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

$$h_{pnd,tot} = h_{par} + h_{top,k} - h_{top,1}, \quad (1.106)$$

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

$$V_P - V_{Pk} = h_{par} \left(\sum_{m=1}^k a_{ik} + a_{i,k+1}(1 - a_{s,k+1}) + a_{i,k+1}a_{s,k+1}V_{sw} \right) \quad \text{for } h_{par} \leq h_{s,k+1}, \quad (1.107)$$

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

$$V_P - V_{Pk} = h_{par} \left(\sum_{m=1}^k a_{ik} + a_{i,k+1}(1 - a_{s,k+1}) \right) + a_{i,k+1}a_{s,k+1}V_{sw}h_{s,k+1} + a_{i,k+1}a_{s,k+1}(h_{par} - h_{s,k+1}) \quad \text{for } h_{par} > h_{s,k+1} \quad (1.108)$$

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 [\[16\]](#). The hydraulic head of melt water on sea ice (i.e., its height above sea level) drives flushing of melt water through the porous sea ice and into the underlying ocean. The mushy thermodynamics scheme ($ktherm = 2$) handles flushing. For $ktherm \neq 2$ we model the vertical flushing rate using Darcy's law for flow through a porous medium

$$w = -\frac{\Pi_v}{\mu} \rho_o g \frac{\Delta H}{h_i}, \quad (1.109)$$

where w is the vertical mass flux per unit perpendicular cross-sectional area (i.e., the vertical component of the Darcy velocity), Π_v is the vertical component of the permeability tensor (assumed to be isotropic in the horizontal), μ is the

viscosity of water, ρ_o is the ocean density, g is gravitational acceleration, ΔH is the the hydraulic head, and h_i is the thickness of the ice through which the pond flushes. As proposed by [25] the vertical permeability of sea ice can be calculated from the liquid fraction ϕ :

$$\Pi_v = 3 \times 10^{-8} \phi^3 \text{m}^2. \quad (1.110)$$

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 [18], the solid fraction is determined by:

$$\phi = \frac{c_i - S}{c_i - S_{br}(T)}, \quad (1.111)$$

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

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

$$h_{sl} = h_{sub} - 0.4 \sum_{n=1}^N a_{in} h_{in} - \beta h_{i1}, \quad (1.112)$$

where $0.4 \sum_{n=1}^N a_{in} h_{in}$ is the mean thickness of the basal depth classes, and h_{sub} is the depth of the submerged portion of the floe. Figure 6 (b) depicts the relationship between the hydraulic head and the depths and heights that appear in Equation (1.112). The depth of the submerged portion of the floe is determined from hydrostatic equilibrium to be

$$h_{sub} = \frac{\rho_m}{\rho_w} V_P + \frac{\rho_s}{\rho_w} V_s + \frac{\rho_i}{\rho_w} V_i, \quad (1.113)$$

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

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

$$\rho_i L_0 \frac{dh_{ipnd}}{dt} = k_i \frac{\partial T_i}{\partial z} - k_p \frac{\partial T_p}{\partial z}, \quad (1.114)$$

where L_0 is the latent heat of fusion of pure ice per unit volume, T_i and T_p are the ice surface and pond temperatures, and k_i and k_p are the thermal conductivity of the ice lid and pond respectively. The second term on the right hand-side is close to zero since the pond is almost uniformly at the freezing temperature [68]. Approximating the temperature gradient in the ice lid as linear, the Stefan condition yields the classic Stefan solution for ice lid depth

$$h_{ipnd} = \sqrt{\frac{2k_i}{\rho_s L} \Delta T_i t}, \quad (1.115)$$

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 [21], the pond volume was carried as an ice area tracer, but in [22] and here, pond area and thickness are carried as separate tracers, as in Section *Tracers*.

Unlike the cesm and level-ice melt pond schemes, the liquid pond water in the topo parameterization is not necessarily virtual; it can be withheld from being passed to the ocean model until the ponds drain by setting the namelist variable `l_mpond_fresh = .true.` The refrozen pond lids are still virtual. Extra code needed to track and enforce conservation of water has been added to **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 [35]. The ponds evolve according to physically based process descriptions, assuming a thickness-area ratio for changes in pond volume. A novel aspect of the new scheme is that the ponds are carried as tracers on the level (undeformed) ice area of each thickness category, thus limiting their spatial extent based on the simulated sea ice topography. This limiting is meant to approximate the horizontal drainage of melt water into depressions in ice floes. (The primary difference between the level-ice and topo meltpond parameterizations lies in how sea ice topography is taken into account when determining the areal coverage of ponds.) Infiltration of the snow by melt water postpones the appearance of ponds and the subsequent acceleration of melting through albedo feedback, while snow on top of refrozen pond ice also reduces the ponds' effect on the radiation budget.

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

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

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

$$\frac{\partial}{\partial t}(a_{lvl}a_i) + \nabla \cdot (a_{lvl}a_i\mathbf{u}) = 0, \quad (1.116)$$

$$\frac{\partial}{\partial t}(a_{pnd}a_{lvl}a_i) + \nabla \cdot (a_{pnd}a_{lvl}a_i\mathbf{u}) = 0, \quad (1.117)$$

$$\frac{\partial}{\partial t}(h_{pnd}a_{pnd}a_{lvl}a_i) + \nabla \cdot (h_{pnd}a_{pnd}a_{lvl}a_i\mathbf{u}) = 0, \quad (1.118)$$

$$\frac{\partial}{\partial t}(h_{ipnd}a_{pnd}a_{lvl}a_i) + \nabla \cdot (h_{ipnd}a_{pnd}a_{lvl}a_i\mathbf{u}) = 0. \quad (1.119)$$

(We have dropped the category subscript here, for clarity.) Equations (1.118) and (1.119) express conservation of melt pond volume and pond ice volume, but in this form highlight that the quantities tracked in the code are the tracers h_{pnd} and h_{ipnd} , pond depth and pond ice thickness. Likewise, the level ice fraction a_{lvl} is a tracer on ice area fraction (Equation (1.116)), and pond fraction a_{pnd} is a tracer on level ice (Equation (1.117)).

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 $frzrnd = 'hlid'$ 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

$$\rho_i L \frac{\partial h_i}{\partial t} = k_i \frac{\partial T_i}{\partial z} \sim k_i \frac{\Delta T}{h_i} \quad (1.120)$$

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

$$\Delta h_i = \begin{cases} \sqrt{\beta \Delta t}/2 & \text{if } h_i = 0 \\ \beta \Delta t / 2 h_i & \text{if } h_i > 0, \end{cases} \quad (1.121)$$

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,

$$\Delta h_i = -\min \left(\frac{\max(F_o, 0) \Delta t}{\rho_i L}, h_i \right), \quad (1.122)$$

where F_o 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 $frzrnd = 'cesm'$ 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} :

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

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 Equation (1.122))

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

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 $a_p = a_{pnd} a_{lvt}$ 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, $a_{pnd}^{eff} = a_{pnd} a_{lvt}$.

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 mushy thermodynamics scheme ($ktherm = 2$) handles flushing. For $ktherm \neq 2$, the permeability of the sea ice is calculated using the internal ice temperatures T_i (computed from the enthalpies as in the sea ice thermodynamics). The brine salinity and liquid fraction are given by [52] [eq 3.6] $S_{br} = 1/(10^{-3} - 0.054/T_i)$ and $\phi = S/S_{br}$, where S

is the bulk salinity of the combined ice and brine. The ice is considered permeable if $\phi \geq 0.05$ with a permeability of $p = 3 \times 10^{-8} \min(\phi^3)$ (the minimum being taken over all of the ice layers). A hydraulic pressure head is computed as $P = g\rho_w \Delta h$ where Δh is the height of the pond and sea ice above sea level. Then the volume of water drained is given by

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

where d_p is a scaling factor (dpscale), and $\mu = 1.79 \times 10^{-3} \text{ kg m}^{-1} \text{ s}^{-1}$ 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 $\Delta a_i = \Delta a_{lvl}$.

Thermodynamic surface forcing balance

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

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

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 [9]. Since publication of this technical paper, a number of improvements have been made to the Delta-Eddington scheme, including a surface scattering layer and internal shortwave absorption for snow, generalization for multiple snow layers and more than four layers of ice, and updated IOP values.

The namelist parameters R_{ice} and R_{pnd} adjust the albedo of bare or ponded ice by the product of the namelist value and one standard deviation. For example, if $R_{ice} = 0.1$, the albedo increases by 0.1σ . Similarly, setting $R_{snw} = 0.1$ decreases the snow grain radius by 0.1σ (thus increasing the albedo). Two additional tuning parameters are available for this scheme, dT_{melt} and $rsnw_{melt}$. dT_{melt} is the temperature change needed for a change in snow grain radius from non-melting to melting, and $rsnw_{melt}$ is the maximum snow grain radius when melting. An absorption coefficient for algae ($kalg$) may also be set. See [9] for details; the CESM melt pond and Delta-Eddington parameterizations are further explained and validated in [29].

Shortwave radiation: CCSM3

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

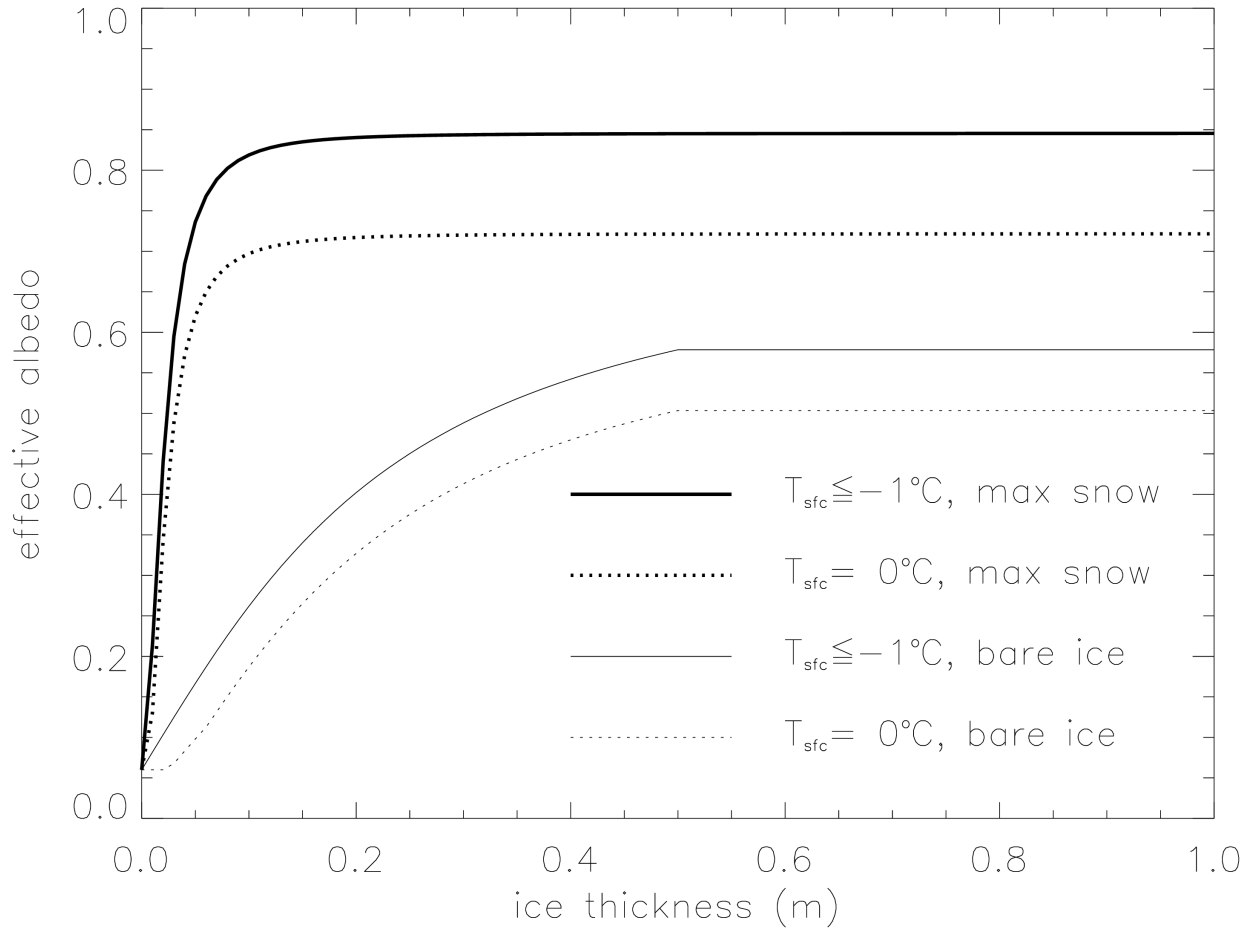


Fig. 1.7: Figure 7

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

The net absorbed shortwave flux is $F_{swabs} = \sum (1 - \alpha_j) F_{sw\downarrow}$, where the summation is over four radiative categories (direct and diffuse visible, direct and diffuse near infrared). The flux penetrating into the ice is $I_0 = i_0 F_{swabs}$, where

$i_0 = 0.70(1 - f_{snow})$ for visible radiation and $i_0 = 0$ for near IR. Radiation penetrating into the ice is attenuated according to Beer's Law:

$$I(z) = I_0 \exp(-\kappa_i z), \quad (1.123)$$

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 m^{-1} for visible wavelengths [15]. 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, $F_{L\uparrow} = \epsilon \sigma (T_{sf}^K)^4$, 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 $F_{L\uparrow}$.) The sensible heat flux is proportional to the difference between air potential temperature and the surface temperature of the snow or snow-free ice,

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

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

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

where $q_1 = 1.16378 \times 10^7 \text{ kg/m}^3$, $q_2 = 5897.8 \text{ K}$, 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 [48]:

$$F_{bot} = -\rho_w c_w c_h u_* (T_w - T_f), \quad (1.124)$$

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 $u_{*min} = 5 \times 10^{-4} \text{ m/s}$, 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 [49] and [64]. F_{bot} and the fraction of ice melting laterally are scaled so that $F_{bot} + F_{side} \geq F_{frzmlt}$ in the case that $F_{frzmlt} < 0$ (melting; see Section [Growth and melting](#)).

New temperatures

Zero-layer thermodynamics ($ktherm = 0$) An option for zero-layer thermodynamics [60] is available in this version of CICE by setting the namelist parameter `ktherm` to 0 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 other configurations, which we describe here.

Bitz and Lipscomb thermodynamics ($ktherm = 1$)

The “BL99” thermodynamic sea ice model is based on [50] and [7], and is described more fully in [41]. 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

$$S_{ik} = \frac{1}{2} S_{max} [1 - \cos(\pi z^{(\frac{a}{z+b})})], \quad (1.125)$$

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 [59]. 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 [50]. Equation (1.125) is fairly accurate for ice that has drained at the top surface due to summer melting. It is not a good approximation for cold first-year ice, which has a more vertically uniform salinity because it has not yet drained. However, the effects of salinity on heat capacity are small for temperatures well below freezing, so the salinity error does not lead to significant temperature errors.

Temperature updates.

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

$$\mathbf{Ax} = \mathbf{b}, \quad (1.126)$$

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

There are four general cases: (1) $T_{sf} < 0^{circ}C$, snow present; (2) $T_{sf} = 0^{circ}C$, snow present; (3) $T_{sf} < 0^{circ}C$, snow absent; and (4) $T_{sf} = 0^{circ}C$, snow absent. For case 1 we have one equation (the top row of the matrix) for the new surface temperature, N_s equations for the new snow temperatures, and N_i equations for the new ice temperatures. For cases 2 and 4 we omit the equation for the surface temperature, which is held at $0^{circ}C$, 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 [50]:

$$\rho_i c_i \frac{\partial T_i}{\partial t} = \frac{\partial}{\partial z} \left(K_i \frac{\partial T_i}{\partial z} \right) - \frac{\partial}{\partial z} [I_{pen}(z)], \quad (1.127)$$

where $\rho_i = 917 \text{ kg/m}^3$ 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 [53]

$$c_i(T, S) = c_0 + \frac{L_0 \mu S}{T^2}, \quad (1.128)$$

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

Following [74] and [50], the standard thermal conductivity ($conduct = \text{'MU71'}$) is given by

$$K_i(T, S) = K_0 + \frac{\beta S}{T}, \quad (1.129)$$

where $K_0 = 2.03 \text{ W/m/deg}$ is the conductivity of fresh ice and $\beta = 0.13 \text{ W/m/ppt}$ is an empirical constant. Experimental results [70] suggest that Equation (1.129) may not be a good description of the thermal conductivity of sea ice. In particular, the measured conductivity does not markedly decrease as T approaches $0^{circ}C$, but does decrease near the top surface (regardless of temperature).

An alternative parameterization based on the “bubbly brine” model of [55] for conductivity is available (*conduct* = ‘bubbly’):

$$K_i = \frac{\rho_i}{\rho_0} (2.11 - 0.011T + 0.09S/T), \quad (1.130)$$

where ρ_i and $\rho_0 = 917 \text{ kg/m}^3$ are densities of sea ice and pure ice. Whereas the parameterization in Equation (1.129) asymptotes to a constant conductivity of $2.03 \text{ W m}^{-1} \text{ K}^{-1}$ with decreasing T , K_i in Equation (1.130) continues to increase with colder temperatures.

The equation for temperature changes in snow is analogous to Equation (1.127), with $\rho_s = 330 \text{ kg/m}^3$, $c_s = c_0$, and $K_s = 0.30 \text{ W/m/deg}$ 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 Equation (1.127) to finite-difference form. The resulting equations are second-order accurate in space, except possibly at material boundaries, and first-order accurate in time. Before writing the equations in full we give finite-difference expressions for some of the terms.

First consider the terms on the left-hand side of Equation (1.127). 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

$$c_{ik} = c_0 + \frac{L_0 \mu S_{ik}}{T_{ik}^m T_{ik}^{m+1}}, \quad (1.131)$$

which ensures that energy is conserved during a change in temperature. This can be shown by using Equation (1.128) to integrate $c_i dT$ from T_{ik}^m to T_{ik}^{m+1} ; the result is $c_{ik}(T_{ik}^{m+1} - T_{ik}^m)$, where c_{ik} is given by Equation (1.131). The specific heat is a nonlinear function of T_{ik}^{m+1} , the unknown new temperature. We can retain a set of linear equations, however, by initially guessing $T_{ik}^{m+1} = T_{ik}^m$ and then iterating the solution, updating T_{ik}^{m+1} in Equation (1.131) with each iteration until the solution converges.

Next consider the first term on the right-hand side of Equation (1.127). The first term describes heat diffusion and is discretized for a given ice or snow layer k as

$$\frac{\partial}{\partial z} \left(K \frac{\partial T}{\partial z} \right) = \frac{1}{\Delta h} [K_k^* (T_{k-1}^{m+1} - T_k^{m+1}) - K_{k+1}^* (T_k^{m+1} - T_{k+1}^{m+1})], \quad (1.132)$$

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

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

which reduces to the appropriate values in the limits $K_k \gg K_{k-1}$ (or vice versa) and $h_k \gg h_{k-1}$ (or vice versa). The effective conductivity at the top (bottom) interface of the ice-snow column is given by $K^* = 2K/\Delta h$, where K and

Δh are the thermal conductivity and thickness of the top (bottom) layer. The second term on the RHS of Equation (1.127) is discretized as

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

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

$$F_0 = F_{ct}, \quad (1.133)$$

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

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

where T_{sf}^* is the surface temperature from the most recent iteration, and F_0^* and $(dF_0/dT_{sf})^*$ are functions of T_{sf}^* . We initialize $T_{sf}^* = T_{sf}^m$ and update it with each iteration. Thus we can rewrite Equation (1.133) 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

$$\left[\left(\frac{dF_0}{dT_{sf}} \right)^* - K_1^* \right] T_{sf}^{m+1} + K_1^* T_1^{m+1} = \left(\frac{dF_0}{dT_{sf}} \right)^* T_{sf}^* - F_0^*, \quad (1.134)$$

the first equation in the set of equations (1.126). The temperature change in ice/snow layer k is

$$\rho_k c_k \frac{(T_k^{m+1} - T_k^m)}{\Delta t} = \frac{1}{\Delta h_k} [K_k^* (T_{k-1}^{m+1} - T_k^{m+1}) - K_{k+1} (T_k^{m+1} - T_{k+1}^{m+1})], \quad (1.135)$$

where $T_0 = T_{sf}$ in the equation for layer 1. In tridiagonal matrix form, Equation (1.135) becomes

$$-\eta_k K_k T_{k-1}^{m+1} + [1 + \eta_k (K_k + K_{k+1})] T_k^{m+1} - \eta_k K_{k+1} T_{k+1}^{m+1} = T_k^m + \eta_k I_k, \quad (1.136)$$

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 Equation (1.126) is similar to Equation (1.136), 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 *Atmosphere*). In this case there is no surface flux equation. The top layer temperature is computed by an equation similar to Equation (1.136) 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. $T_{sf} \leq 0^{circ}C$.
2. The change in T_{sf} since the previous iteration is less than a prescribed limit, ΔT_{\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 ΔF_{\max} .

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

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

$$q_s(T) = -\rho_s(-c_0T + L_0).$$

Sea ice enthalpy is more complex, because of brine pockets whose salinity varies inversely with temperature. Since the salinity is prescribed, there is a one-to-one relationship between temperature and enthalpy. The specific heat of sea ice, given by Equation (1.128), includes not only the energy needed to warm or cool ice, but also the energy used to freeze or melt ice adjacent to brine pockets. Equation (1.128) can be integrated to give the energy δ_e required to raise the temperature of a unit mass of sea ice of salinity S from T to T' :

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

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

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

Multiplying by ρ_i to change the units from J/kg to J/m³ and adding a term for the energy needed to raise the meltwater temperature to T , we obtain the sea ice enthalpy:

$$q_i(T, S) = -\rho_i \left[c_0(T_m - T) + L_0 \left(1 - \frac{T_m}{T} \right) - c_w T_m \right] \quad (1.137)$$

Note that Equation (1.137) 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

$$\begin{aligned} a &= c_0, \\ b &= (c_w - c_0)T_m - \frac{q_i}{\rho_i} - L_0, \\ c &= L_0T_m. \end{aligned}$$

The other root is unphysical.

Mushy thermodynamics ($ktherm = 2$)

The “mushy” thermodynamics option treats the sea ice as a mushy layer [18] in which the ice is assumed to be composed of microscopic brine inclusions surrounded by a matrix of pure water ice. Both enthalpy and salinity are prognostic variables. The size of the brine inclusions is assumed to be much smaller than the size of the ice layers, allowing a continuum approximation: a bulk sea-ice quantity is taken to be the liquid-fraction-weighted average of that quantity in the ice and in the brine.

Enthalpy and mushy physics.

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

$$q = \phi q_{br} + (1 - \phi) q_i = \phi \rho_w c_w T + (1 - \phi)(\rho_i c_i T - \rho_i L_0) \quad (1.138)$$

where q_{br} is the brine enthalpy, q_i is the pure ice enthalpy, ρ_i and c_i are density and heat capacity of the ice, ρ_w and c_w are density and heat capacity of the brine and L_0 is the latent heat of melting of pure ice. We assume that the specific heats of the ice and brine are fixed at the values of cp_ice and cp_ocn , respectively. The enthalpy is the energy required to raise the temperature of the sea ice to T , including both sensible and latent heat changes. Since the sea ice contains salt, it usually will be fully melted at a temperature below $0^{circ}C$. Equations (1.137) and (1.138) are equivalent except for the density used in the term representing the energy required to bring the melt water temperature to $(\rho_i$ and ρ_w in equations (1.137) and (1.138), respectively).

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

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

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

Within the parameterizations of brine drainage the brine density is a function of brine salinity [52]:

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

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

The permeability of ice is computed from the liquid fraction as in [25]:

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

where ϕ_{Π} is 0.05.

The liquidus relation used in the mushy layer module is based on observations of [5]. A piecewise linear relation can be fitted to observations of Z (the ratio of mass of salt (in g) to mass of pure water (in kg) in brine) to the melting temperature: $Z = aT + b$. Salinity is the mass of salt (in g) per mass of brine (in kg) so is related to Z by

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

The data is well fitted with two linear regions,

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

where

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

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

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

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

Two stage outer iteration.

As for the BL99 thermodynamics [7] there are two qualitatively different situations that must be considered when solving for the vertical thermodynamics: the surface can be melting and at the melting temperature, or the surface can be colder than the melting temperature and not melting. In the BL99 thermodynamics these two situations were treated within the same iterative loop, but here they are dealt with separately. If at the beginning of the time step the ice surface is cold and not melting, we solve the ice temperatures assuming that this is also true at the end of the time step. Once we have solved for the new temperatures we test to see if the answer is consistent with this assumption. If the surface temperature is below the melting temperature then we have found the appropriate consistent solution. If the surface is above the melting temperature at the end of the initial solution attempt, we recalculate the new temperatures assuming the surface temperature is fixed at the melting temperature. Alternatively if the surface is at the melting temperature at the start of a time step, we assume initially that this is also the case at the end of the time step, solve for the new temperatures and then check that the surface conductive heat flux is less than the surface atmospheric heat flux as is required for a melting surface. If this is not the case, the temperatures are recalculated assuming the surface is colder than melting. We have found that solutions of the temperature equations that only treat one of the two qualitatively different solutions at a time are more numerically robust than if both are solved together. The surface state rarely changes qualitatively during the solution so the method is also numerically efficient.

Temperature updates.

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

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

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

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

The conductivity of the mush is given by

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

where $K_i = 2.3 \text{ Wm}^{-1}\text{K}^{-1}$ is the conductivity of pure ice and $K_{br} = 0.5375 \text{ Wm}^{-1}\text{K}^{-1}$ is the conductivity of the brine. The thermal conductivity of brine is a function of temperature and salinity, but here we take it as a constant value for the middle of the temperature range experienced by sea ice, -10°C [61], assuming the brine liquidus salinity at -10°C .

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

$$\frac{q_k^t - q_k^{t_0}}{\Delta t} = \frac{\frac{K_{k+1}^*}{\Delta z_{k+1}'} (T_{k+1}^t - T_k^t) - \frac{K_k^*}{\Delta z_k'} (T_k^t - T_{k-1}^t)}{\Delta h} \quad (1.139)$$

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

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

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

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

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

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

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

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

Salinity updates.

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

The salinity equation takes the form

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

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

A. Gravity drainage. Sea ice initially retains all the salt present in the sea water from which it formed. Cold temperatures near the top surface of forming sea ice result in higher brine salinities there, because the brine is always at its melting temperature. This colder, saltier brine is denser than the underlying sea water and the brine undergoes convective overturning with the ocean. As the dense, cold brine drains out of the ice, it is replaced by fresher seawater, lowering the bulk salinity of the ice. Following [73], gravity drainage is assumed to occur as two simultaneously operating modes: a rapid mode operating principally near the ice base and a slow mode occurring everywhere.

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

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

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

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

The upward advective flow also carries heat, contributing a term to the heat conservation Equation (1.139),

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

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

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

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

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

$$\tilde{w}(z) = w \left(\frac{Ra(z) - Ra_c}{Ra(z)} \right). \quad (1.140)$$

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

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

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

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

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

where

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

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

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

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

The decay constant, λ , is modeled as

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

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

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

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

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

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

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

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

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

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

Basal boundary condition.

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

$$(1 - \phi_i) L_0 \rho_i \frac{\partial h}{\partial t} = K \left. \frac{\partial T}{\partial z} \right|_i - K_w \left. \frac{\partial T}{\partial z} \right|_w \quad (1.141)$$

where $(1 - \phi_i)$ is the solid fraction of new ice formed and the right hand is the difference in heat flux at the ice–ocean interface between the ice side and the ocean side of the interface. However, with mushy layers there is usually no discontinuity in solid fraction across the interface, so $\phi_i = 1$ and Equation (1.141) cannot be used explicitly. To circumvent this problem we set the interface solid fraction to be 0.15, a value that reproduces observations. ϕ_i is a namelist parameter ($phi_i_mushy = 0.85$). The basal ice temperature is set to the liquidus temperature T_f of the ocean surface salinity.

Tracer consistency.

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

Growth and melting

Melting at the top surface is given by

$$q \delta h = \begin{cases} (F_0 - F_{ct}) \Delta t & \text{if } F_0 > F_{ct} \\ 0 & \text{otherwise} \end{cases} \quad (1.142)$$

where q is the enthalpy of the surface ice or snow layer¹ (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

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

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

$$q \delta h = (F_{cb} - F_{bot}) \Delta t, \quad (1.143)$$

where F_{bot} is given by Equation (1.124) 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 Equation (1.143) is the enthalpy of the bottom ice layer. If ice is growing, q is the enthalpy of new ice with temperature T_f and salinity S_{max} ($ktherm = 1$) or ocean surface salinity ($ktherm = 2$). This ice is added to the bottom layer.

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

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

$$(\rho L_v - q) \delta h = F_l \Delta t, \quad (1.144)$$

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

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

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

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

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

Lateral melting is accomplished by multiplying the state variables by $1 - r_{side}$, where r_{side} is the fraction of ice melted laterally [49][64], and adjusting the ice energy and fluxes as appropriate. We assume a floe diameter of 300 m.

Snow ice formation.

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

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

Thus the snow base lies below sea level when

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

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

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

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

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

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

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

1.3 User Guide

1.3.1 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 or OpenMP threads 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 processors in a distributed system. The parallelization scheme for CICE was designed so that MPI could be used for the coupling along with MPI, OpenMP or no parallelization internally. The internal parallelization method is set at compile time with the *NTASK* and *THRD* definitions 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.

1.3.1.1 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).

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 (**:math:‘langle‘std:math:‘rangle‘** works for most systems)

makedep.c perl script that determines module dependencies

script that sets up the run directory and compiles the code

modules based on “shared” code in CESM

shr_orb_mod.F90 orbital parameterizations

documentation

cicedoc.pdf this document

PDF/ PDF documents of numerous publications related to CICE

institution-specific modules

cice/ official driver for CICE v5 (LANL)

CICE.F90 main program

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

CICE_RunMod.F90_debug debugging version of **CICE_RunMod.F90**

ice_constants.F90 physical and numerical constants and parameters

sample diagnostic output files

input files that may be modified for other CICE configurations

col/ column configuration files

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

gx1/ $\langle 1^\circ \rangle$ displaced pole grid files

global_gx1.grid $\langle 1^\circ \rangle$ displaced pole grid (binary)

global_gx1.kmt $\langle 1^\circ \rangle$ land mask (binary)

ice.restart_file pointer for restart file name

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

ice_in_v4.1 namelist input data for default CICE v4.1 configuration

iced_gx1_v5.nc restart file used for initial condition

gx3/ $\langle 3^\circ \rangle$ displaced pole grid files

global_gx3.grid $\langle 3^\circ \rangle$ displaced pole grid (binary)

global_gx3.kmt $\langle 3^\circ \rangle$ land mask (binary)

global_gx3.grid.nc $\langle 3^\circ \rangle$ displaced pole grid ()

global_gx3.kmt.nc $\langle 3^\circ \rangle$ land mask ()

ice.restart_file pointer for restart file name

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

iced_gx3_v5.nc restart file used for initial condition

convert_restarts.f90 Fortran code to convert restart files from v4.1 to v5 (4 ice layers)

run_ice.<OS>.<SITE>.<machine> sample script for running on the given operating system

binary history and restart modules

ice_history_write.F90 subroutines with binary output

ice_restart.F90 read/write binary restart files

history and restart modules

ice_history_write.F90 subroutines with output

ice_restart.F90 read/write restart files

parallel I/O history and restart modules

ice_history_write.F90 subroutines with output using PIO

ice_pio.F90 subroutines specific to PIO

ice_restart.F90 read/write restart files using PIO

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

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

general CICE source code

handles most work associated with the aerosol tracers

handles most work associated with the age tracer

skeletal layer biogeochemistry

stability-based parameterization for calculation of turbulent ice–atmosphere fluxes

for decomposing global domain into blocks

evolves the brine height tracer

keeps track of what time it is

miscellaneous diagnostic and debugging routines

for distributing blocks across processors

decompositions, distributions and related parallel processing info

domain and block sizes

elastic-anisotropic-plastic dynamics component

elastic-viscous-plastic dynamics component

code shared by EVP and EAP dynamics

unit numbers for I/O

handles most work associated with the first-year ice area tracer

fluxes needed/produced by the model

routines to read and interpolate forcing data for stand-alone ice model runs

grid and land masks

initialization and accumulation of history output variables

history output of biogeochemistry variables

history output of form drag variables

history output of ridging variables

history output of melt pond variables
code shared by all history modules
namelist and initializations
utilities for managing ice thickness distribution
basic definitions of reals, integers, etc.
handles most work associated with the level ice area and volume tracers
mechanical redistribution component (ridging)
CESM melt pond parameterization
level-ice melt pond parameterization
topo melt pond parameterization
mixed layer ocean model
orbital parameters for Delta-Eddington shortwave parameterization
utilities for reading and writing files
driver for reading/writing restart files
code shared by all restart options
basic restoring for open boundary conditions
shortwave and albedo parameterizations
space-filling-curves distribution method
essential arrays to describe the state of the ice
routines for time stepping the major code components
zero-layer thermodynamics of [60]
multilayer thermodynamics of [7]
thermodynamic changes mostly related to ice thickness distribution
mushy-theory thermodynamics of:cite:THB13
code shared by all thermodynamics parameterizations
vertical growth rates and fluxes
driver for horizontal advection
horizontal advection via incremental remapping
driver for ice biogeochemistry and brine tracer motion
parameters and shared code for biogeochemistry and brine height
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/**

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

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

restart/ restart directory

iced_gx3_v5.nc 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/**

1.3.1.2 Grid, boundary conditions and masks

The spatial discretization is specialized for a generalized orthogonal B-grid as in [51] or [63]. 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 [33]. 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/viewvc/MITgcm/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 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 and regional 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 $\langle 3^\circ \rangle$ POP grid and land mask; **global_gx1.grid** and **global_gx1.kmt** contain the $\langle 1^\circ \rangle$ 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 ($\langle OS \rangle \cdot \langle SITE \rangle \cdot \langle machine \rangle$ = Linux.LANL.conejo). There are versions of the gx3 grid files available.

In CESM, the sea ice model may exchange coupling fluxes using a different grid than the computational grid. This functionality is activated using the namelist variable *gridcpl_file*.

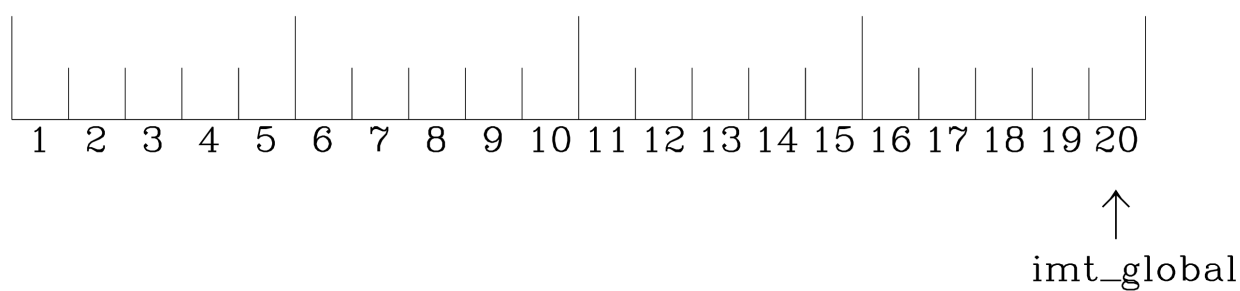
1.3.1.2.1 Grid domains and blocks

In general, the global gridded domain is $n_x_global \times n_y_global$, while the subdomains used in the block distribution are $n_x_block \times n_y_block$. 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.

Figure 8 : Grid parameters for a sample one-dimensional, 20-cell global domain decomposed into four local subdomains. Each local domain has one ghost (halo) cell on each side, and the physical portion of the local domains are labeled *ilo:ihi*. The parameter *n_x_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.

The user chooses a block size $BLCKX \times BLCKY$ 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 *Performance*. The script **comp_ice** computes the maximum number of blocks on each processor for typical

Global (Physical) Domain



Local Domain (nghost=1)

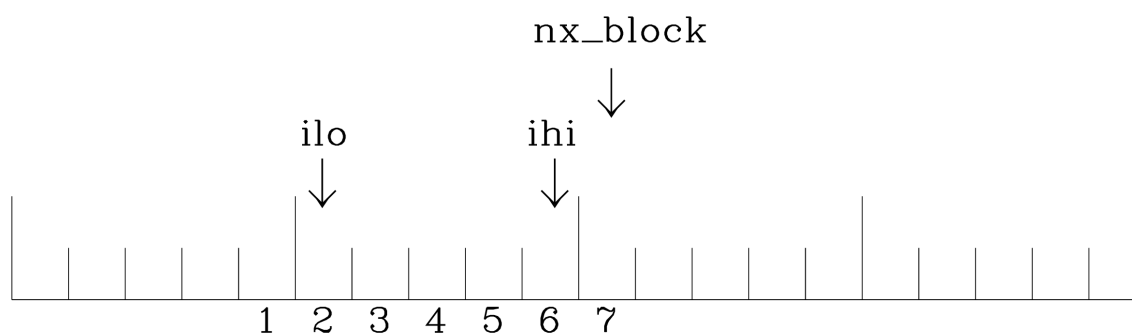


Fig. 1.8: Figure 8

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 *debug* 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 *debug* flag must be manually set in the code in each case (independently of the *debug* 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.

Alternatively, a new variable is provided in the history files, *blkmask*, which labels the blocks in the grid decomposition according to $blkmask = my_task + iblk/100$.

1.3.1.2.2 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 $nx_global/2$ and nx_global on the top U-row of the physical grid, and points with U-index i and $nx_global - i$ 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 (halo) 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 $nx_global - i + 1$ 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 $nx_global/2 + 1$ on the top T-row of the physical grid, and points with T-index i and $nx_global - i + 2$ 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 $nx_global - i + 1$ 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.

1.3.1.2.3 Bio-grid

The bio-grid is a vertical grid used for solving the brine height variable h_b . In the future, it will also be used for discretizing the vertical transport equations of biogeochemical tracers. The bio-grid is a non-dimensional vertical grid which takes the value zero at h_b and one at the ice–ocean interface. The number of grid levels is specified during compilation in **comp_ice** by setting the variable *NBGCLYR* equal to an integer (n_b).

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

1.3.1.2.4 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.

1.3.1.2.5 Boundary conditions

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.

Open/cyclic 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 halo 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; the restoring timescale *trestore* may be used (it is 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. The ‘displaced_pole’ grid option should not be used unless the regional grid contains land all along the north and south boundaries. The current form of the boundary condition routines does not allow Neumann boundary conditions, which must be set explicitly. This has been done in an unreleased branch of the code; contact Elizabeth for more information.

For exact restarts using restoring, set *restart_ext* = true in namelist to use the extended-grid subroutines.

On tripole grids, the order of operations used for calculating elements of the stress tensor can differ on either side of the fold, leading to round-off differences. Although restarts using the extended grid routines are exact for a given run, the solution will differ from another run in which restarts are written at different times. For this reason, explicit halo updates of the stress tensor are implemented for the tripole grid, both within the dynamics calculation and for restarts. This has not been implemented yet for tripoleT grids, pending further testing.

1.3.1.2.6 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.

Improved parallel performance may result from utilizing halo masks for boundary updates of the full ice state, incremental remapping transport, or for EVP or EAP dynamics. These options are accessed through the logical namelist flags *maskhalo_bound*, *maskhalo_remap*, and *maskhalo_dyn*, respectively. Only the halo cells containing needed information are communicated.

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.

1.3.1.3 Test configurations

1.3.1.4 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 of namelist options*), 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, core variables are read and reinitialized in *restartfile*, while tracer variables needed for specific configurations are read in separate restart routines associated with each tracer or specialized parameterization. Finally, albedo and other quantities dependent on the initial ice state are set. Some of these parameters will be described in more detail in *Table of namelist options*.

The restart files supplied with the code release include the core variables on the default configuration, that is, with seven vertical layers and the ice thickness distribution defined by *kcatbound* = 0. Restart information for some tracers is also included in the restart files.

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. With this arrangement, the tracer restart flags can be set to true or false, depending on whether the tracer restart data exist. With the second option, tracer restart flags are set to ‘continue’ for all active tracers.

An additional namelist option, *restart_ext* specifies whether halo cells are included in the restart files. This option is useful for tripole and regional grids, but can not be used with PIO.

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.

Table 4 : Ice initial state resulting from combinations of ice_ic, runtime and restart. ^aIf false, restart is reset to true. ^brestart is reset to false. ^cice_ic is reset to 'none.'

Table 1.4: Table 4

ice_ic			
	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

1.3.1.5 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 $ndtd$ are specified, and $dt_dyn = dt/ndtd$.

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

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

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

As discussed in section *Mechanical redistribution* and [44], 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 [57], as in Equation (1.101), limits the time step to ~ 30 minutes for the old ridging scheme ($krdg_partic = 0$), and to ~ 2 hours for the new scheme ($krdg_partic = 1$), assuming $\Delta x = 10$ 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.

In the classic EVP or EAP approach ($k_{dyn} = 1$ or 2 , $revised_evp = false$), 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_o (eyc), defines the elastic wave damping timescale T , described in Section *Dynamics*, 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 [32]. 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.

For the revised EVP approach ($k_{dyn} = 1$, $revised_evp = \text{true}$), the relaxation parameter $arlx1i$ effectively sets the damping timescale in the problem, and $brlx$ represents the effective subcycling [8]. In practice the parameters $S_e > 0.5$ and $\xi < 1$ are set, along with an estimate of the ice strength per unit mass, and the damping and subcycling parameters are then calculated. With the addition of the revised EVP approach to CICE, the code now uses these parameters internally for both classic and revised EVP configurations (see Section [Revised approach](#)).

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 (e.g., $\Delta t = 1$ day, given daily forcing) do not produce accurate results in the dynamics component. The reasons for this error are discussed in [32]; see [36] for its practical effects. The thermodynamics component is stable for any time step, as long as the surface temperature T_{sfc} is computed internally. The numerical constraint on the thermodynamics time step is associated with the transport scheme rather than the thermodynamic solver.

1.3.1.6 Model output

1.3.1.6.1 History files

Model output data is averaged over the period(s) given by *histfreq* and *histfreq_n*, and written to binary or 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 chosen in **comp_ice** (set *IO_TYPE*). The history files are CF-compliant; header information for data contained in the files is displayed with the command *nc-dump -h filename.nc*. Parallel output is available using the PIO library; the attribute *io_flavor* distinguishes output files written with PIO from those written with standard netCDF. 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 [History fields](#).

With this release, the history module has been divided into several modules based on the desired formatting and on the variables themselves. Parameters, variables and routines needed by multiple modules is in **ice_history_shared.F90**, while the primary routines for initializing and accumulating all of the history variables are in **ice_history.F90**. These routines call format-specific code in the **io_binary**, **io_netcdf** and **io_pio** directories. History variables specific to certain components or parameterizations are collected in their own history modules (**ice_history_bgc.F90**, **ice_history_drag.F90**, **ice_history_mechred.F90**, **ice_history_pond.F90**).

The history modules allow 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_⟨var⟩*, 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, *melth* 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, *melth* is called *melth* in the monthly file (for backward compatibility with the default configuration) and *melth_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 true 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)**. Several 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.

1.3.1.6.2 Diagnostic files

Like *histfreq*, the parameter *diagfreq* 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_in**.

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 [Timers](#) contains instructions for adding timers.

The timings provided by these timers are not mutually exclusive. For example, the column timer (5) includes the timings from 6–10, and subroutine *bound* (timer 15) 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.

Table 5 : CICE timers

Table 1.5: Table 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
16	BGC	biogeochemistry

1.3.1.6.3 Restart files

CICE now provides restart data in binary unformatted or formats, via the *IO_TYPE* flag in **comp_ice** and namelist variable *restart_format*. Restart and history files must use the same format. As with the history output, there is also an option for writing parallel restart files using PIO.

The restart files created by CICE contain all of the variables needed for a full, exact restart. The filename begins with the character string ‘iced.’, and the restart dump frequency is given by the namelist variables *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*. The code assumes that auxiliary binary tracer restart files will be identified using the same pointer and file name prefix, but with an additional character string in the file name that is associated with each tracer set. All variables are included in restart files.

Additional namelist flags provide further control of restart behavior. *dump_last* = true causes a set of restart files to be written at the end of a run when it is otherwise not scheduled to occur. The flag *use_restart_time* enables the user to choose to use the model date provided in the restart files. If *use_restart_time* = false then the initial model date stamp is determined from the namelist parameters. *lcdf64* = true sets 64-bit output, allowing larger file sizes with version 3.

Routines for gathering, scattering and (unformatted) reading and writing of the “extended” global grid, including the physical domain and ghost (halo) 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. Extended grid restarts are not available when using PIO; in this case extra halo update calls fill ghost cells for tripole grids (do not use PIO for regional grids).

Two restart files are included with the CICE v5 code distribution, for the gx3 and gx1 grids. They were created using the default model configuration (settings as in **comp_ice** and **ice_in**), but initialized with no ice. The gx3 case was run for 1 year using the 1997 forcing data provided with the code. The gx1 case was run for 20 years, so that the date of restart in the file is 1978-01-01. Note that the restart dates provided in the restart files can be overridden using the namelist variables *use_restart_time*, *year_init* and *istep0*. The forcing time can also be overridden using *fyear_init*.

Several changes in CICE v5 have made restarting from v4.1 restart files difficult. First, the ice and snow enthalpy state variables are now carried as tracers instead of separate arrays, and salinity has been added as a necessary restart field. Second, the default number of ice layers has been increased from 4 to 7. Third, netcdf format is now used for all I/O; it is no longer possible to have history output as and restart output in binary format. However, some facilities

are included with CICE v5 for converting v4.1 restart files to the new file structure and format, provided that the same number of ice layers and basic physics packages will be used for the new runs. See Section [Restarts](#) for details.

1.3.2 Execution procedures

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

1. Download the forcing data used for testing from the CICE-Consortium github page, <https://github.com/CICE-Consortium>.
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**’.
5. To clean the compile directory and start fresh, simply execute ‘/bin/rm -rf compile’ from the run directory.

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 [Initial setup](#).

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 (or binary) history output files **iceh_[timeID].nc (.da)**. Using the ⟨3°⟩ grid, the log file should be similar to **ice.log.:math:‘langle’OS:math:‘rangle’**, provided for the user’s convenience. These log files were created using MPI on 4 processors on the ⟨3°⟩ 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. Loops over blocks have been threaded throughout the code, so that their work will be divided among *OMP_NUM_THREADS* if *THRD* is ‘yes.’ 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 [Performance](#). 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.

[Table 6](#) : Configuration options available in **comp_ice**.

Table 1.6: Table 6

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
TRFY	0 or 1	set to 1 for first-year 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	0 or 1	number of aerosol tracers
TRBRINE	set to 1 for brine height tracer	
NBGCLYR	(integer)	number of vertical layers for biogeochemical transport
IO_TYPE	none/netcdf/pio	use 'none' if library is unavailable, 'pio' for PIO
DITTO	yes/no	for reproducible diagnostics
BARRIERS	yes/no	flushes MPI buffers during global scatters and gathers
THRD	yes/no	set to yes for OpenMP threaded parallelism
OMP_NUM_THREADS	(integer)	the number of OpenMP threads requested
NUMIN	(integer)	smallest unit number assigned to CICE files
NUMAX	(integer)	largest unit number assigned to CICE files

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).

A more flexible option is available for double-precision MPI calculations, using the namelist variable *bfbflag*. When true, this flag produces bit-for-bit identical diagnostics with different tasks, threads, blocks and grid decompositions.

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 [Index of primary variables and parameters](#). 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 *ndtd*, and for classic EVP, the number of EVP subcycles *ndte*; *dte* is then calculated in subroutine *init_evp*. The primary reason for doing it this way is to ensure that *ndte* is an integer. (This is done differently for *revised_evp* = true.; see Section [Dynamics](#)).

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 [Initialization and coupling](#) for more details. Restarts are exact for MPI or single processor runs.

1.3.2.1 Scripts

1.3.2.2 Directories

1.3.2.3 Local modifications

1.3.2.4 Forcing data

1.3.3 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 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 optimize the volume-to-surface ratio for communications.

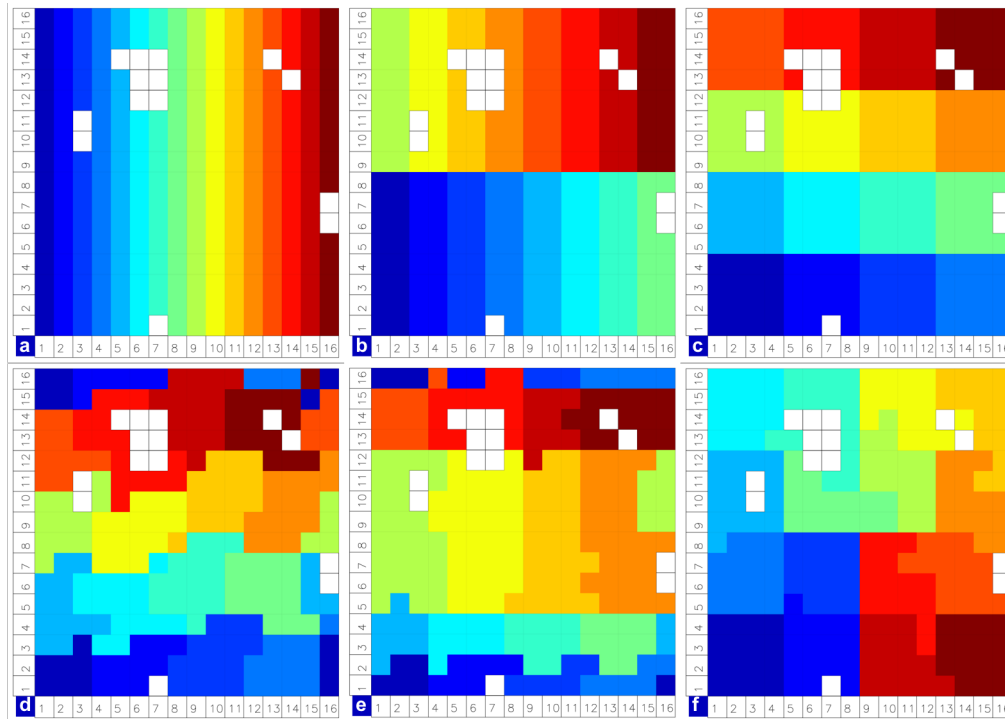


Fig. 1.9: Figure 9

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.

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. There are also three additional distribution types (‘roundrobin,’ ‘sectrobin,’ ‘sectcart’) that improve land-block elimination rates and also allow more flexibility in the number of processors used. 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 met, a Cartesian distribution is used instead.

While the Cartesian distribution groups sets of blocks by processor, the ‘roundrobin’ distribution loops through the blocks and processors together, putting one block on each processor until the blocks are gone. This provides good load balancing but poor communication characteristics due to the number of neighbors and the amount of data needed to communicate. The ‘sectrobin’ and ‘sectcart’ algorithms loop similarly, but put groups of blocks on each processor to improve the communication characteristics. In the ‘sectcart’ case, the domain is divided into two (east-west) halves and the loops are done over each, sequentially. [Figure 10](#) provides an overview of the pros and cons for the distribution types.

Figure 10 : Scorecard for block distribution choices in CICE, courtesy T. Craig. For more information, see <http://www.cesm.ucar.edu/events/ws.2012/Presentations/SEWG2/craig.pdf>

The *maskhalo* options in the namelist improve performance by removing unnecessary halo communications where there is no ice. There is some overhead in setting up the halo masks, which is done during the timestepping procedure as the ice area changes, but this option usually improves timings even for relatively small processor counts. T. Craig has found that performance improved by more than 20% for combinations of updated decompositions and masked haloes, in CESM’s version of CICE. A practical guide for choosing a CICE grid decomposition, based on experience in CESM, is available: <http://oceans11.lanl.gov/drupal/CICE/DecompositionGuide>

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.

Figure 11 illustrates the computational expense of various options, relative to the total time (excluding initialization) of a 7-layer configuration using BL99 thermodynamics, EVP dynamics, and the ‘ccsm3’ shortwave parameterization on the gx1 grid, run for one year from a no-ice initial condition. The block distribution consisted of 20×192 blocks spread over 32 processors (‘slenderX2’) with no threads and -O2 optimization. Timings varied by about $\pm 3\%$ in identically configured runs due to machine load. Extra time required for tracers has two components, that needed to carry the tracer itself (advection, category conversions) and that needed for the calculations associated with the particular tracer. The age tracers (FY and iage) require very little extra calculation, so their timings represent essentially the time needed just to carry an extra tracer. The topo melt pond scheme is slightly faster than the others because it calculates pond area and volume once per grid cell, while the others calculate it for each thickness category.

Figure 11 : Change in ‘TimeLoop’ timings from the 7-layer configuration using BL99 thermodynamics and EVP dynamics. Timings were made on a nondedicated machine, with variations of about $\pm 3\%$ in identically configured runs (light grey). Darker grey indicates the time needed for extra required options; The Delta-Eddington radiation scheme is required for all melt pond schemes and the aerosol tracers, and the level-ice pond parameterization additionally

Grading CICE Decompositions

decomposition	cice load balance north/south	cice load balance radiation	number of neighbors	amount of data to communicate	land block elimination in decomp	flexibility wrt pe counts
1. cartesian square-pop	F	F	B	A	F	C
2. cartesian slenderX1	A	B	A	C	F	F
3. cartesian slenderX2	B	C	B	B	F	F
4. spacecurve	D	D	B	B	A	A
5. roundrobin	A	A	D	D	A	A
6. sectrobin	B	A	C	C	A	A
7. sectcart	B	A	B	C	F	C

Fig. 1.10: Figure 10

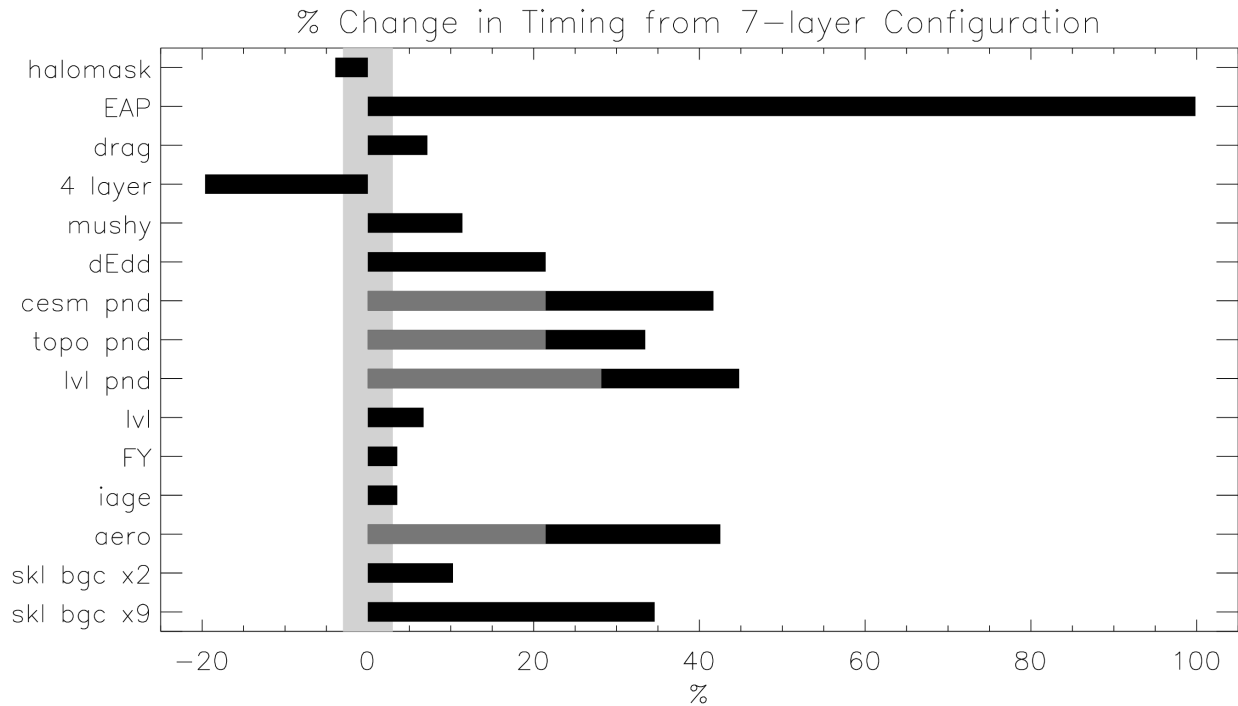


Fig. 1.11: Figure 11

requires the level-ice tracers.

1.3.4 Adding things

1.3.4.1 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. A use *ice_timers* statement may need to be added to the subroutine being modified. 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.

1.3.4.2 History fields

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

and in **ice_history.F90**:

1. broadcast the flag

2. add a call to `define_hist_field`
3. add a call to `accum_hist_field`

The example is for a standard, two-dimensional (horizontal) field; for other array sizes, choose another history variable with a similar shape as an example. Some history variables, especially tracers, are grouped in other files according to their purpose (bgc, melt ponds, etc.).

To add an output frequency for an existing variable, see section [History files](#).

1.3.4.3 Tracers

Each optional tracer has its own module, **ice_[tracer].F90**, which also contains as much of the additional tracer code as possible, and for backward compatibility of binary restart files, each new tracer has its own binary restart file. We recommend that the logical namelist variable `tr_[tracer]` be used for all calls involving the new tracer outside of **ice_[tracer].F90**, in case other users do not want to use that tracer.

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

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

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

1. **ice_domain_size.F90**: increase `max_ntrcr` (can also add option to **comp_ice** and **bld/Macros.***)
2. **ice_state.F90**: declare `nt_[tracer]` and `tr_[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**: (some of this may be done in **ice_[tracer].F90** instead)
 - add new module and `tr_[tracer]` to list of used modules and variables
 - add logical namelist variable `tr_[tracer]`
 - initialize namelist variable
 - 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, 2+`nt_[tracer]` for dependence on other tracers)
6. **ice_itd.F90**, **ice_mechred.F90**: Account for new dependencies if needed.
7. **CICE_InitMod.F90**: initialize tracer (includes reading restart file)
8. **CICE_RunMod.F90**, **ice_step_mod.F90**:

- call routine to write tracer restart data
 - call physics routines in **ice_[tracer].F90** (often called from **ice_step_mod.F90**)
9. **ice_restart.F90**: define restart variables (for binary, and PIO)
 10. **ice_history_[tracer].F90**: add history variables (Section *History fields*)
 11. **ice_in**: add namelist variables to *tracer_nml* and *icefields_nml*
 12. If strict conservation is necessary, add diagnostics as noted for topo ponds in Section *Melt ponds*.

1.3.5 Troubleshooting

Check the FAQ: <http://oceans11.lanl.gov/drupal/CICE/FAQ>.

1.3.5.1 Initial setup

The script **comp_ice** is configured so that the files **grid**, **kmt**, **ice_in**, **run_ice**, **iced_gx3_v5.0** 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 *NICELYR*, 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 (e.g., 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.

1.3.5.2 Restarts

CICE version 5 introduces a new model configuration that makes restarting from older simulations difficult. In particular, the number of ice categories, the category boundaries, and the number of vertical layers within each category must be the same in the restart file and in the run restarting from that file. Moreover, significant differences in the physics, such as the salinity profile, may cause the code to fail upon restart. Therefore, new model configurations may need to be started using `runtype = 'initial'`. Binary restart files that were provided with CICE v4.1 were made using the BL99 thermodynamics with 4 layers and 5 thickness categories (`kcatbound = 0`) and therefore can not be used for the default CICE v5 configuration (7 layers). In addition, CICE's default restart file format is now `netcdf` instead of binary.

Restarting a run using `runtype = 'continue'` requires restart data for all tracers used in the new run. If tracer restart data is not available, use `runtype = 'initial'`, setting `ice_ic` to the name of the core restart file and setting to true the namelist restart flags for each tracer that is available. The unavailable tracers will be initialized to their default settings.

On tripole grids, use `restart_ext = true` when using either binary or regular (non-PIO) `netcdf`.

Provided that the same number of ice layers (default: 4) will be used for the new runs, it is possible to convert v4.1 restart files to the new file structure and then to `netcdf` format. If the same physical parameterizations are used, the code should be able to execute from these files. However if different physics is used (for instance, mushy thermo instead of BL99), the code may still fail. To convert a v4.1 restart file:

1. Edit the code `input_templates/convert_restarts.F90` for your model configuration and path names. Compile and run this code to create a binary restart file that can be read using v5. Copy the resulting file to the `restart/` subdirectory in your working directory.
2. In your working directory, turn off all tracer restart flags in `ice_in` and set the following:
 - `runtype = 'initial'`
 - `ice_ic = './restart/[your binary file name]'`
 - `restart = .true.`
 - `use_restart_time = .true.`
3. In `CICE_InitMod.F90`, comment out the call to `restartfile(ice_ic)` and uncomment the call to `restart_file_v4(ice_ic)` immediately below it. This will read the v4.1 binary file and write a v5 file containing the same information.

If restart files are taking a long time to be written serially (i.e., not using PIO), see the next section.

1.3.5.3 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` or `ice_dyn_eap.F90`. You will take a hit for the extra computations, but it will not be as bad as running with the underflows.

In some configurations, multiple calls to scatter or gather global variables may overflow MPI's buffers, causing the code to slow down (particularly when writing large output files such as restarts). To remedy this problem, set `BARRIERS` yes in `comp_ice`. This synchronizes MPI messages, keeping the buffers in check.

1.3.5.4 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_RunMod.F90_debug`, which can be substituted for `CICE_RunMod.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.

1.3.5.5 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 old CCSM radiative scheme (*shortwave* = 'default' or 'ccsm3'), a sizable fraction (more than 10%) of the total shortwave radiation is absorbed at the surface but should be penetrating into the ice interior instead. This is due to use of the aggregated, effective albedo rather than the bare ice albedo when *snowpatch* < 1.
3. The date-of-onset diagnostic variables, *melt_onset* and *frz_onset*, are not included in the core 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.
8. Latitude and longitude fields in the history output may be wrong when using padding.

1.3.5.6 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 thermodynamic 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.

1.3.5.7 Proliferating subprocess parameterizations

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

1.3.6 Testing CICE

Version 6, August 2017 This documents how to use the testing features developed for the CICE Consortium CICE sea ice model.

1.3.6.1 Individual tests and test suites

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

```
> create.case -t smoke -m wolf -g gx3 -p 8x2 -s diag1,run5day -testid myid
```

where -m designates a specific machine. Test suites are multiple tests that are specified in an input file and are started on the command line like

```
> create.case -ts base_suite -m wolf -testid myid
```

create.case with -t or -ts require a testid to uniquely name test directories. The format of the case directory name for a test will always be `${machine}_${test}_${grid}_${pes}_${soptions}.${testid}`

To build and run a test, the process is the same as a case, cd into the test directory,

```
run cice.build
run cice.submit
```

The test results will be generated in a local file called “test_output”.

When running a test suite, the create.case command line automatically generates all the tests under a directory names `${test_suite}.${testid}`. It then automatically builds and submits all tests. When the tests are complete, run the results.csh script to see the results from all the tests.

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

smoke - Runs the model for default length. The length and options can be set with the -s command line option. The test passes if the model completes successfully.

restart - Runs the model for 10 days, writing a restart file at day 5 and again at day 10. Runs the model a second time starting from the day 5 restart and writing a restart at day 10 of the model run. The test passes if both the 10 day and 5 day restart run complete and if the restart files at day 10 from both runs are bit-for-bit identical.

Please run ‘./create.case -h’ for additional details.

1.3.6.2 Additional testing options

There are several additional options on the create.case command line for testing that provide the ability to regression test and compare tests to each other.

- bd defines a baseline directory where tests can be stored for regression testing
- bg defines a version name that where the current tests can be saved for regression testing
- bc defines a version name that the current tests should be compared to for regression testing
- td provides a way to compare tests with each other

To use -bg,

```
> create.case -ts base_suite -m wolf -testid v1 -bg version1 -bd $SCRATCH/CICE_BASELINES
```

will copy all the results from the test suite to \$SCRATCH/CICE_BASELINES/version1.

To use -bc,

```
> create.case -ts base_suite -m wolf -testid v2 -bc version1 -bd $SCRATCH/CICE_BASELINES
```

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

-bc and -bg can be combined,

```
> create.case -ts base_suite -m wolf -testid v2 -bg version2 -bc version1 -bd $SCRATCH/CICE_BASELINES
```

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

-bg, -bc, and -bd are used for regression testing. There is a default -bd on each machine.

-td allows a user to compare one test result to another. For instance, `> create.case -t smoke -m wolf -g gx3 -p 8x2 -s run5day -testid t01` `> create.case -t smoke -m wolf -g gx3 -p 4x2 -s run5day -testid t01 -td smoke_gx3_8x2_run5day`

An additional check will be done for the second test (because of the -td argument), and it will compare the output from the first test “smoke_gx3_8x2_run5day” to the output from it’s test “smoke_gx3_4x2_run5day” and generate a result for that. It’s important that the first test complete before the second test is done. Also, the -td option works only if the testid and the machine are the same for the baseline run and the current run.

1.3.6.3 Test suite format

The format for the test suite file is relatively simple. It is a text file with white space delimited columns like,

Table 1.7: Table 7

#Test	Grid	PEs	Sets	BFB-compare
smoke	gx3	8x2	diag1,run5day	
smoke	gx3	8x2	diag24,run1year,medium	
smoke	gx3	4x1	debug,diag1,run5day	
smoke	gx3	8x2	debug,diag1,run5day	
smoke	gx3	4x2	diag1,run5day	smoke_gx3_8x2_diag1_run5day
smoke	gx3	4x1	diag1,run5day,thread	smoke_gx3_8x2_diag1_run5day
smoke	gx3	4x1	diag1,run5day	smoke_gx3_4x1_diag1_run5day_thread
restart	gx3	8x1		
restart	gx3	4x2	debug	

The first column is the test name, the second the grid, the third the pe count, the fourth column is the -s options and the fifth column is the -td argument. The fourth and fifth columns are optional. The argument to -ts defines which filename to choose and that argument can contain a path. create.case will also look for the filename in configuration/scripts/tests where some preset test suites are defined.

1.3.6.4 Example Tests (Quickstart)

1.3.6.4.1 To generate a baseline dataset for a test case

```
./create.case -t smoke -m wolf -bg cicev6.0.0 -testid t00
cd wolf_smoke_gx3_4x1.t00
./cice.build
./cice.submit
# After job finishes, check output
cat test_output
```

1.3.6.4.2 To run a test case and compare to a baseline dataset

```
./create.case -t smoke -m wolf -bc cicev6.0.0 -testid t01
cd wolf_smoke_gx3_4x1.t01
./cice.build
./cice.submit
# After job finishes, check output
cat test_output
```

1.3.6.4.3 To run a test suite to generate baseline data

```
./create.case -m wolf -ts base_suite -testid t02 -bg cicev6.0.0bs
cd base_suite.t02
# Once all jobs finish, concatenate all output
./results.csh # All tests results will be stored in results.log
# To plot a timeseries of “total ice extent”, “total ice area”, and “total ice volume”
./timeseries.csh <directory>
ls *.png
```

1.3.6.4.4 To run a test suite to compare to baseline data

```
./create.case -m wolf -ts base_suite -testid t03 -bc cicev6.0.0bs
cd base_suite.t03
# Once all jobs finish, concatenate all output
./results.csh # All tests results will be stored in results.log
# To plot a timeseries of “total ice extent”, “total ice area”, and “total ice volume”
./timeseries.csh <directory>
ls *.png
```

1.3.6.4.5 To compare to another test

First:

```
./create.case -m wolf -t smoke -testid t01 -p 8x2
```

```
cd wolf_smoke_gx3_8x2.t01
```

```
./cice.build
```

```
./cice.submit
```

After job finishes, check output

```
cat test_output
```

Then, do the comparison:

```
./create.case -m wolf -t smoke -testid t01 -td smoke_gx3_8x2 -s thread -p 4x1
```

```
cd wolf_smoke_gx3_4x1_thread.t01
```

```
./cice.build
```

```
./cice.submit
```

After job finishes, check output

```
cat test_output
```

1.3.6.4.6 Additional Details

- In general, the baseline generation, baseline compare, and test diff are independent.
- **Use the ‘-bd’ flag to specify the location where you want the baseline dataset** to be written. Without specifying ‘-bd’, the baseline dataset will be written to the default baseline directory found in the env.<machine> file (ICE_MACHINE_BASELINE).
- **If ‘-bd’ is not passed, the scripts will look for baseline datasets in the default** baseline directory found in the env.<machine> file (ICE_MACHINE_BASELINE). If the ‘-bd’ option is passed, the scripts will look for baseline datasets in the location passed to the -bd argument.
- **To generate a baseline dataset for a specific version (for regression testing),** use ‘-bg <version_name>’. The scripts will then place the baseline dataset in \$ICE_MACHINE_BASELINE/<version_name>/
- **The ‘-testid’ flag allows users to specify a testing id that will be added to the** end of the case directory. For example, “./create.case -m wolf -t smoke -testid t12 -p 4x1” creates the directory wolf_smoke_gx3_4x1.t12. This flag is REQUIRED if using -t or -ts.

1.3.6.5 Code compliance test

Additions and changes to CICE and Icepack are expected to be bit-for-bit unless there is a strong justification for non-reproducibility, such as a bug-fix or approved scientific alteration to existing code. However, situations do arise when additions to CICE or Icepack are not bit-for-bit, but are also not expected to change the science of CICE and Icepack. In that instant, further evidence is required in the initial testing phase to support the premise that code changes have not altered the science of the model. To support this testing, a *t*-test is being/has been implemented in the CICE testing infrastructure.

1.3.6.5.1 Method

Welch's two-sided t -test is used to help determine whether or not two different simulations that should be identical are significantly different for any grid cell of the CICE gx-1/3 domain for grid-cell averaged sea ice thickness, h , ice concentration, c , and pack velocity components $\mathbf{u} = \mathbf{u}(u, v)$. In this circumstance, we seek to determine whether or not the null hypothesis, H_0 , is true. The null hypothesis is: Two simulations that are not bit-for-bit identical are ostensibly the same at every model grid point. The test begins from the standpoint that two CICE simulations *should* be bit-for-bit but are suspected of only being different at the level of computational innaccuracy. Therefore, we seek to limit a t -test Type II error, where a test would erroneously confirm the null hypothesis, H_0 . To that end, we choose to test the hypothesis that grid-point means from CICE simulation 'a' are different from CICE simulation 'b' at a relatively low confidence interval. Formally, we test the hypothesis $H_0 : \bar{x}_a = \bar{x}_b$, $H_1 : \bar{x}_a \neq \bar{x}_b$ for each of the aforementioned variables at every model grid point using a two-sided t -test with a 68, 80 and 95% confidence interval. Here, $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ is the time series mean of n samples x_i representing h , c , u or v , and daily samples are used from 5-year stand-alone CICE simulations. More frequent output is unnecessary, because each of h , c , u and v typically have a high degree of auto-correlation in sea ice models.

Due to the strong auto-correlation in geo-located sea ice time series, we calculate a two-sided t -statistic to compare \bar{x}_a and \bar{x}_b , given their respective standard deviations, σ_a and σ_b , and effective sample sizes, n'_a and n'_b , following [75]:

$$t = \frac{\bar{x}_a - \bar{x}_b}{\sqrt{\frac{\sigma_a^2}{n'_a} + \frac{\sigma_b^2}{n'_b}}}. \quad (1.145)$$

The null hypothesis $H_0 : \bar{x}_a = \bar{x}_b$ is true when

$$-t_{crit}(1-\alpha/2, N) < t < t_{crit}(1-\alpha/2, N) \quad (1.146)$$

for critical t -distribution values, t_{crit} , at the α significance level for effective degrees of freedom $N = n'_a + n'_b - 2$. At the 80% confidence interval, $\alpha = 0.20$, with corresponding tabulated values of $t_{crit}(0.9, N)$ obtained from a t -distribution look-up table. From [79], we use an unbiased standard deviation estimate,

$$\sigma = \sqrt{\frac{1}{n' - 1} \sum_{i=1}^n (x_i - \bar{x})^2}, \quad (1.147)$$

for the effective sample size,

$$n' \approx n \frac{1 - r_1}{1 + r_1}, \quad (1.148)$$

where r_1 is the lag-1 autocorrelation given by:

$$r_1 = \frac{\sum_{i=1}^{n-1} [(x_i - \bar{x}_{1:n-1})(x_{i+1} - \bar{x}_{2:n})]}{\sqrt{\sum_{i=1}^{n-1} (x_i - \bar{x}_{1:n-1})^2 \sum_{i=2}^n (x_i - \bar{x}_{2:n})^2}}. \quad (1.149)$$

In equation (1.149), $\bar{x}_{1:n-1}$ is the mean of all samples except the last, and $\bar{x}_{2:n}$ is the mean of samples except the first, and both differ from the overall mean \bar{x} in equations (1.145) and (1.147), which we repeat here for clarity:

$$\bar{x}_{1:n-1} = \frac{1}{n-1} \sum_{i=1}^{n-1} x_i, \quad \bar{x}_{2:n} = \frac{1}{n-1} \sum_{i=2}^n x_i, \quad \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad (1.150)$$

In applying equations (1.145) through (1.150), we are accounting for the fact, however imperfectly, that a t -test should be a comparison of the means from two series of independent samples. The typical affect of applying these equations to sea ice model output is that $n' \ll n$. For that reason, we need a lengthy time series to narrow the range of acceptable

values in (1.146). There is little point in using more frequent output from CICE than daily instantaneous values, since this would have little impact on decreasing r_1 in (1.149).

Using these equations, a standard procedure in testing for science-changing answers in CICE and Icepack is as follows: First, make every attempt to obtain bit-for-bit reproducibility in the model code. Once all available software-testing options have been exhausted, and the source of the bit-for-bit test failure has been pinpointed, proceed with the t -test documented above if the expectation is that code alterations should not be science-changing. Equations (1.145) through (1.150) are implemented in the reverse order from which they are presented here, and applied individually to daily samples of h , c , u and v from 5-year time series at every model grid point: i) Calculate $\bar{x}_{1:n-1}$, $\bar{x}_{2:n}$, and \bar{x} in (1.150) for simulations a and b ; ii) Compute (1.149), (1.148) and (1.147), in that order, for each simulation a and b , and finally; iii) Determine whether the null hypothesis is true at each model grid point in (1.146) using equation (1.145) and a lookup t -distribution table. Should H_0 be confirmed at each grid point, and for each variable h , c , u and v , this test contributes to evidence that changes to CICE and Icepack code are unlikely to alter scientific results. To guard against the possibility of a Type II error, the test should be performed for several different confidence intervals, nominally set at 68, 80 and 95%, the first and last of these values corresponding to σ and 2σ tests.

1.3.6.5.2 Practical Testing Procedure

To be placed here: Write up of how to actually do this test within the testing software to be added by Elizabeth, Rick, Matt, Tony et al. . . .

Implementation notes: 1) Provide a pass/fail on each of the confidence intervals, 2) Facilitate output of a bitmap for each test so that locations of failures can be identified.

1.3.7 Table of namelist options

Table 1.8: Table 8

variable	options/format	description	recommended value
<i>setup_nml</i>			
		<i>Time, Diagnostics</i>	
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	
ndtd	integer	number of dynamics/advection/ridging/steps per thermo timestep	1
		<i>Initialization/Restarting</i>	
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.
use_restart_time	true/false	set initial date using restart file	.true.

Continued on next page

Table 1.8 – continued from previous page

variable	options/format	description	recommended value
restart_format	nc	read/write restart files (use with PIO)	
	bin	read/write binary restart files	
lcdf64	true/false	if true, use 64-bit format	
restart_dir	path/	path to restart directory	
restart_ext	true/false	read/write halo cells in restart files	
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	
		<i>Model Output</i>	
bfbflag	true/false	for bit-for-bit diagnostic output	
diagfreq	integer	frequency of diagnostic output in dt	24
	e.g., 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	
debug	true/false	if true, write extra diagnostics	.false.
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 (not written)	
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’
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)	
<i>grid_nml</i>			
		<i>Grid</i>	
grid_format	nc	read grid and kmt files	‘bin’
	bin	read direct access, binary file	
grid_type	rectangular	defined in <i>rectgrid</i>	

Continued on next page

Table 1.8 – continued from previous page

variable	options/format	description	recommended value
	displaced_pole	read from file in <i>popgrid</i>	
	tripole	read from file in <i>popgrid</i>	
	regional	read from file in <i>popgrid</i>	
grid_file	filename	name of grid file to be read	‘grid’
kmt_file	filename	name of land mask file to be read	‘kmt’
gridcpl_file	filename	input file for coupling grid info	
kcatbound	0	original category boundary formula	0
	1	new formula with round numbers	
	2	WMO standard categories	
	-1	one category	
<i>domain_nml</i>			
		<i>Domain</i>	
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	
	roundrobin	1 block per proc until blocks are used	
	sectcart	blocks distributed to domain quadrants	
	sectrobin	several blocks per proc until used	
	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	
ew_boundary_type	cyclic	periodic boundary conditions in x-direction	
	open	Dirichlet boundary conditions in x	
ns_boundary_type	cyclic	periodic boundary conditions in y-direction	
	open	Dirichlet boundary conditions in y	
	tripole	U-fold tripole boundary conditions in y	
	tripoleT	T-fold tripole boundary conditions in y	
maskhalo_dyn	true/false	mask unused halo cells for dynamics	
maskhalo_remap	true/false	mask unused halo cells for transport	
maskhalo_bound	true/false	mask unused halo cells for boundary updates	
<i>tracer_nml</i>			
		<i>Tracers</i>	
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	

Continued on next page

Table 1.8 – continued from previous page

variable	options/format	description	recommended value
restart_pond_top	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_aero	true/false	aerosols	
restart_aero	true/false	restart tracer values from file	
<i>thermo_nml</i>			
		<i>Thermodynamics</i>	
kitd	0	delta function ITD approximation	1
	1	linear remapping ITD approximation	
ktherm	0	zero-layer thermodynamic model	
	1	Bitz and Lipscomb thermodynamic model	
	2	mushy-layer thermodynamic model	
conduct	MU71	conductivity [50]	
	bubbly	conductivity [55]	
a_rapid_mode	real	brine channel diameter	0.5×10^{-3} m
Rac_rapid_mode	real	critical Rayleigh number	10
aspect_rapid_mode	real	brine convection aspect ratio	1
dSdt_slow_mode	real	drainage strength parameter	-1.5×10^{-7} m/s/K
phi_c_slow_mode	$0 < \phi_c < 1$	critical liquid fraction	0.05
phi_i_mushy	$0 < \phi_i < 1$	solid fraction at lower boundary	0.85
<i>dynamics_nml</i>			
		<i>Dynamics</i>	
kdyn	0	dynamics OFF	1
	1	EVP dynamics	
	2	EAP dynamics	
revised_evp	true/false	use revised EVP formulation	
ndte	integer	number of EVP subcycles	120
advection	remap	linear remapping advection	‘remap’
	upwind	donor cell advection	
kstrength	0	ice strength formulation [26]	1
	1	ice strength formulation [57]	
krdg_partic	0	old ridging participation function	1
	1	new ridging participation function	
krdg_redist	0	old ridging redistribution function	1
	1	new ridging redistribution function	
mu_rdg	real	e-folding scale of ridged ice	
Cf	real	ratio of ridging work to PE change in ridging	17.
<i>shortwave_nml</i>			
		<i>Shortwave</i>	
shortwave	default	NCAR CCSM3 distribution method	
	dEdd	Delta-Eddington method	
albedo_type	default	NCAR CCSM3 albedos	‘default’
	constant	four constant albedos	
albicev	$0 < \alpha < 1$	visible ice albedo for thicker ice	

Continued on next page

Table 1.8 – continued from previous page

variable	options/format	description	recommended value
albice_i	$0 < \alpha < 1$	near infrared ice albedo for thicker ice	
albsnow_v	$0 < \alpha < 1$	visible, cold snow albedo	
albsnow_i	$0 < \alpha < 1$	near infrared, cold snow albedo	
ahmax	real	albedo is constant above this thickness	0.3 m
R_ice	real	tuning parameter for sea ice albedo from Delta-Eddington shortwave	
R_pnd	real	... for ponded sea ice albedo ...	
R_snw	real	... for snow (broadband albedo) ...	
dT_mlt	real	Δ temperature per Δ snow grain radius	
rsnw_mlt	real	maximum melting snow grain radius	
kalg	real	absorption coefficient for algae	
<i>ponds_nml</i>			
<i>Melt Ponds</i>			
hpl	real	critical ice lid thickness for topo ponds	0.01 m
hs0	real	snow depth of transition to bare sea ice	0.03 m
hs1	real	snow depth of transition to pond ice	0.03 m
dpscale	real	time scale for flushing in permeable ice	1×10^{-3}
frz_pnd	hlid	Stefan refreezing with pond ice thickness	'hlid'
	cesm	CESM refreezing empirical formula	
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
<i>zbgc_nml</i>			
<i>Biogeochemistry</i>			
tr_brine	true/false	brine height tracer	
tr_zaero	true/false	vertical aerosol tracers	
modal_aero	true/false	modal aerosols	
restore_bgc	true/false	restore bgc to data	
“solve_zsal”	true/false	update salinity tracer profile	
bgc_data_dir	path/	data directory for bgc	
skl_bgc	true/false	biogeochemistry	
sil_data_type	default	default forcing value for silicate	
	clim	silicate forcing from ocean climatology [23]	
nit_data_type	default	default forcing value for nitrate	
	clim	nitrate forcing from ocean climatology [23]	
	sss	nitrate forcing equals salinity	
fe_data_type	default	default forcing value for iron	
	clim	iron forcing from ocean climatology	
bgc_flux_type	Jin2006	ice–ocean flux velocity of [37]	
	constant	constant ice–ocean flux velocity	
restart_bgc	true/false	restart tracer values from file	
tr_bgc_C_sk	true/false	algal carbon tracer	
tr_bgc_chl_sk	true/false	algal chlorophyll tracer	
tr_bgc_Am_sk	true/false	ammonium tracer	
tr_bgc_Sil_sk	true/false	silicate tracer	
tr_bgc_DMSPp_sk	true/false	particulate DMSP tracer	

Continued on next page

Table 1.8 – continued from previous page

variable	options/format	description	recommended value
tr_bgc_DMSPd_sk	true/false	dissolved DMSP tracer	
tr_bgc_DMS_sk	true/false	DMS tracer	
phi_snow	real	snow porosity for brine height tracer	
<i>forcing_nml</i>			
		<i>Forcing</i>	
formdrag	true/false	calculate form drag	
atmbndy	default	stability-based boundary layer	‘default’
	constant	bulk transfer coefficients	
fyear_init	yyyy	first year of atmospheric forcing data	
ycycle	integer	number of years in forcing data cycle	
atm_data_format	nc	read atmo forcing files	
	bin	read direct access, binary files	
atm_data_type	default	constant values defined in the code	
	LYq	AOMIP/Large-Yeager forcing data	
	monthly	monthly forcing data	
	ncar	NCAR bulk forcing data	
	oned	column 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	
highfreq	true/false	high-frequency atmo coupling	
natmiter	integer	number of atmo boundary layer iterations	
calc_Tsfc	true/false	calculate surface temperature	.true.
precip_units	mks	liquid precipitation data units	
	mm_per_month		
	mm_per_sec	(same as MKS units)	
tfrz_option	minuslp8	constant ocean freezing temperature (-1.8°C)	
	linear_salt	linear function of salinity (ktherm=1)	
	mushy_layer	matches mushy-layer thermo (ktherm=2)	
ustar_min	real	minimum value of ocean friction velocity	0.0005 m/s
fbot_xfer_type	constant	constant ocean heat transfer coefficient	
	Cdn_ocn	variable ocean heat transfer coefficient	
update_ocn_f	true	include frazil water/salt fluxes in ocn fluxes	
	false	do not include (when coupling with POP)	
l_mpond_fresh	true	retain (topo) pond water until ponds drain	
	false	release (topo) pond water immediately to ocean	
oceanmixed_ice	true/false	active ocean mixed layer calculation	.true. (if uncoupled)
ocn_data_format	nc	read ocean forcing files	
	bin	read direct access, binary files	
sss_data_type	default	constant values defined in the code	
	clim	climatological data	
	near	POP ocean forcing data	
sst_data_type	default	constant values defined in the code	
	clim	climatological data	

Continued on next page

Table 1.8 – continued from previous page

variable	options/format	description	recommended value
	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_tracer_nml			
		<i>History Fields</i>	
f_<var>	string	frequency units for writing <var> 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	
	l	write history every time step	
	x	do not write <var> to history	
	md	e.g., write both monthly and daily files	
f_<var>_ai		grid cell average of <var> ($\times a_i$)	

1.4 Index of primary variables and parameters

This index defines many of the symbols used frequently in the ice model code. Values appearing in this list are fixed or recommended; most namelist parameters are indicated (E_o) with their default values. For other namelist options, see Section [Table 8](#). All quantities in the code are expressed in MKS units (temperatures may take either Celsius or Kelvin units).

1.4.1 Comprehensive Alphabetical Index

Table 1.9: Alphabetical Index

A			
a11,a12	structure tensor components		
a2D	history field accumulations, 2d		
a3Dz	history field accumulations, 3D vertical		
a3Db	history field accumulations, 3D bio grid		
a3Dc	history field accumulations, 3D categories		
a4Di	history field accumulations, 4D categories, vertical ice		
a4Db	history field accumulations, 4D categories, vertical bio grid		
a4Ds	history field accumulations, 4D categories, vertical snow		
a_min	minimum area concentration for computing velocity	0.001	
a_rapid_mode	• brine channel diameter		
advection	• type of advection algorithm used ('remap' or 'up-wind')	remap	

Continued on next page

Table 1.9 – continued from previous page

ahmax	• thickness above which ice albedo is constant	0.3m
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')	
albcnt	counter for averaging albedo	
albice	bare ice albedo	
albice_i	• near infrared ice albedo for thicker ice	
albice_v	• visible ice albedo for thicker ice	
albocn	ocean albedo	0.06
albpnd	melt pond albedo	
albsno	snow albedo	
albsnow_i	• near infrared, cold snow albedo	
albsnow_v	• visible, cold snow albedo	
algalN	algal nitrogen concentration	mmol/m ³
alv(n)dr(f)	albedo: visible (near IR), direct (diffuse)	
alv(n)dr(f)_ai	grid-box-mean value of alv(n)dr(f)	
amm	ammonia/um concentration	mmol/m ³
ANGLE	for conversions between the POP grid and latitude-longitude grids	radians
ANGLET	ANGLE converted to T-cells	radians
apartcn	participation function	
apeff_ai	grid-cell-mean effective pond fraction	
apondn	area concentration of melt ponds	
arlx1i	relaxation constant for dynamics (stress)	
araftn	area fraction of rafted ice	
aredistrn	redistribution function: fraction of new ridge area	
ardgn	fractional area of ridged ice	
aspect_rapid_mode	• brine convection aspect ratio	1
astar	e-folding scale for participation function	0.05
atm_data_dir	• directory for atmospheric forcing data	
atm_data_format	• format of atmospheric forcing files	
atm_data_type	• type of atmospheric forcing	
atmbndy	• atmo boundary layer parameterization ('default' or 'constant')	
avail_hist_fields	type for history field data	
awtidf	weighting factor for near-ir, diffuse albedo	0.36218
awtidr	weighting factor for near-ir, direct albedo	0.00182
awtvdf	weighting factor for visible, diffuse albedo	0.63282
awtvdr	weighting factor for visible, direct albedo	0.00318
B		
bfb_flag	• for bit-for-bit reproducible diagnostics	
bgc_data_dir	• data directory for bgc	
bgc_flux_type	• ice-ocean flux velocity	
bgc_tracer_type	tracer_type for bgc tracers	
bgrid	nondimensional vertical grid points for bio grid	
bignum	a large number	10 ³⁰

Continued on next page

Table 1.9 – continued from previous page

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		
bphi	porosity of ice layers on bio grid		
brlx	relaxation constant for dynamics (momentum)		
bTiz	temperature of ice layers on bio grid		
C			
c<n>	real(<i>n</i>)		
calc_strair	• if true, calculate wind stress	T	
calc_Tsfc	• if true, calculate surface temperature	T	
Cdn_atm	atmospheric drag coefficient		
Cdn_ocn	ocean drag coefficient		
Cf	• ratio of ridging work to PE change in ridging	17.	
cgrid	vertical grid points for ice grid (compare bgrid)		
char_len	length of character variable strings	80	
char_len_long	length of longer character variable strings	256	
check_step	time step on which to begin writing debugging data		
check_umax	if true, check for ice speed > umax_stab		
cldf	cloud fraction		
cm_to_m	cm to meters conversion	0.01	
coldice	value for constant albedo parameterization	0.70	
coldsnow	value for constant albedo parameterization	0.81	
conduct	• conductivity parameterization		
congel	basal ice growth	m	
cosw	cosine of the turning angle in water	1.	
coszen	cosine of the zenith angle		
Cp	proportionality constant for potential energy	kg/m ² /s ²	
cp_air	specific heat of air	1005.0 J/kg/K	
cp_ice	specific heat of fresh ice	2106. J/kg/K	
cp_ocn	specific heat of sea water	4218. J/kg/K	
cp_wv	specific heat of water vapor	1.81x10 ³ J/kg/K	
cp063	diffuse fresnel reflectivity (above)	0.063	
cp455	diffuse fresnel reflectivity (below)	0.455	
Cs	fraction of shear energy contributing to ridging	0.25	
Cstar	constant in Hibler ice strength formula	20.	
cxm	combination of HTN values		

Continued on next page

Table 1.9 – continued from previous page

cyp	combination of HTE values		
cym	combination of HTE values		
cyp	combination of HTE values		
D			
daice_da	data assimilation concentration increment rate		
daiddt	ice area tendency due to dynamics/transport	1/s	
daiddt	ice area tendency due to thermodynamics	1/s	
dalb_mlt	[see ice_shortwave.F90]	-0.075	
dalb_mlti	[see ice_shortwave.F90]	-0.100	
dalb_mltv	[see ice_shortwave.F90]	-0.150	
darcy_V	Darcy velocity used for brine height tracer		
dardg1(n)dt	rate of fractional area loss by ridging ice (category n)	1/s	
dardg2(n)dt	rate of fractional area gain by new ridges (category n)	1/s	
daymo	number of days in one month		
daycal	day number at end of month		
days_per_year	• number of days in one year	365	
dbl_kind	definition of double precision	selected_real_kind(13)	
debug	• write extra diagnostics	.false.	
Delta	function of strain rates (see Section <i>Dynamics</i>)	1/s	
denom1	combination of constants for stress equation		
depressT	ratio of freezing temperature to salinity of brine	0.054 deg/ppt	
dhbr_bt	change in brine height at the bottom of the column		
dhbr_top	change in brine height at the top of the column		
dhsn	depth difference for snow on sea ice and pond ice		
diag_file	• diagnostic output file (alternative to standard out)		
diag_type	• where diagnostic output is written	stdout	
diagfreq	• how often diagnostic output is written (10 = once per 10 dt)		
distrb	distribution data type		
distrb_info	block distribution information		
distribution_type	• method used to distribute blocks on processors		
distribution_weight	• weighting method used to compute work per block		
divu	strain rate I component, velocity divergence	1/s	
divu_adv	divergence associated with advection	1/s	
dms	dimethyl sulfide concentration	mmol/m ³	
dmsp	dimethyl sulfoniopropionate concentration	mmol/m ³	
dpscale	• time scale for flushing in permeable ice	1×10^{-3}	
dragio	drag coefficient for water on ice	0.00536	
dSdt_slow_mode	• drainage strength parameter		
dsnow	change in snow thickness	m	
dt	• thermodynamics time step	3600. s	
dt_dyn	dynamics/ridging/transport time step		
dT_mlt	• Δ temperature per Δ snow grain radius	1. deg	
dte	subcycling time step for EVP dynamics (Δt_e)	s	
dte2T	dte / 2(damping time scale)		

Continued on next page

Table 1.9 – continued from previous page

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	
dxhy	combination of HTE values	
dxt	width of T cell (Δx) through the middle	m
dxu	width of U cell (Δx) through the middle	m
dyhx	combination of HTN values	
dyn_dt	dynamics and transport time step (Δt_{dyn})	s
dyt	height of T cell (Δy) through the middle	m
dyu	height of U cell (Δy) through the middle	m
dvidtd	ice volume tendency due to dynamics/transport	m/s
dvidtt	ice volume tendency due to thermodynamics	m/s
dvirg(n)dt	ice volume ridging rate (category n)	m/s
E		
e11, e12, e22	strain rate tensor components	
ecci	yield curve minor/major axis ratio, squared	1/4
eice(n)	energy of melting of ice per unit area (in category n)	J/m ²
emissivity	emissivity of snow and ice	0.95
eps13	a small number	10 ⁻¹³
eps16	a small number	10 ⁻¹⁶
esno(n)	energy of melting of snow per unit area (in category n)	J/m ²
evap	evaporative water flux	kg/m ² /s
ew_boundary_type	• type of east-west boundary condition	
eyc	coefficient for calculating the parameter E, $0 < eyc < 1$	0.36
F		
faero_atm	aerosol deposition rate	kg/m ² /s
faero_ocn	aerosol flux to the ocean	kg/m ² /s
fbot_xfer_type	• type of heat transfer coefficient under ice	
fcondtop(n)(_f)	conductive heat flux	W/m ²
fcor_blk	Coriolis parameter	1/s
ferrmax	max allowed energy flux error (thermodynamics)	1 x 10 ⁻³ W/m ²
ffracn	fraction of fsurf used to melt pond ice	
fhocn	net heat flux to ocean	W/m ²
fhocn_ai	grid-box-mean net heat flux to ocean (fhocn)	W/m ²
field_loc_center	field centered on grid cell	1
field_loc_Eface	field centered on east face	4
field_loc_NEcorner	field on northeast corner	2
field_loc_Nface	field centered on north face	3
field_loc_noupdate	ignore location of field	-1
field_loc_unknown	unknown location of field	0
field_loc_Wface	field centered on west face	5
field_type_angle	angle field type	3
field_type_noupdate	ignore field type	-1
field_type_scalar	scalar field type	1
field_type_unknown	unknown field type	0
field_type_vector	vector field type	2
first_ice	flag for initial ice formation	

Continued on next page

Table 1.9 – continued from previous page

flat	latent heat flux	W/m ²	
floediam	effective floe diameter for lateral melt	300. m	
floeshape	floe shape constant for lateral melt	0.66	
flux_bio	all biogeochemistry fluxes passed to ocean		
flux_bio_ai	all biogeochemistry fluxes passed to ocean, grid cell mean		
flw	incoming longwave radiation	W/m ²	
flwout	outgoing longwave radiation	W/m ²	
fm	Coriolis parameter * mass in U cell	kg/s	
formdrag	• calculate form drag		
fpond	fresh water flux to ponds	kg/m ² /s	
fr_resp	bgc respiration fraction	0.05	
frain	rainfall rate	kg/m ² /s	
frazil	frazil ice growth	m	
fresh	fresh water flux to ocean	kg/m ² /s	
fresh_ai	grid-box-mean fresh water flux (fresh)	kg/m ² /s	
frz_onset	day of year that freezing begins		
frzmlt	freezing/melting potential	W/m ²	
frzmlt_init	freezing/melting potential at beginning of time step	W/m ²	
frzmlt_max	maximum magnitude of freezing/melting potential	1000. W/m ²	
frzpond	• Stefan refreezing of melt ponds	‘hlid’	
fsalt	net salt flux to ocean	kg/m ² /s	
fsalt_ai	grid-box-mean salt flux to ocean (fsalt)	kg/m ² /s	
fsens	sensible heat flux	W/m ²	
fsnow	snowfall rate	kg/m ² /s	
fsnowrdg	snow fraction that survives in ridging	0.5	
fsurf(n)(_f)	net surface heat flux excluding fcondtop	W/m ²	
fsw	incoming shortwave radiation	W/m ²	
fswabs	total absorbed shortwave radiation	W/m ²	
fswfac	scaling factor to adjust ice quantities for updated data		
fswint	shortwave absorbed in ice interior	W/m ²	
fswpenl	shortwave penetrating through ice layers	W/m ²	
fswthru	shortwave penetrating to ocean	W/m ²	
fswthru_ai	grid-box-mean shortwave penetrating to ocean (fswthru)	W/m ²	
fyear	current data year		
fyear_final	last data year		
fyear_init	• initial data year		
G			
gravit	gravitational acceleration	9.80616 m/s ²	
grid_file	• input file for grid info		
grid_format	• format of grid files		
grid_type	• ‘rectangular’, ‘displaced_pole’, ‘column’ or ‘regional’		
gridcpl_file	• input file for coupling grid info		

Continued on next page

Table 1.9 – continued from previous page

grow_net	specific biogeochemistry growth rate per grid cell	s^{-1}	
Gstar	piecewise-linear ridging participation function parameter	0.15	
H			
halo_info	information for updating ghost cells		
heat_capacity	• if true, use salinity-dependent thermodynamics	T	
hfrazilmin	minimum thickness of new frazil ice	0.05 m	
hi_min	minimum ice thickness for thinnest ice category	0.01 m	
hi_ssl	ice surface scattering layer thickness	0.05 m	
hicen	ice thickness in category n	m	
highfreq	• high-frequency atmo coupling	F	
hin_old	ice thickness prior to growth/melt	m	
hin_max	category thickness limits	m	
hist_avg	• if true, write averaged data instead of snapshots	T	
histfreq	• units of history output frequency: y, m, w, d or l		
histfreq_n	• integer output frequency in histfreq units		
history_dir	• path to history output files		
history_file	• history output file prefix		
hm	land/boundary mask, thickness (T-cell)		
hmix	ocean mixed layer depth	20. m	
hour	hour of the year		
hp0	pond depth at which shortwave transition to bare ice occurs	0.2 m	
hp1	• critical ice lid thickness for topo ponds (dEdd)	0.01 m	
hpmmin	minimum melt pond depth (shortwave)	0.005 m	
hpondn	melt pond depth	m	
hs_min	minimum thickness for which T_s is computed	$1. \times 10^{-4}$ 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	
hs_ssl	snow surface scattering layer thickness	0.04 m	
Hstar	determines mean thickness of ridged ice	25. m	
HTE	length of eastern edge (Δy) of T-cell	m	
HTN	length of northern edge (Δx) of T-cell	m	
HTS	length of southern edge (Δx) of T-cell	m	
HTW	length of western edge (Δy) T-cell	m	
I			
i(j)_glob	global domain location for each grid cell		
i0vis	fraction of penetrating visible solar radiation	0.70	
iblkp	block on which to write debugging data		
i(j)block	Cartesian i,j position of block		
ice_hist_field	type for history variables		
ice_ic	• choice of initial conditions (see Table 4)		
ice_stdout	unit number for standard output		
ice_stderr	unit number for standard error output		

Continued on next page

Table 1.9 – continued from previous page

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		
igrid	interface points for vertical bio grid		
i(j)hi	last i(j) index of physical domain (local)		
i(j)lo	first i(j) index of physical domain (local)		
incond_dir	• directory to write snapshot of initial condition		
incond_file	• prefix for initial condition file name		
int_kind	definition of an integer	selected_real_kind(6)	
integral_order	polynomial order of quadrature integrals in remapping	3	
ip, jp	local processor coordinates on which to write debugging data		
istep	local step counter for time loop		
istep0	• number of steps taken in previous run	0	
istep1	total number of steps at current time step		
Iswabs	shortwave radiation absorbed in ice layers	W/m ²	
J			
K			
kalg	• absorption coefficient for algae		
kappav	visible extinction coefficient in ice, wavelength < 700nm	1.4 m^{-1}	
kcatbound	• category boundary formula		
kdyn	• type of dynamics (1 = EVP, 0 = off)	1	
kg_to_g	kg to g conversion factor	1000.	
kice	thermal conductivity of fresh ice ([7])	2.03 W/m/deg	
kitd	• type of itd conversions (0 = delta function, 1 = linear remap)	1	
kmt_file	• input file for land mask info		
krdg_partic	• ridging participation function	1	
krdg_redist	• ridging redistribution function	1	
krgdn	mean ridge thickness per thickness of ridging ice		
kseaice	thermal conductivity of ice for zero-layer thermodynamics	2.0 W/m/deg	
ksno	thermal conductivity of snow	0.30 W/m/deg	
kstrength	• ice strength formulation (1 = [57], 0 = [26])	1	
ktherm	• thermodynamic formulation (0 = zero-layer, 1 = [7], 2 = mushy)		
L			
l_brine	flag for brine pocket effects		
l_conservation_check	if true, check conservation when ridging		

Continued on next page

Table 1.9 – continued from previous page

<code>l_fixed_area</code>	flag for prescribing remapping fluxes	
<code>l_mpond_fresh</code>	• if true, retain (topo) pond water until ponds drain	
<code>latpnt</code>	• desired latitude of diagnostic points	degrees N
<code>latt(u)_bounds</code>	latitude of T(U) grid cell corners	degrees N
<code>lcdf64</code>	• if true, use 64-bit format	
<code>Lfresh</code>	latent heat of melting of fresh ice = $L_{\text{sub}} - L_{\text{vap}}$	J/kg
<code>lhcoef</code>	transfer coefficient for latent heat	
<code>lmask_n(s)</code>	northern (southern) hemisphere mask	
<code>local_id</code>	local address of block in current distribution	
<code>log_kind</code>	definition of a logical variable	kind(.true.)
<code>lonpnt</code>	• desired longitude of diagnostic points	degrees E
<code>lont(u)_bounds</code>	longitude of T(U) grid cell corners	degrees E
<code>Lsub</code>	latent heat of sublimation for fresh water	2.835×10^6 J/kg
<code>ltripole_grid</code>	flag to signal use of tripole grid	
<code>Lvap</code>	latent heat of vaporization for fresh water	2.501×10^6 J/kg
M		
<code>m_min</code>	minimum mass for computing velocity	0.01 kg/m ²
<code>m_to_cm</code>	meters to cm conversion	100.
<code>m1</code>	constant for lateral melt rate	1.6×10^{-6} m/s deg ^{-m2}
<code>m2</code>	constant for lateral melt rate	1.36
<code>m2_to_km2</code>	m ² to km ² conversion	1×10^{-6}
<code>maskhalo_bound</code>	• turns on <i>bound_state</i> halo masking	
<code>maskhalo_dyn</code>	• turns on dynamics halo masking	
<code>maskhalo_remap</code>	• turns on transport halo masking	
<code>master_task</code>	task ID for the controlling processor	
<code>max_blocks</code>	maximum number of blocks per processor	
<code>max_ntrcr</code>	maximum number of tracers available	5
<code>maxraft</code>	maximum thickness of ice that rafts	1. m
<code>mday</code>	day of the month	
<code>meltb</code>	basal ice melt	m
<code>meltil</code>	lateral ice melt	m
<code>melts</code>	snow melt	m
<code>meltt</code>	top ice melt	m
<code>min_salin</code>	threshold for brine pockets	0.1 ppt
<code>mlt_onset</code>	day of year that surface melt begins	
<code>month</code>	the month number	
<code>monthp</code>	previous month number	
<code>mps_to_cmpdy</code>	m per s to cm per day conversion	8.64×10^6
<code>mtask</code>	local processor number that writes debugging data	
<code>mu_rdg</code>	• e-folding scale of ridged ice	
<code>my_task</code>	task ID for the current processor	
N		
<code>n_aero</code>	number of aerosol species	
<code>natmiter</code>	• number of atmo boundary layer iterations	5
<code>nblocks</code>	number of blocks on current processor	

Continued on next page

Table 1.9 – continued from previous page

nblocks_tot	total number of blocks in decomposition		
nblocks_x(y)	total number of blocks in x(y) direction		
nbtrcr	number of biology tracers		
ncat	number of ice categories	5	
ncat_hist	number of categories written to history		
ndte	• number of subcycles	120	
ndtd	• number of dynamics/advection steps under thermo	1	
new_day	flag for beginning new day		
new_hour	flag for beginning new hour		
new_month	flag for beginning new month		
new_year	flag for beginning new year		
nghost	number of rows of ghost cells surrounding each subdomain	1	
ngroups	number of groups of flux triangles in remapping	5	
nhlat	northern latitude of artificial mask edge	30°S	
nilyr	number of ice layers in each category	4	
nit	nitrate concentration	mmol/m ³	
nit_data_type	• forcing type for nitrate		
nlt_bgc_[chem]	ocean sources and sinks for biogeochemistry		
nml_filename	namelist file name		
nprocs	• total number of processors		
npt	• total number of time steps (dt)		
ns_boundary_type	• type of north-south boundary condition		
nslyr	number of snow layers in each category		
nspint	number of solar spectral intervals		
nstreams	number of history output streams (frequencies)		
nt_<trcr>	tracer index		
ntrace	number of fields being transported		
ntrcr	number of tracers		
nu_diag	unit number for diagnostics output file		
nu_dump	unit number for dump file for restarting		
nu_dump_eap	unit number for EAP dynamics dump file for restarting		
nu_dump_[tracer]	unit number for tracer dump file for restarting		
nu_forcing	unit number for forcing data file		
nu_grid	unit number for grid file		
nu_hdr	unit number for binary history header file		
nu_history	unit number for history file		
nu_kmt	unit number for land mask file		
nu_nml	unit number for namelist input file		
nu_restart	unit number for restart input file		
nu_restart_eap	unit number for EAP dynamics restart input file		
nu_restart_[tracer]	unit number for tracer restart input file		
nu_rst_pointer	unit number for pointer to latest restart file		
num_avail_hist_fields_[shape]	number of history fields of each array shape		
nvar	number of horizontal grid fields written to history		
nvarz	number of category, vertical grid fields written to history		
nx(y)_block	total number of gridpoints on block in x(y) direction		

Continued on next page

Table 1.9 – continued from previous page

nx(y)_global	number of physical gridpoints in x(y) direction, global domain		
nyr	year number		
O			
ocean_bio	concentrations of bgc constituents in the ocean		
oceanmixed_file	• data file containing ocean forcing data		
oceanmixed_ice	• if true, use internal ocean mixed layer		
ocn_data_dir	• directory for ocean forcing data		
ocn_data_format	• format of ocean forcing files		
omega	angular velocity of Earth	7.292×10^{-5} rad/s	
opening	rate of ice opening due to divergence and shear	1/s	
P			
p001	1/1000		
p01	1/100		
p025	1/40		
p027	1/36		
p05	1/20		
p055	1/18		
p1	1/10		
p111	1/9		
p15	15/100		
p166	1/6		
p2	1/5		
p222	2/9		
p25	1/4		
p333	1/3		
p4	2/5		
p5	1/2		
p52083	25/48		
p5625m	-9/16		
p6	3/5		
p666	2/3		
p75	3/4		
phi_c_slow_mode	• critical liquid fraction		
phi_i_mushy	• solid fraction at lower boundary		
phi_sk	skeletal layer porosity		
phi_snow	• snow porosity for brine height tracer		
pi	π		
pi2	2π		
pih	$\pi/2$		
piq	$\pi/4$		
pi(j,b,m)loc	x (y, block, task) location of diagnostic points		
plat	grid latitude of diagnostic points		
plon	grid longitude of diagnostic points		
pndaspect	• aspect ratio of pond changes (depth:area)	0.8	
pointer_file	• input file for restarting		
potT	atmospheric potential temperature	K	
PP_net	total primary productivity per grid cell	mg C/m ² /s	
precip_units	• liquid precipitation data units		

Continued on next page

Table 1.9 – continued from previous page

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	$2.75 \times 10^4 \text{ N/m}$
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	$1.16378 \times 10^7 \text{ kg/m}^3$
qqqocn	for saturated specific humidity over ocean	$6.275724 \times 10^6 \text{ kg/m}^3$
Qref	2m atmospheric reference specific humidity	kg/kg
R		
R_C2N	algal carbon to nitrate factor	7. mole/mole
R_gC2molC	mg/mmol carbon	12.01 mg/mole
R_chl2N	algal chlorophyll to nitrate factor	3. mg/mmol
R_ice	• parameter for Delta-Eddington ice albedo	
R_pnd	• parameter for Delta-Eddington pond albedo	
R_S2N	algal silicate to nitrate factor	0.03 mole/mole
R_snw	• parameter for Delta-Eddington snow albedo	
r16_kind	definition of quad precision	selected_real_kind(26)
Rac_rapid_mode	• critical Rayleigh number	10
rad_to_deg	degree-radian conversion	$180/\pi$
radius	earth radius	$6.37 \times 10^6 \text{ m}$
rdg_conv	convergence for ridging	1/s
rdg_shear	shear for ridging	1/s
real_kind	definition of single precision real	selected_real_kind(6)
refindx	refractive index of sea ice	1.310
revp	real(revised_evp)	
restart	• if true, initialize using restart file instead of defaults	T
restart_age	• if true, read age restart file	
restart_bgc	• if true, read bgc restart file	
restart_dir	• path to restart/dump files	
restart_file	• restart file prefix	
restart_format	• restart file format	
restart_[tracer]	• if true, read tracer restart file	
restart_ext	• if true, read/write halo cells in restart file	
restore_bgc	• if true, restore nitrate/silicate to data	
restore_ice	• if true, restore ice state along lateral boundaries	
restore_sst	• restore sst to data	
revised_evp	• if true, use revised EVP parameters and approach	
rfracmin	• minimum melt water fraction added to ponds	0.15
rfracmax	• maximum melt water fraction added to ponds	1.0
rhoa	air density	kg/m^3
rhofresh	density of fresh water	1000.0 kg/m^3

Continued on next page

Table 1.9 – continued from previous page

rhoi	density of ice	917. kg/m ³	
rhos	density of snow	330. kg/m ³	
rhosi	average sea ice density (for hbrine tracer)	940. kg/m ³	
rhow	density of seawater	1026. kg/m ³	
rnilyr	real(nlyr)		
rside	fraction of ice that melts laterally		
rsnw_fresh	freshly fallen snow grain radius	100. × 10 ⁻⁶ m	
rsnw_melt	• melting snow grain radius	1000. × 10 ⁻⁶ m	
rsnw_nonmelt	nonmelting snow grain radius	500. × 10 ⁻⁶ m	
rsnw_sig	standard deviation of snow grain radius	250. × 10 ⁻⁶ m	
runid	• identifier for run		
runtype	• type of initialization used		
S			
s11, s12, s22	stress tensor components		
salinz	ice salinity profile	ppt	
saltmax	max salinity, at ice base ([7])	3.2 ppt	
scale_factor	scaling factor for shortwave radiation components		
sec	seconds elapsed into idate		
secday	number of seconds in a day	86400.	
shcoef	transfer coefficient for sensible heat		
shear	strain rate II component	1/s	
shlat	southern latitude of artificial mask edge	30°N	
shortwave	• flag for shortwave parameterization ('default' or 'dEdd')		
sig1(2)	principal stress components (diagnostic)		
sil	silicate concentration	mmol/m ³	
sil_data_type	• forcing type for silicate		
sinw	sine of the turning angle in water	0.	
sk_l	skeletal layer thickness	0.03 m	
snoice	snow-ice formation	m	

Continued on next page

Table 1.9 – continued from previous page

snowpatch	length scale for parameterizing nonuniform snow coverage	0.02 m
skl_bgc	• biogeochemistry on/off	
spval	special value (single precision)	10^{30}
spval_dbl	special value (double precision)	10^{30}
ss_tltx(y)	sea surface in the x(y) direction	m/m
sss	sea surface salinity	ppt
sss_data_type	• source of surface salinity data	
sst	sea surface temperature	C
sst_data_type	• source of surface temperature data	
Sswabs	shortwave radiation absorbed in snow layers	W/m^2
stefan-boltzmann	Stefan-Boltzmann constant	$5.67 \times 10^{-8} \text{ W/m}^2 \text{ K}^4$
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
strltlx(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	$1/\text{m}^2$
tareas	area of southern hemisphere T-cells	m^2
tcstr	string identifying T grid for history variables	
tday	absolute day number	
Tf	freezing temperature	C
Tffresh	freezing temp of fresh ice	273.15 K
tfrz_option	• form of ocean freezing temperature	
thinS	minimum ice thickness for brine tracer	
time	total elapsed time	s
time_beg	beginning time for history averages	
time_bounds	beginning and ending time for history averages	
time_end	ending time for history averages	
time_forc	time of last forcing update	s
Timelt	melting temperature of ice top surface	0. C
tinyarea	puny * tarea	m^2
TLAT	latitude of cell center	radians

Continued on next page

Table 1.9 – continued from previous page

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
Tmltz	melting temperature profile of ice	
Tocnfrz	temperature of constant freezing point parameterization	-1.8 C
tr_aero	• if true, use aerosol tracers	
tr_bgc_[tracer]	• if true, use biogeochemistry tracers	
tr_brine	• if true, use brine height tracer	
tr_FY	• if true, use first-year area tracer	
tr_iage	• if true, use ice age tracer	
tr_lvl	• if true, use level ice area and volume tracers	
tr_pond_cesm	• if true, use CESM melt pond scheme	
tr_pond_lvl	• if true, use level-ice melt pond scheme	
tr_pond_topo	• if true, use topo melt pond scheme	
trcr	ice tracers	
trcr_depend	tracer dependency on basic state variables	
Tref	2m atmospheric reference temperature	K
trestore	• restoring time scale	days
tripole	if true, block lies along tripole boundary	
tripoleT	if true, tripole boundary is T-fold; if false, U-fold	
Tsf_errmax	max allowed T_{sf} error (thermodynamics)	$5. \times 10^{-4}$ 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
TTToen	for saturated specific humidity over ocean	5107.4 K
U		
uarea	area of U-cell	m ²
uarear	1/uarea	m ⁻²
uatm	wind velocity in the x direction	m/s
ULAT	latitude of U-cell centers	radians
ULON	longitude of U-cell centers	radians
umask	land/boundary mask, velocity (U-cell)	
umax_stab	ice speed threshold (diagnostics)	1. m/s
umin	min wind speed for turbulent fluxes	1. m/s
uocn	ocean current in the x-direction	m/s
update_ocn_f	• if true, include frazil ice fluxes in ocean flux fields	
use_leap_years	• if true, include leap days	
use_restart_time	• if true, use date from restart file	
ustar_min	• minimum friction velocity under ice	
ucstr	string identifying U grid for history variables	
uvel	x-component of ice velocity	m/s

Continued on next page

Table 1.9 – continued from previous page

uvel_init	x-component of ice velocity at beginning of time step	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	
viscosity_dyn	dynamic viscosity of brine	1.79×10^{-3} kg/m/s	
vocn	ocean current in the y-direction	m/s	
vonkar	von Karman constant	0.4	
vraftn	volume of rafted ice	m	
vrdsn	volume of ridged ice	m	
vrdsntrn	redistribution function: fraction of new ridge volume		
vsno(n)	volume per unit area of snow (in category n)	m	
vvel	y-component of ice velocity	m/s	
vvel_init	y-component of ice velocity at beginning of time step	m/s	
W			
warmice	value for constant albedo parameterization	0.68	
warmnsno	value for constant albedo parameterization	0.77	
wind	wind speed	m/s	
write_history	if true, write history now		
write_ic	• if true, write initial conditions		
write_restart	if 1, write restart now		
X			
Y			
ycycle	• number of years in forcing data cycle		
yday	day of the year		
yield_curve	type of yield curve	ellipse	
yieldstress11(12, 22)	yield stress tensor components		
year_init	• the initial year		
Z			
zlvl	atmospheric level height	m	
zref	reference height for stability	10. m	
zTrf	reference height for T_{ref} , Q_{ref} , U_{ref}	2. m	
zvir	gas constant (water vapor)/gas constant (air) - 1	0.606	

1.5 References

1.5.0.0.3 References

1.6 Useful tools

- search

BIBLIOGRAPHY

- [1] T.L. Amundrud, H. Mallin, and R.G. Ingram. Geometrical constraints on the evolution of ridged sea ice. *J. Geophys. Res. Oceans*, 2004. URL: <http://dx.doi.org/10.1029/2003JC002251>.
- [2] K.C. Armour, C.M. Bitz, L. Thompson, and E.C. Hunke. Controls on Arctic sea ice from first-year and multi-year ice survivability. *J. Climate*, 24:2378–2390, 2011. URL: <http://dx.doi.org/10.1175/2010JCLI3823.1>.
- [3] K.R. Arrigo, J.N. Kremer, and C.W. Sullivan. A simulated Antarctic fast ice ecosystem. *J. Geophys. Res. Oceans*, 98:6929–6946, 1993. URL: <http://dx.doi.org/10.1029/93JC00141>.
- [4] S.P.S. Arya. A drag partition theory for determining the large-scale roughness parameter and wind stress on the Arctic pack ice. *J. Geophys. Res.*, 80:3447–3454, 1975. URL: <http://dx.doi.org/10.1029/JC080i024p03447>.
- [5] A. Assur. Composition of sea ice and its tensile strength. In *Arctic sea ice; conference held at Easton, Maryland, February 24–27, 1958*, volume 598, pages 106–138. Publs. Natl. Res. Coun. Wash., Washington, D.C., 1958.
- [6] C.M. Bitz, M.M. Holland, M. Eby, and A.J. Weaver. Simulating the ice-thickness distribution in a coupled climate model. *J. Geophys. Res. Oceans*, 106:2441–2463, 2001. URL: <http://dx.doi.org/10.1029/1999JC000113>.
- [7] C.M. Bitz and W.H. Lipscomb. An energy-conserving thermodynamic sea ice model for climate study. *J. Geophys. Res. Oceans*, 104(C7):15669–15677, 1999. URL: <http://dx.doi.org/10.1029/1999JC900100>.
- [8] S. Bouillon, T. Fichefet, V. Legat, and G. Madec. The elastic-viscous-plastic method revisited. *Ocean Modelling*, 71:1–12, 2013. URL: <http://dx.doi.org/10.1016/j.ocemod.2013.05.013>.
- [9] B.P. Briegleb and B. Light. *A Delta-Eddington multiple scattering parameterization for solar radiation in the sea ice component of the Community Climate System Model*. NCAR Technical Note NCAR/TN-472+STR, National Center for Atmospheric Research, 2007. URL: https://github.com/CICE-Consortium/CICE/blob/master/doc/PDF/BL_NCAR2007.pdf.
- [10] 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. URL: [http://dx.doi.org/10.1175/1520-0485\(2004\)034<1776:OTCSOT>1.0.CO;2](http://dx.doi.org/10.1175/1520-0485(2004)034<1776:OTCSOT>1.0.CO;2).
- [11] C. Deal, M. Jin, S. Elliott, E. Hunke, M. Maltrud, and N. Jeffery. Large scale modeling of primary production and ice algal biomass within Arctic sea ice in 1992. *J. Geophys. Res. Oceans*, 2011. URL: <http://dx.doi.org/10.1029/2010JC006409>.
- [12] J.K. Dukowicz and J.R. Baumgardner. Incremental remapping as a transport/advection algorithm. *J. Comput. Phys.*, 160:318–335, 2000. URL: <http://dx.doi.org/10.1006/jcph.2000.6465>.
- [13] 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(1):195–208, 1993. URL: [https://doi.org/10.1175/1520-0426\(1993\)010<195:ARAIOT>1.0.CO;2](https://doi.org/10.1175/1520-0426(1993)010<195:ARAIOT>1.0.CO;2).
- [14] 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(C4):7991–8014, 1994. URL: <http://dx.doi.org/10.1029/93JC03455>.

- [15] E.E. Ebert, J.L. Schramm, and J.A. Curry. Disposition of solar radiation in sea ice and the upper ocean. *J. Geophys. Res. Oceans*, 100:15965–15975, 1995. URL: <http://dx.doi.org/10.1029/95JC01672>.
- [16] H. Eicken, T.C. Grenfell, D.K. Perovich, J.A. Richter-Menge, and K. Frey. Hydraulic controls of summer Arctic pack ice albedo. *J. Geophys. Res. Oceans*, 2004. URL: <http://dx.doi.org/10.1029/2003JC001989>.
- [17] S. Elliott, C. Deal, G. Humphries, E. Hunke, N. Jeffery, M. Jin, M. Levasseur, and J. Stefels. Pan-Arctic simulation of coupled nutrient-sulfur cycling due to sea ice biology: Preliminary results. *J. Geophys. Res. Biogeo.*, 2011. URL: <http://dx.doi.org/10.1029/2011JG001649>.
- [18] D.L. Feltham, N. Untersteiner, J.S. Wettlaufer, and M.G. Worster. Sea ice is a mushy layer. *Geophys. Res. Lett.*, 2006. URL: <http://dx.doi.org/10.1029/2006GL026290>.
- [19] G.M. Flato and W.D. Hibler. Ridging and strength in modeling the thickness distribution of Arctic sea ice. *J. Geophys. Res. Oceans*, 100:18611–18626, 1995. URL: <http://dx.doi.org/10.1029/95JC02091>.
- [20] D. Flocco and D.L. Feltham. A continuum model of melt pond evolution on Arctic sea ice. *J. Geophys. Res. Oceans*, 2007. URL: <http://dx.doi.org/10.1029/2006JC003836>.
- [21] D. Flocco, D.L. Feltham, and A.K. Turner. Incorporation of a physically based melt pond scheme into the sea ice component of a climate model. *J. Geophys. Res. Oceans*, 2010. URL: <http://dx.doi.org/10.1029/2009JC005568>.
- [22] D. Flocco, D. Schroeder, D.L. Feltham, and E.C. Hunke. Impact of melt ponds on Arctic sea ice simulations from 1990 to 2007. *J. Geophys. Res. Oceans*, 2012. URL: <http://dx.doi.org/10.1029/2012JC008195>.
- [23] H.E. Garcia, R.A. Locarnini, T.P. Boyer, and J.I. Antonov. Nutrients (phosphate, nitrate, silicate). In *World Ocean Atlas 2005*, volume 4. NOAA Atlas NESDIS 64, NOAA, 2006.
- [24] 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. URL: <http://dx.doi.org/10.1029/98JC01258>.
- [25] K.M. Golden, H. Eicken, A.L. Heaton, J. Miner, D.J. Pringle, and J. Zhu. Thermal evolution of permeability and microstructure in sea ice. *Geophys. Res. Lett.*, 2007. URL: <http://dx.doi.org/10.1029/2007GL030447>.
- [26] W.D. Hibler. A dynamic thermodynamic sea ice model. *J. Phys. Oceanogr.*, 9:817–846, 1979. URL: [http://dx.doi.org/10.1175/1520-0485\(1979\)009<1081:ADTSIM>2.0.CO;2](http://dx.doi.org/10.1175/1520-0485(1979)009<1081:ADTSIM>2.0.CO;2).
- [27] W.D. Hibler. Modeling a variable thickness sea ice cover. *Mon. Wea. Rev.*, 108:1943–1973, 1980. URL: [http://dx.doi.org/10.1175/1520-0493\(1980\)108<1943:MAVTSI>2.0.CO;2](http://dx.doi.org/10.1175/1520-0493(1980)108<1943:MAVTSI>2.0.CO;2).
- [28] W.D. Hibler and K. Bryan. A diagnostic ice-ocean model. *J. Phys. Oceanogr.*, 17:987–1015, 1987. URL: [http://dx.doi.org/10.1175/1520-0485\(1987\)017<1087:ADIM>2.0.CO;2](http://dx.doi.org/10.1175/1520-0485(1987)017<1087:ADIM>2.0.CO;2).
- [29] M.M. Holland, D.A. Bailey, B.P. Briegleb, B. Light, and E. Hunke. Improved sea ice shortwave radiation physics in CCSM4: The impact of melt ponds and aerosols on Arctic sea ice. *J. Climate*, 25:1413–1430, 2012. URL: <http://dx.doi.org/10.1175/JCLI-D-11-00078.1>.
- [30] E.C. Hunke. Viscous-plastic sea ice dynamics with the EVP model: Linearization issues. *J. Comp. Phys.*, 170:18–38, 2001. URL: <http://dx.doi.org/10.1006/jcph.2001.6710>.
- [31] E.C. Hunke and C.M. Bitz. Age characteristics in a multidecadal Arctic sea ice simulation. *J. Geophys. Res. Oceans*, 2009. URL: <http://dx.doi.org/10.1029/2008JC005186>.
- [32] E.C. Hunke and J.K. Dukowicz. An elastic-viscous-plastic model for sea ice dynamics. *J. Phys. Oceanogr.*, 27:1849–1867, 1997. URL: [http://dx.doi.org/10.1175/1520-0485\(1997\)027<1849:AEVPMF>2.0.CO;2](http://dx.doi.org/10.1175/1520-0485(1997)027<1849:AEVPMF>2.0.CO;2).
- [33] 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. URL: [http://dx.doi.org/10.1175/1520-0493\(2002\)130<1848:TEVPSI>2.0.CO;2](http://dx.doi.org/10.1175/1520-0493(2002)130<1848:TEVPSI>2.0.CO;2).

- [34] 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. URL: <https://github.com/CICE-Consortium/CICE/blob/master/doc/PDF/LAUR-03-2219.pdf>.
- [35] E.C. Hunke, D.A. Hebert, and O. Lecomte. Level-ice melt ponds in the Los Alamos Sea Ice Model, CICE. *Ocean Modelling*, 71:26–42, 2013. URL: <http://dx.doi.org/10.1016/j.ocemod.2012.11.008>.
- [36] E.C. Hunke and Y. Zhang. A comparison of sea ice dynamics models at high resolution. *Mon. Wea. Rev.*, 127:396–408, 1999. URL: [http://dx.doi.org/10.1175/1520-0493\(1999\)127<T1>textless{}0396:ACOSID\T1\textgreater{}2.0.CO;2](http://dx.doi.org/10.1175/1520-0493(1999)127<T1>textless{}0396:ACOSID\T1\textgreater{}2.0.CO;2).
- [37] M. Jin, C. Deal, J. Wang, K.H. Shin, N. Tanaka, T.E. Whiteledge, S.H. Lee, and R.R. Grading. Controls of the landfast ice-ocean ecosystem offshore Barrow, Alaska. *Ann. Glaciol.*, 44:63–72, 2006. URL: <https://github.com/CICE-Consortium/CICE/blob/master/doc/PDF/JDWSTWLG06.pdf>.
- [38] R.E. Jordan, E.L. Andreas, and A.P. Makshtas. Heat budget of snow-covered sea ice at North Pole 4. *J. Geophys. Res. Oceans*, 104(C4):7785–7806, 1999. URL: <http://dx.doi.org/10.1029/1999JC900011>.
- [39] B.G. Kauffman and W.G. Large. The CCSM coupler, version 5.0.1. *NCAR Tech. Note*, 2002. URL: https://github.com/CICE-Consortium/CICE/blob/master/doc/PDF/KL_NCAR2002.pdf.
- [40] D. Lavoie, K. Denman, and C. Michel. Modeling ice algal growth and decline in a seasonally ice- covered region of the Arctic (Resolute Passage, Canadian Archipelago). *J. Geophys. Res. Oceans*, 2005. URL: <http://dx.doi.org/10.1029/2005JC002922>.
- [41] W.H. Lipscomb. *Modeling the Thickness Distribution of Arctic Sea Ice*. Dept. of Atmospheric Sciences University of Washington, Seattle, 1998. PhD thesis. URL: <http://hdl.handle.net/1773/10081>.
- [42] W.H. Lipscomb. Remapping the thickness distribution in sea ice models. *J. Geophys. Res. Oceans*, 106:13989–14000, 2001. URL: <http://dx.doi.org/10.1029/2000JC000518>.
- [43] W.H. Lipscomb and E.C. Hunke. Modeling sea ice transport using incremental remapping. *Mon. Wea. Rev.*, 132:1341–1354, 2004. URL: [http://dx.doi.org/10.1175/1520-0493\(2004\)132<T1>textless{}1341:MSITUI\T1\textgreater{}2.0.CO;2](http://dx.doi.org/10.1175/1520-0493(2004)132<T1>textless{}1341:MSITUI\T1\textgreater{}2.0.CO;2).
- [44] W.H. Lipscomb, E.C. Hunke, W. Maslowski, and J. Jakacki. Ridging, strength, and stability in high-resolution sea ice models. *J. Geophys. Res. Oceans*, 2007. URL: <http://dx.doi.org/10.1029/2005JC003355>.
- [45] P. Lu, Z. Li, B. Cheng, and M. Lepparanta. A parametrization fo the ice-ocean drag coefficient. *J. Geophys. Res. Oceans*, 2011. URL: <http://dx.doi.org/10.1029/2010JC006878>.
- [46] C. Lüpkes, V.M. Gryanik, J. Hartmann, and E.L. Andreas. A parametrization, based on sea ice morphology, of the neutral atmospheric drag coefficients for weather prediction and climate models. *J. Geophys. Res. Atmos.*, 2012. URL: <http://dx.doi.org/10.1029/2012JD017630>.
- [47] G.A. Maykut. Large-scale heat exchange and ice production in the central Arctic. *J. Geophys. Res. Oceans*, 87:7971–7984, 1982. URL: <http://dx.doi.org/10.1029/JC087iC10p07971>.
- [48] G.A. Maykut and M.G. McPhee. Solar heating of the Arctic mixed layer. *J. Geophys. Res. Oceans*, 100:24691–24703, 1995. URL: <http://dx.doi.org/10.1029/95JC02554>.
- [49] G.A. Maykut and D.K. Perovich. The role of shortwave radiation in the summer decay of a sea ice cover. *J. Geophys. Res. Oceans*, 92:7032–7044, 1987. URL: <http://dx.doi.org/10.1029/JC092iC07p07032>.
- [50] G.A. Maykut and N. Untersteiner. Some results from a time dependent thermodynamic model of sea ice. *J. Geophys. Res.*, 76:1550–1575, 1971. URL: <http://dx.doi.org/10.1029/JC076i006p01550>.
- [51] R.J. Murray. Explicit generation of orthogonal grids for ocean models. *J. Comput. Phys.*, 126:251–273, 1996. URL: <http://dx.doi.org/10.1006/jcph.1996.0136>.
- [52] D. Notz. *Thermodynamic and Fluid-Dynamical Processes in Sea Ice*. University of Cambridge, UK, 2005. PhD thesis. URL: <http://ulmss-newton.lib.cam.ac.uk/vwebv/holdingsInfo?bibId=27224>.

- [53] N. Ono. Specific heat and heat of fusion of sea ice. In H. Oura, editor, *Physics of Snow and Ice*, volume I, pages 599–610. Institute of Low Temperature Science, Hokkaido, Japan, 1967.
- [54] C.L. Parkinson and W.M. Washington. A large-scale numerical model of sea ice. *J. Geophys. Res. Oceans*, 84(C1):331–337, 1979. URL: <http://dx.doi.org/10.1029/JC084iC01p00311>.
- [55] D.J. Pringle, H. Eicken, H.J. Trodahl, and L.G.E. Backstrom. Thermal conductivity of landfast Antarctic and Arctic sea ice. *J. Geophys. Res. Oceans*, 2007. URL: <http://dx.doi.org/10.1029/2006JC003641>.
- [56] A. Rosati and K. Miyakoda. A general circulation model for upper ocean simulation. *J. Phys. Oceanogr.*, 18:1601–1626, 1988. URL: [https://doi.org/10.1175/1520-0485\(1988\)018<1601:AGCMFUVT1>textless{}1601:AGCMFUVT1>textgreater{}2.0.CO;2](https://doi.org/10.1175/1520-0485(1988)018<1601:AGCMFUVT1>textless{}1601:AGCMFUVT1>textgreater{}2.0.CO;2).
- [57] D.A. Rothrock. The energetics of plastic deformation of pack ice by ridging. *J. Geophys. Res.*, 80:4514–4519, 1975. URL: <http://dx.doi.org/10.1029/JC080i033p04514>.
- [58] E.M. Schulson. Brittle failure of ice. *Eng. Fract. Mech.*, 68:1839–1887, 2001. URL: [http://dx.doi.org/10.1016/S0013-7944\(01\)00037-6](http://dx.doi.org/10.1016/S0013-7944(01)00037-6).
- [59] W. Schwarzacher. Pack ice studies in the Arctic Ocean. *J. Geophys. Res.*, 64:2357–2367, 1959. URL: <http://dx.doi.org/10.1029/JZ064i012p02357>.
- [60] A.J. Semtner. A Model for the Thermodynamic Growth of Sea Ice in Numerical Investigations of Climate. *J. Phys. Oceanogr.*, 6:379–389, 1976. URL: [http://dx.doi.org/10.1175/1520-0485\(1976\)006<379:AMFTTGVT1>textless{}0379:AMFTTGVT1>textgreater{}2.0.CO;2](http://dx.doi.org/10.1175/1520-0485(1976)006<379:AMFTTGVT1>textless{}0379:AMFTTGVT1>textgreater{}2.0.CO;2).
- [61] G. Siedler and H. Peters. Physical properties (general) of sea water. In *Landolt-Börnstein: Numerical data and functional relationships in science and technology, New Series V/3a*, pages 233–264. Springer, 1986.
- [62] R.D. Smith, J.K. Dukowicz, and R.C. Malone. Parallel ocean general circulation modeling. *Physica D*, 60(1):38–61, 1992. URL: [https://doi.org/10.1016/0167-2789\(92\)90225-C](https://doi.org/10.1016/0167-2789(92)90225-C).
- [63] 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. URL: <https://github.com/CICE-Consortium/CICE/blob/master/doc/PDF/LAUR-95-1146.pdf>.
- [64] M. Steele. Sea ice melting and floe geometry in a simple ice-ocean model. *J. Geophys. Res. Oceans*, 97:17729–17738, 1992. URL: <http://dx.doi.org/10.1029/92JC01755>.
- [65] 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(C9):21061–21079, 1997. URL: <http://dx.doi.org/10.1029/97JC01454>.
- [66] J. Stefels, M. Steinke, S. Turner, G. Malin, and S. Belviso. Environmental constraints on the production and removal of the climatically active gas dimethylsulphide (DMS) and implications for ecosystem modelling. *Bio-geochemistry*, 83:245–275, 2007. URL: http://dx.doi.org/10.1007/978-1-4020-6214-8_18.
- [67] A.H. Stroud. *Approximate Calculation of Multiple Integrals*. Prentice-Hall, 1971. Englewood Cliffs, New Jersey.
- [68] P.D. Taylor and D.L. Feltham. A model of melt pond evolution on sea ice. *J. Geophys. Res. Oceans*, 2004. URL: <http://dx.doi.org/10.1029/2004JC002361>.
- [69] A.S. Thorndike, D.A. Rothrock, G.A. Maykut, and R. Colony. The thickness distribution of sea ice. *J. Geophys. Res.*, 80:4501–4513, 1975. URL: <http://dx.doi.org/10.1029/JC080i033p04501>.
- [70] H.J. Trodahl, S.O.F. Wilkinson, M.J. McGuinness, and T.G. Haskeel. Thermal conductivity of sea ice: dependence on temperature and depth. *Geophys. Res. Lett.*, 28:1279–1282, 2001. URL: <http://dx.doi.org/10.1029/2000GL012088>.
- [71] M. Tsamados, D.L. Feltham, D. Schroeder, D. Flocco, S.L. Farrell, N.T. Kurtz, S.W. Laxon, and S. Bacon. Impact of variable atmospheric and oceanic form drag on simulations of Arctic sea ice. *J. Phys. Oceanogr.*, 44:1329–1353, 1999. URL: <https://doi.org/10.1175/JPO-D-13-0215.1>.

- [72] 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. URL: <http://dx.doi.org/10.1029/2012JC007990>.
- [73] A.K. Turner, E.C. Hunke, and C.M. Bitz. Two modes of sea-ice gravity drainage: a parameterization for large-scale modeling. *J. Geophys. Res. Oceans*, 118:2279–2294, 2013. URL: <http://dx.doi.org/10.1002/jgrc.20171>.
- [74] N. Untersteiner. Calculations of temperature regime and heat budget of sea ice in the Central Arctic. *J. Geophys. Res.*, 69:4755–4766, 1964. URL: <http://dx.doi.org/10.1029/JZ069i022p04755>.
- [75] H. von Storch and F.W. Zwiers. *Statistical Analysis in Climate Research*. Cambridge University Press, 1999. Cambridge, UK.
- [76] J. Weiss and E.M. Schulson. Coulombic faulting from the grain scale to the geophysical scale: lessons from ice. *J. of Phys. D: Appl. Phys.*, 42:214017, 2009. URL: <http://stacks.iop.org/0022-3727/42/i=21/a=214017>.
- [77] A.V. Wilchinsky and D.L. Feltham. Dependence of sea ice yield-curve shape on ice thickness. *J. Phys. Oceanogr.*, 34:2852–2856, 2004. URL: <http://dx.doi.org/10.1175/JPO2667.1>.
- [78] A.V. Wilchinsky and D.L. Feltham. Modelling the rheology of sea ice as a collection of diamond-shaped floes. *J. Non-Newtonian Fluid Mech.*, 138:22–32, 2006. URL: <http://dx.doi.org/10.1016/j.jnnfm.2006.05.001>.
- [79] D.S. Wilks. *Statistical methods in the atmospheric sciences*. Academic Press, 2006. 2nd ed.