Naval Research Laboratory

Stennis Space Center, MS 39529-5004

NRL/MR/7320--08-9150

Software Design Description for the Polar Ice Prediction System (PIPS) Version 3.0

PREPARED FOR:

Naval Oceanographic Office
Systems Integration Division

PREPARED BY:

PAMELA G. POSEY
LUCY F. SMEDSTAD
RUTH H. PRELLER
E. JOSEPH METZGER

Ocean Dynamics and Prediction Branch
Oceanography Division

AND

SUZANNE CARROLL

Planning Systems, Inc.
Stennis Space Center, Mississippi

November 5, 2008

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 05-11-2008	**2. REPORT TYPE** Memorandum Report	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Software Design Description for the Polar Ice Prediction System (PIPS) Version 3.0		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER 0602435N
6. AUTHOR(S) Pamela G. Posey, Lucy F. Smedstad, Ruth H. Preller, E. Joseph Metzger, and Suzanne Carroll*		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER 73-6057-08-5
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory, Oceanography Division, Stennis Space Center, MS 39529-5004		**8. PERFORMING ORGANIZATION REPORT NUMBER** NRL/MR/7320--08-9150
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research, One Liberty Center, 875 North Randolph St., Arlington, VA 22203-1995		**10. SPONSOR / MONITOR'S ACRONYM(S)** ONR
		11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

*Planning Systems, Inc., MSAAP Building 9121, Stennis Space Center, MS 39529

14. ABSTRACT

The Polar Ice Prediction System (PIPS) Version 3.0 is a dynamic sea-ice model that forecasts conditions in all sea-ice covered areas in the northern hemisphere (down to 30° north in latitude). It has a horizontal resolution of approximately 9 km. The vertical resolution in the model has been set at 45 levels so that Arctic shelves, continental slopes, and submarine ridges are accurately represented. Currently, the domain includes the Irminger, Labrador, North and Baltic Seas on the Atlantic side and the Bering Sea, Sea of Japan, and the Sea of Okhotsk on the Pacific. The PIPS 3.0 system is based on the Los Alamos ice model and coupled (via file transfer) to the operational, global Navy Coastal Ocean Model (gNCOM). The system forecasts daily ice thickness, concentration, and drift in the Arctic Ocean. This report documents the mathematical formulations, flow charts, and descriptions of the programs and subroutines.

15. SUBJECT TERMS

Ice forecast	Ice edge
Ice drift	Ice model

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	UL	147	Pamela Posey
Unclassified	Unclassified	Unclassified			**19b. TELEPHONE NUMBER** *(include area code)* (228) 688-5596

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

TABLE OF CONTENTS

TABLE OF FIGURES

1.0 SCOPE

1.1 Identification

The software described in this document is identified as the Polar Ice Prediction System (PIPS) Version 3.0. PIPS 3.0 is a dynamic sea-ice model that forecasts conditions in all sea-ice covered areas in the northern hemisphere (down to 30° north in latitude). It has a horizontal resolution of approximately 9 km. The vertical resolution in the model has been set at 45 levels so that Arctic shelves, continental slopes and submarine ridges are accurately represented. This allows for 17 levels in the upper 300 m of the water column and a maximum layer thickness in the deep ocean of 300 m. The array size is 1280x720. Currently the domain includes the Irminger, Labrador, North and Baltic Seas on the Atlantic side and the Bering Sea, Sea of Japan and the Sea of Okhotsk on the Pacific side.

PIPS 3.0 model bathymetry south of 64° N is derived from the ETOP05 database, Navy Research Lab charts and Canadian Hydrographic Service charts. Bathymetry north of 64 N comes from the 2.5 km resolution digital International Bathymetric Chart of the Arctic Ocean (IBCAO).

The PIPS 3.0 system uses the Los Alamos ice model, CICE (version 3.1), containing improved procedures for model thermodynamics, physics parameterizations and energy based ridging. It has the ability to predict multi-category ice thickness. The CICE model is designed to run on massively parallel computers [37,10,11]. The CICE model is presently being coupled (via file transfer) to the operational, global Navy Coastal Ocean Model (NCOM), to predict ice thickness, concentration, and drift in the Arctic Ocean. NCOM is a baroclinic, hydrostatic, Boussinesq, free-surface ocean model that allows its vertical coordinate to consist of sigma coordinates for the upper layers and z-levels below a user-specified depth [5],[34]. NCOM runs operationally at NAVOCEANO at a resolution of 1/8° globally. PIPS 3.0 also forecasts surface ocean current and temperature in the surrounding seas. It is currently being tested for success in coupling with the next generation global HYbrid Coordinate Ocean Model (HYCOM). For more information on HYCOM, visit the website at http://hycom.rsmas.miami.edu/hycom/.

PIPS 3.0 is driven by heat fluxes and surface winds from the Navy Operational Global Atmospheric Prediction System (NOGAPS). Daily updates are accomplished through an objective analysis of ice concentration data from the Special Sensor Microwave/Imager (SSM/I) located on the Defense Meteorological Satellite Program (DMSP) satellite.

There are four primary components that work together to comprise the PIPS 3.0 model:

- a thermodynamic model that calculates snowfall as well as local growth rates of snow and ice due to vertical conductive, radiative and turbulent fluxes;
- an ice dynamics model, which predicts the velocity field of the ice pack based on a model of the material strength of the ice;
- a transport model that depicts advection of the aerial concentration, ice volumes and other state variables;
- a ridging parameterization that transports ice among thickness categories based on

Manuscript approved October 3, 2008.

energetic balances and rates of strain.

1.2 Document Overview

The purpose of this Software Design Description (SDD) is to describe the software design and code of the Polar Ice Prediction System (PIPS) Version 3.0. Because the PIPS 3.0 model is largely based on the CICE model, this document reflects the information found in the Los Alamos Sea Ice (CICE) Software User's Manual [1]. The SDD includes the mathematical formulation and solution procedures for PIPS 3.0 as well as flow charts and descriptions of the programs, modules, and subroutines. This document, along with a User's Manual [2] and a Validation Test Report [3], forms a comprehensive documentation package for the PIPS 3.0 model system.

Generally speaking, subroutine names are given in *italic* and file names are **boldface** in this document. Symbols used in the code are typewritten, while corresponding symbols in this document are in the *math* font which is similar to *italic*.

2.0 REFERENCED DOCUMENTS

2.1 PIPS 3.0 Software Documentation

[1] E. C. Hunke and W.H. Lipscomb, "CICE: the Los Alamos Sea Ice Model Documentation and Software", available at http://climate.lanl.gov/Models/CICE/.

[2] P.G. Posey, L.F. Smedstad, R.H. Preller, E.J. Metzger and S.N. Carroll. "User's Manual for the Polar Ice Prediction System (PIPS) Version 3.0", NRL/MR/7320—08-9154, Ocean Modeling Division, Naval Research Laboratory, Stennis Space Center, MS, 2008.

[3] P.G. Posey, L.F. Smedstad, R.H. Preller, E.J. Metzger and S.N. Carroll. "Validation Test Report for the Polar Ice Prediction System (PIPS) Version 3.0", PSI Technical Report SSC-003-06, Ocean Modeling Division, Naval Research Laboratory, Stennis Space Center, MS, 2008.

2.2 General Technical Documentation

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

[5] C.N. Barron, A.B. Kara, P.J. Martin, R.C. Rhodes, and L.F. Smedstad. Formulation, implementation and examination of vertical coordinate choices in the Global Navy Coastal Ocean Model (NCOM). *Ocean Modelling*, 11:347-375, 2006.

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

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

[8] 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(7), 1776-1780, 2004.

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

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

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

[12] 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.

[13] 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.

[14] 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.

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

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

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

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

[19] 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.

[20] 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.

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

[22] R. E. Jordan, E. L. Andreas, and A. P. Makshtas. Heat budget of snow-covered sea ice at North Pole 4. *J. Geophys. Res.–Oceans*, 104:7785–7806, 1999.

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

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

[25] W. H. Lipscomb. Remapping the thickness distribution in sea ice models. *J. Geophys. Res.–Oceans*, 106:13989–14000, 2001.

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

[27] W.H. Lipscomb, E.C. Hunke, W. Maslowski, and J. Jakacki. Improving ridging schemes for high resolution sea ice models. *J. Geophys. Res.-Oceans*, Vol. 112, 2007.

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

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

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

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

[32] R.J. Murray. Explicit generation of orthogonal grids for ocean models. *J Comput. Phys*., 120:251-273, 1996.

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

[34] R.C. Rhodes, H.E. Hurlburt, A.J. Wallcraft, C.N. Barron, P.J. Martin, E.J. Metzger, J.F. Shriver, D.S. Ko, O.M. Smedstad, S.L. Cross and A.B. Kara. Navy Real-Time Global Modeling Systems. *Oceanography*, 15(1): 29-44, 2002.

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

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

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

[38] 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.

[39] P. K. Smolarkiewicz. A fully multi-dimensional positive definite advection transport algorithm with small implicit diffusion. *J. Comput. Phys.*, 54:325–362, 1984.

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

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

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

[43] 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.

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

3.0 PIPS 3.0 SOFTWARE SUMMARY

PIPS 3.0 is written in fixed-format FORTRAN90 and runs on UNIX host platforms, including SGI Origin 3000 (<**OS> = IRIX64** below), SGI Altix (**Linux**), IBM Power4 (**AIX**) and Cray X1 (**UNICOS**). The code is parallelized through grid decomposition with MPI for message passing between processors, with four processors allocated to each hemisphere. The code has been optimized for vector architectures and tested on Fujitsu VPP 5000, Cray X1, and NEC platforms. At NAVOCEANO, PIPS 3.0 is run using 32 processors on an IBM platform with a ConsumableMemory of 500 mb. With respect to hardware resources, a one-day run of PIPS 3.0 at NAVOCEANO requires 0.85 Processor Hrs.

4.0 PIPS 3.0 SOFTWARE INVENTORY

4.1 PIPS 3.0 Components

A complete description of each PIPS 3.0 module and subsequent subroutines, including a definition, purpose, relationship to other modules, etc. is found in Section 5.3. The PIPS 3.0 software is run through a series of modules, makefiles, input files, and scripts.

4.1.1 PIPS 3.0 Modules

CICE.f, ice_albedo.f, ice_atmo.f, ice_calendar.f, ice_constants.f, ice_coupling.f, ice_diagnostics.f, ice_domain.f, ice_dyn_evp.f, ice_exit.f, ice_fileunits.f, ice_flux.f, ice_flux_in.f, ice_grid.f, ice_history.f, ice_init.f, ice_itd.f, ice_itd_linear.f, ice_kinds_mod.f, ice_mechred.f, ice_model_size.f, ice_mpi_internal.f, ice_ocean.f, ice_read_write.f, ice_scaling.f, ice_state.f, ice_therm_itd.f, ice_therm_vertical.f, ice_timers.f, ice_transport_mpdata.f, ice_transport_remap.f, ice_work.f

4.1.2 PIPS 3.0 Subroutines

abort_ice.f, absorbed_solar.f, add_new_ice.f, add_new_snow.f, aggregate.f, aggregate_area.f, albedos.f, asum_ridging.f, atmo_boundary_layer.f, bound.f, bound_aggregate.f, bound_ijn.f, bound_narr.f, bound_narr_ne.f, bound_state.f, bound_sw.f, calendar.f, check_monotonicity.f, check_state.f, column_conservation_check.f, column_sum.f, columgrid.f, complete_getflux_ocn.f, conductivity.f, conservation_check_vthermo.f, conserved_sums.f, construct_fields.f, departure_points.f, dumpfile.f, end_run.f, evp.f, evp_finish.f, evp_prep.f, exit_coupler.f, file_year.f, fit_line.f, flux_integrals.f, freeboard.f, from_coupler.f, frzmlt_bottom_lateral.f, get_sum.f, getflux.f, global_conservation.f, global_gather.f, global_scatter.f, ice_bcast_char.f, ice_bcast_iscalar.f, ice_bcast_logical.f, ice_bcast_rscalar.f, ice_coupling_setup.f, ice_global_real_minmax.f, ice_open.f, ice_read.f, ice_strength.f, ice_timer_clear(n).f, ice_timer_print(n).f, ice_timer_start(n).f, ice_timer_stop(n).f, ice_write.f, ice_write_hist.f, icecdf.f, init_calendar.f, init_constants.f, init_cpl.f, init_diagnostics.f, init_diags.f, init_evp.f, init_flux.f, init_flux_atm.f, init_flux_ocn.f, init_getflux.f, init_grid.f, init_hist.f, init_itd.f, init_mass_diags.f, init_mechred.f, init_remap.f, init_state.f, init_thermo_vars.f, init_thermo_vertical.f, init_vertical_profile.f, input_data.f, integer function lenstr(label).f, interp_coeff.f, interp_coeff_monthly.f, interpolate_data.f, lateral_melt.f, limited_gradient.f, linear_itd.f, load_tracers.f, local_max_min.f, locate_triangles.f, make_masks.f, makemask.f, merge_fluxes.f, mixed_layer.f, mpdata(narrays,phi).f, NCAR_bulk_dat.f, NCAR_files.f, pipsgrid.f, popgrid.f, prepare_forcing.f, principal_stress.f, print_state.f, read_clim_data.f, read_data.f, real function ice_global_real_maxval.f, real function ice_global_real_minval.f, real function ice_global_real_sum.f, rebin.f, rectgrid.f, reduce_area.f, restartfile.f, ridge_ice.f, ridge_prep.f, ridge_shift.f, runtime_diags.f, scale_fluxes.f, scale_hist_fluxes.f, setup_mpi.f, shift_ice.f, sss_clim.f, sss_sst_restore.f, sst_ic.f, stepu.f, stress.f, surface_fluxes.f, t2ugrid.f, temperature_changes.f, thermo_itd.f, thermo_vertical.f, thickness_changes.f, timers.f, Tlatlon.f, to_coupler.f, to_tgrid.f, to_ugrid.f, transport_mpdata.f, transport_remap.f, triangle_coordinates.f, tridiag_solver.f, u2tgrid.f, unload_tracers.f, update_fields.f, update_state_vthermo.f, zap_small_areas.f

4.1.3 PIPS 3.0 Input File Namelist Parameters

The **ice_in.txt** file is an input file of namelist parameters used in running the PIPS 3.0 model at NAVOCEANO. The variables are described in the order they appear in the **ice_in** and with the default values and directory paths used in execution at NAVOCEANO.

4.1.3.1 Ice Namelist (ice_nml)

Name	Type/Options	Description	Default Values / Directory Location
year_init	yyyy	The initial year, if not using restart	= 0001
istep0	integer	Initial time step number	= 0
dt	seconds	Thermo/transport time step length	= 2800.
ndte	integer	Number of EVP subcycles	= 120
npt	integer	Total number of time steps to take	= 70
diagfreq	integer	Frequency of diagnostic output in dt eg., 10 is once every 10 time steps	= 30
histfreq	y, m, w, d, 1	Write history output once a year, month, week, day, or every time step	= 'h'

Name	Type/Options	Description	Default Values / Directory Location
dumpfreq	y, m, d	Write restart every dumpfreq_n years, months, days	= 'd'
dumpfreq_n	integer	Frequency restart data is written	= 1
hist_avg	true/false	Write time-averaged data if true Write snapshots of data if false	= .false.
restart	true/false	Initialize using restart file	= .true.
print_points	true/false	Print diagnostic data for two grid points	= .true.
print_global	true/false	Print diagnostic data, global sums	= .true.
kitd	0 /1	If 0, delta function ITD approx. If 1, linear remapping ITD approx.	= 1
kcatbound	0/1	If 0, original category boundary formula If 1, new category boundary formula	= 1
kdyn	0 /1	If 0, EVP dynamics OFF If 1, EVP dynamics ON	= 1
kstrength	0 /1	If 0, [15] ice strength formulation If 1, [35] ice strength formulation	= 1
krdg_partic	0/1	If 0, old ridging participation function If 1, new ridging participation function	= 1
krdg_redist	0/1	If 0, old ridging redistribution function If 1, new ridging redistribution function	= 0
evp_damping	true/false	If true, damp elastic waves, [17]	= .false.
advection	remap, mpdata, upwind	remap: Linear remapping advection mpdata: 2nd order	= 'remap'

Name	Type/Options	Description	Default Values / Directory Location
		MPDATA upwind: 1st order MPDATA	
grid_type	rectangular, displaced_pole	rectangular: Defined in *rectgrid* displaced_pole: Read from file in *popgrid*	= 'pips'
grid_file	filename	Name of grid file to be read	= 'grid_cice_1280x720.r'
kmt_file	filename	Name of land mask file to be read	= 'kmt'
dump_file	filename prefix	Output file for restart dump	= 'iced'
restart_dir	path/	path to restart directory	= '/scr/posey/pips3c/'
pointer_file	pointer filename	Contains restart filename	= '/scr/posey/pips3c/ice.restart_file'
hist_dir	path/	path to history output directory	= '/scr/posey/pips3c/'
history_file	filename prefix	Output file for history	= 'iceh'
diag_file	filename	Diagnostic output file	= 'ice_diag.d'
oceanmixed_ice	true/false	Active ocean mixed layer calculation	= .true.
albicev	$0 < \alpha < 1$	Visible ice albedo for thicker ice	= 0.65
albicei	$0 < \alpha < 1$	Near infrared ice albedo for thicker ice	= 0.65
albsnowv	$0 < \alpha < 1$	Visible, cold snow albedo	= 0.85
albsnowi	$0 < \alpha < 1$	Near infrared, cold snow albedo	= 0.85
ycycle	integer	No. of years in forcing data cycle	= 1
fyear_init	yyyy	First year of atmospheric forcing data	= 2008
atm_data_dir	path/	Path to atm forcing data directory	= '/scr/posey/pips3c/data_in/'
ocn_data_dir	path/	Path to oceanic forcing data directory	= '/scr/posey/pips3c/data_in/'

Table 1: Ice namelist parameters.

4.1.3.2 Ice Fields Namelist (icefields_nml)

Name	Description	Default Values
f_hi	Ice thickness	= .true.
f_hs	Snow thickness	= .true.
f_Tsfc	Temperature of ice/snow top surface (in category *n*)	= .false.
f_aice	Ice concentration	= .true.
f_uvel	*x*-component of velocity	= .true.
f_vvel	*y*-component of velocity	= .true.
f_fswdn	Incoming shortwave radiation down	= .false.
f_flwdn	Incoming longwave radiation down	= .fasle.
f_snow	Snowfall rate	= .false.
f_snow_ai	Snowfall rate weighted by aice	= .false.
f_rain	Rainfall rate	= .false.
f_rain_ai	Rainfall rate weighted by aice	= .false.
f_sst	Sea surface temperature	= .true.
f_sss	Sea surface salinity	= .true.
f_uocn	Ocean current, *x* direction	= .true.
f_vocn	Ocean current, *y* direction	= .true.
f_frzmlt	Freezing/melting potential	= .false.
f_fswabs	Absorbed shortwave radiation	= .false.
f_fswabs_ai	Absorbed shortwave radiation weighted by aice	= .false.
f_albsni	Snow/ice broad band albedo	= .false.

Name	Description	Default Values
f_alvdr		= .false.
f_alidr		= .false.
f_flat	Latent heat flux	= .false.
f_flat_ai	Latent heat flux weighted by aice	= .false.
f_fsens	Sensible heat flux	= .false.
f_fsens_ai	Sensible heat flux weighted by aice	= .false.
f_flwup	Incoming longwave radiation upward	= .false.
f_flwup_ai	Incoming longwave radiation upward weighted by aice	= .false.
f_evap	Evaporation water flux	= .false.
f_evap_ai	Evaporation water flux weighted by aice	= .false.
f_Tref	2m atmospheric reference temperature	= .false.
f_Qref	2 m atmospheric reference specific humidity	= .false.
f_congel	Basal ice growth	= .false.
f_frazil	Frazil ice growth	= .false.
f_snoice	Snow-ice formation	= .false.
f_meltt	Top ice melt	= .false.
f_meltb	Basal ice melt	= .false.
f_meltl	Lateral ice melt	= .false.
f_fresh	Fresh water flux to ocean	= .false.
f_fresh_ai	Fresh water flux to ocean weighted by aice	= .false.
f_fsalt	Net salt flux to ocean	= .false.
f_fsalt_ai	Net salt flux to ocean weighted by aice	= .false.

Name	Description	Default Values
f_fhnet	Net heat flux to ocean	= .false.
f_fhnet_ai	Net heat flux to ocean weighted by aice	= .true.
f_fswthru	Shortwave penetrating to ocean	= .false.
f_fswthru_ai	Shortwave penetration to ocean weighted by aice	= .false.
f_strairx	Stress on ice by air in the *x*-direction (centered in U cell)	= .true.
f_strairy	Stress on ice by air in *y*-direction (centered in T cell)	= .true.
f_strtltx	Surface stress due to sea surface slope in x-direction	= .false.
f_strtlty	Surface stress due to sea surface slope in y-direction	= .false.
f_strcorx	Coriolis stress (x)	= .false.
f_strcory	Coriolis stress (y)	= .false.
f_strocnx	Ice-ocean stress, *x* dir. (U cell)	= .true.
f_strocny	Ice-ocean stress, *y*-dir. (T cell)	= .true.
f_strintx	Divergence of internal ice stress, *x*-direction	= .true.
f_strinty	Divergence of internal ice stress, *y*-direction	= .true.
f_strength	Ice strength (pressure)	= .true.
f_opening	Lead area opening rate	= .true.
f_divu	Strain rate I component, velocity divergence	= .false.
f_shear	Strain rate II component	= .false.
f_sig1	Principal stress components (diagnostic)	= .false.

Name	Description	Default Values
f_sig2	Principal stress components (diagnostic)	= .false.
f_dvidtt	Ice volume tendency due to thermodynamics	= .false.
f_dvidtd	Ice volume tendency due to dynamics/transport	= .false.
f_daidtt	Ice area tendency due to thermodynamics	= .false.
f_daidtd	Ice area tendency due to dynamics/transport	= .false.
f_mlt_onset	Day of year that surface melt begins	= .false.
f_frz_onset	Day of year that freezing begins	= .false.
f_dardg1dt	Ice area ridging rate	= .false.
f_dardg2dt	Ridge area formation rate	= .false.
f_dvirdgdt	Ice volume ridging rate	= .false.
f_hisnap	Ice volume snapshot	= .false.
f_aisnap	Ice area snapshot	= .false.
f_aice1 (through 5)	Ice concentration in grid cell in categories 1 through 5	= .true.
f_aice6 (through 10)	Ice concentration in grid cell in categories 6 through 10	= .false.
f_vice1 (through 5)	Volume per unit area of ice in categories 1 through 5	= .true.
f_vice6 (through 10)	Volume per unit area of ice in categories 6 through 10	= .false.

Table 2: Ice field namelist parameters.

4.1.4 PIPS 3.0 Macros File

Macros.AIX.txt is a file containing macros necessary for compiling the PIPS 3.0 code on the NAVOCEANO IBM platform “Babbage”. A detailed description of the macro is provided in the PIPS 3.0 User’s Manual [2]. The primary macros utilized in this file are summarized below.

Macro	Description
`-lmass`	IBM - tuned intrinsic library
`-qsmp=noauto`	enables SMP directives, but does not add any
`-qstrict`	don't turn divides into multiplies, etc
`-qhot`	higher-order -transformations (eg. loop padding)
`-qalias=noaryoverlp`	assume no array overlap with respect to equivalence, etc
`-qmaxmem=1`	memory available to compiler during optimization
`-qipa=level=2`	InterProcedure Analysis (eg. inlining) => slow compiles
`-p –pg`	enable profiling (use in both FFLAGS and LDFLAGS)
`-qreport`	for smp/omp only
`-bmaxdata:0x80000000`	use maximum allowed data segment size
`-g`	always leave it on because overhead is minimal
`-qflttrap=...`	enable default sigtrap (core dump)
`-C`	runtime array bounds checking (runs slow)
`-qinitauto=...`	initializes automatic variables

Table 3: PIPS 3.0 macros from Macros.AIX.txt file.

4.1.5 PIPS 3.0 Makefile

A makefile is used in the execution of PIPS 3.0 at NAVOCEANO. A comprehensive description of the makefile is provided in the PIPS 3.0 User's Manual [2]. The command line variables and usage examples are defined below.

Command-line variables:

1. MACFILE=<file> ~ The macros definition file to use/include in a run.
2. EXEC=<name> ~ The name given to an executable. The default is *a.out*.
3. VPATH=<vpath> ~ VPATH, default is . (*cwd* only).
4. SRCS=<files> ~ A list of source files. The default is all *.c .F .F90* files in VPATH.
5. VPFILE=<file> ~ A file with a list of directories. It is used to create VPATH.
6. SRCFILE=<file> ~ A file with a list of source files. It is used to create SRCS.
7. DEPGEN=<exec> ~ A dependency generator utility, with a default of *makdep*.
8. <macro defns> ~ Any macro definitions found in this file or the included MACFILE will be over-ridden by command line macro definitions.
9. MODEL=<model> ~ A standard macro definition, often found in the included MACFILE. It is used to trigger special compilation flags.

Usage examples:

```
 % gmake MACFILE=Macros.AIX VPFILE=Filepath MODEL=ccm3 EXEC=atm
 % gmake MACFILE=Macros.AIX VPFILE=Filepath SRCFILE=Srclist EXEC=pop
```

```
% gmake MACFILE=Macros.C90 VPATH="dir1 dir2" SRCS="file1.c file2.F90"
% gmake MACFILE=Macros.SUN SRCS="test.F"
```

4.2 PIPS 3.0 Software Organization and Implementation

4.2.1 Logical Component Call Trees

4.2.1.1 Primary Tree (beginning at the program '`icemodel`')

```
icemodel
|
+-ice_albedo-+-ice_kinds_mod
|            |
|            +-ice_domain-+-ice_kinds_mod
|                         |
|                         +-ice_model_size--ice_kinds_mod
|
+-ice_calendar--ice_constants-+-ice_kinds_mod
|                             |
|                             +-ice_domain
|
+-ice_coupling-+-ice_kinds_mod
|              |
|              +-ice_model_size
|              |
|              +-ice_constants
|              |
|              +-ice_calendar
|              |
|              +-ice_state-+-ice_kinds_mod
|              |           |
|              |           +-ice_model_size
|              |           |
|              |           +-ice_domain
|              |
|              +-ice_flux-+-ice_kinds_mod
|              |          |
|              |          +-ice_domain
|              |          |
|              |          +-ice_constants
|              |
|              +-ice_albedo
|              |
|              +-ice_mpi_internal-+-ice_kinds_mod
|              |                  |
|              |                  +-ice_domain
|              |
|              +-ice_timers-+-ice_kinds_mod
|              |            |
|              |            +-ice_constants
|              |
|              +-ice_fileunits--ice_kinds_mod
|              |
|              +-ice_work-+-ice_kinds_mod
|                         |
|                         +-ice_domain
|
+-ice_diagnostics-+-ice_domain
|                 |
|                 +-ice_constants
|                 |
|                 +-ice_calendar
|
|                  |
|                  +-ice_fileunits
|                  |
|                  +-ice_work
|
+-ice_domain
|
+-ice_dyn_evp-+-ice_kinds_mod
|              |
|              +-ice_domain
|              |
|              +-ice_constants
|              |
|              +-ice_state
|              |
|              +-ice_work
|
+-ice_fileunits
|
+-ice_flux_in-+-ice_kinds_mod
|              |
|              +-ice_domain
|              |
|              +-ice_constants
|              |
|              +-ice_flux
|              |
|              +-ice_calendar
|              |
|              +-ice_read_write-+-ice_model_size
|              |                 |
|              |                 +-ice_domain
|              |                 |
|              |                 +-ice_mpi_internal
|              |                 |
|              |                 +-ice_fileunits
|              |                 |
|              |                 +-ice_work
|              |
|              +-ice_fileunits
|
+-ice_grid-+-ice_kinds_mod
|           |
|           +-ice_constants
|           |
|           +-ice_domain
|           |
|           +-ice_fileunits
|           |
|           +-ice_mpi_internal
|           |
|           +-ice_work
|
+-ice_history-+-ice_kinds_mod
|              |
|              +-ice_domain
|              |
|              +-ice_read_write
|              |
|              +-ice_fileunits
|              |
|              +-ice_work
|
+-ice_init--ice_domain
|
+-ice_itd-+-ice_kinds_mod
|          |
|          +-ice_model_size
|          |
|         +-ice_constants
|         |
|         +-ice_state
|         |
|         +-ice_fileunits
|
+-ice_itd_linear-+-ice_model_size
|                |
|                +-ice_kinds_mod
|                |
|                +-ice_domain
|                |
|                +-ice_constants
|                |
|                +-ice_state
|                |
|                +-ice_itd
|                |
|                +-ice_calendar
|                |
|                +-ice_fileunits
|
+-ice_kinds_mod
|
+-ice_mechred-+-ice_model_size
|             |
|             +-ice_constants
|             |
|             +-ice_state
|             |
|             +-ice_itd
|             |
|             +-ice_grid
|             |
|             +-ice_fileunits
|             |
|             +-ice_domain
|             |
|             +-ice_calendar
|             |

|             +-ice_work
|
+-ice_mpi_internal
|
+-ice_ocean-+-ice_kinds_mod
|           |
|           +-ice_constants
|
+-ice_scaling-+-ice_domain
|             |
|             +-ice_kinds_mod
|             |
|             +-ice_constants
|             |
|             +-ice_state
|             |
|             +-ice_flux
|             |
|             +-ice_grid
|
+-ice_therm_vertical-+-ice_model_size
|                    |
|                    +-ice_kinds_mod
|                    |
|                    +-ice_domain
|                    |
|                    +-ice_fileunits
|                    |
|                    +-ice_constants
|                    |
|                    +-ice_calendar
|                    |
|                    +-ice_grid
|                    |
|                    +-ice_state
|                    |
|                    +-ice_flux
|                    |
|                    +-ice_itd
|                    |
|                    +-ice_diagnostics
|
+-ice_therm_itd-+-ice_kinds_mod
|               |
|               +-ice_model_size
|               |
|               +-ice_constants
|               |
|               +-ice_domain
|               |
|               +-ice_state
|               |
|               +-ice_flux
|               |
|               +-ice_diagnostics
|               |
|               +-ice_calendar
|               |
|               +-ice_grid
|               |
|               +-ice_itd
|
+-ice_timers
|
+-ice_transport_mpdata-+-ice_model_size
|                      |
|                      +-ice_domain
|                      |
|                      +-ice_constants
|                      |
|                      +-ice_grid
|                      |
|                      +-ice_fileunits
|                      |
|                      +-ice_init
|                      |
|                      +-ice_work
|
+-ice_transport_remap-+-ice_model_size
|                     |
|                     +-ice_kinds_mod
|                     |
|                     +-ice_domain
|                     |
|                     +-ice_constants
|                     |
|                     +-ice_grid
|                     |
|                     +-ice_fileunits
|                     |
|                     +-ice_calendar
|                     |
|                     +-ice_state
|                     |
|                     +-ice_timers
|                     |
|                     +-ice_itd
|                      |
|                      +-ice_work
|
+-(shr_msg_stdio)
|
+-setup_mpi-+-ice_mpi_internal
|           |
|           +-ice_coupling
|           |
|           +-ice_coupling_setup-+-(cpl_interface_init)
|           |                    |
|           |                    +-(shr_sys_flush)
|           |
|           +-(mpi_init)
|           |
|           +-(mpi_comm_dup)
|           |
|           +-(mpi_comm_size)

|           |
|           +-(mpi_comm_rank)
|           |
|           +-(abort_ice)
|           |
|           +-(mpi_type_vector)
|           |
|           +-(mpi_type_commit)
|
+-ice_timer_clear
|
+-ice_timer_start--timers-+-(mpi_wtime)
|                         |
|                         +-(irtc_rate)
|                         |
|                         +-(rtc)
|
+-init_constants
|
+-input_data-+-ice_albedo
|            |
|            +-ice_diagnostics
|            |
|            +-ice_history
|            |
|            +-ice_calendar
|            |
|            +-ice_dyn_evp
|            |
|            +-ice_flux_in
|            |
|            +-ice_coupling
|            |
|            +-ice_bcast_iscalar--(mpi_bcast)
|            |
|            +-(abort_ice)
|            |
|            +-ice_bcast_rscalar--(mpi_bcast)
|            |
|            +-ice_bcast_char--(mpi_bcast)
|            |
|            +-ice_bcast_logical--(mpi_bcast)
|
+-init_grid
| |
| +-global_scatter-+-ice_model_size
| |                |
| |                +-ice_constants
| |                |
| |                +-(mpi_irecv)
| |                 |
| |                 +-(mpi_isend)(mpi_sendrecv)
| |                 |

| |                 +-(mpi_irecv)
| |                 |
| |                 +-(mpi_isend)
| |                 |
| |                 +-(mpi_wait)
| |
| +-popgrid-+-ice_read_write
| |         |
| |         +-ice_open
| |         |
| |         +-ice_read-+-global_scatter
| |                    |
| |                    +-ice_bcast_logical
| |
| +-pipsgrid-+-ice_read_write
| |          |
| |          +-ice_open
| |          |
| |          +-ice_read
| |
| +-columngrid-+-global_scatter
| |            |
| |            +-ice_model_size
| |            |
| |            +-(abort_ice)
| |
| +-rectgrid-+-global_scatter
| |          |
| |          +-ice_model_size
| |
| +-bound--bound_ijn-+-ice_timers
| |                  |
| |                  +-ice_timer_start
| |                  |
| |                  +-+-ice_timer_stop--timers
| |
| +-(abort_ice)
| |
| +-tlatlon-+-global_gather-+-ice_model_size
| |         |               |
| |         |               +-ice_constants
| |         |               |
| |         |               +-(mpi_irecv)
| |         |               |
| |         |               +-(mpi_barrier)
| |         |               |
| |         |               +-(mpi_isend)
| |         |               |
| |         |               +-(mpi_wait)
| |         |               |
| |         |               +-(mpi_waitall)
| |         |
| |         +-global_scatter
| |         |
| |         +-ice_model_size
| |         |
| |         +-bound
| |
| +-makemask--bound
|
+-init_remap-+-bound
|            |
|            +-(abort_ice)
|
+-init_calendar
```

```
|
+-init_hist-+-ice_bcast_iscalar
|           |
|           +-ice_bcast_logical
|           |
|           +-ice_constants
|           |
|           +-ice_calendar
|           |
|           +-ice_flux
|           |
|           +-(abort_ice)
|           |
|           +-(shr_sys_flush)
|
+-init_evp-+-ice_calendar
|          |
|          +-ice_fileunits
|          |
|          +-ice_flux
|          |
|          +-(ulat)
|          |
|          +-(bound)
|
+-init_flux-+-ice_constants
|           |
|           +-ice_flux
|           |
|           +-init_flux_atm--ice_state
|           |
|           +-init_flux_ocn
|
+-init_thermo_vertical--ice_itd
|
+-init_mechred
|
+-init_itd--(abort_ice)
|
+-calendar--ice_fileunits
|
+-init_cpl
| |
| +-calendar
| |
| +-ice_bcast_iscalar
| |
| +-ice_bcast_rscalar
| |
| +-(shr_sys_flush)
| |
| +-(tlon)
| |
| +-(tlat)
| |
| +-(tarea)
| |
| +-(hm)
| |
| +-(cpl_interface_contractinit)
| |
| +-(cpl_interface_ibufrecv)
| |
| +-to_coupler
|   |
|   +-get_sum--ice_global_real_sum--(mpi_allreduce)
|   |
|   +-ice_timer_start
|   |
|   +-ice_timer_stop
|   |
|   +-(anglet)
|   |
|   +-(tmask)
|   |
|   +-(shr_sys_flush)
|   |
|   +-(cpl_interface_contractsend)
|   |
|   +-(tarea)
|   |
|   +-(bound)
|
+-init_getflux-+-ncar_files--file_year
|              |
|              +-sss_clim-+-ice_work
|              |          |
|              |          +-ice_open
|              |          |
|              |          +-ice_read
|              |          |
|              |          +-complete_getflux_ocn
|              |
|              +-sst_ic-+-ice_open
|                       |
|                       +-ice_read
|
+-init_state-+-ice_model_size
|            |
|            +-ice_constants
|            |
|            +-ice_flux
|            |
|            +-ice_grid
|            |
|            +-ice_state
|            |
|            +-bound
|            |
|            +-(hin_max)
|            |
|            +-(ilyr1)
|            |
|            +-(tmlt)
|            |
|            +-(aggregate)
|            |
|            +-(bound_aggregate)
|
+-restartfile-+-ice_model_size
|             |
|             +-ice_mpi_internal
|             |
|             +-ice_open
|             |
|             +-ice_bcast_iscalar
|             |
|             +-ice_bcast_rscalar
|             |
|             +-ice_read
|             |
|             +-ice_flux
|             |
|             +-ice_grid
|             |
|             +-ice_calendar
|             |
|             +-ice_state
|             |
|             +-ice_dyn_evp
|             |
|             +-ice_coupling
|             |
|             +-lenstr
|             |
|             +-(bound_state)
|             |
|             +-bound
|             |
|             +-(aggregate)
|             |
|             +-(bound_aggregate)
|
+-albedos-+-ice_constants
|         |
|         +-ice_state
|         |
|         +-(tmask)
|
+-init_diags-+-ice_mpi_internal
|            |
|            +-global_gather
|            |
|            +-(tlat_g)
|            |
|            +-(tlon_g)
|
+-init_diagnostics--ice_state
|
+-from_coupler-+-get_sum
|              |
|              +-ice_timer_start
|              |
|              +-ice_timer_stop
|              |
|              +-(cpl_interface_contractrecv)
|              |
|              +-(tarea)
|              |
|              +-(hm)
|              |
|              +-(bound)
|              |
|              +-(anglet)
|              |
|              +-(t2ugrid)
|
+-getflux-+-sss_sst_restore-+-interp_coeff_monthly
|         |                 |
|         |                 +-read_clim_data-+-ice_open
|         |                 |                |
|         |                 |                +-ice_read
|         |                 |                |
|         |                 |                +-ice_diagnostics
|         |                 |
|         |                 +-interpolate_data
|         |                 |
|         |                 +-complete_getflux_ocn
|         |
|         +-ncar_bulk_dat-+-interp_coeff_monthly
|         |               |
|         |               +-read_clim_data
|         |               |

|         |               +-interpolate_data
|         |               |
|         |               +-interp_coeff
|         |               |
|         |               +-read_data-+-ice_open
|         |                           |
|         |                           +-ice_read
|         |                           |
|         |                           +-ice_diagnostics
|         |                           |
|         |                           +-file_year
|         |
|         +-prepare_forcing--ice_state
|
+-init_mass_diags-+-ice_mpi_internal
|                 |
|                 +-ice_state
|                 |
|                 +-get_sum
|                 |
|                 +-global_gather
|
+-mixed_layer-+-ice_flux
|             |
|             +-ice_calendar
|             |
|             +-ice_grid
|             |
|             +-ice_state
|             |
|             +-ice_albedo
|             |
|             +-(atmo_boundary_layer)
|
+-thermo_vertical
| |
| +-ice_work
| |
| +-init_flux_atm
| |
| +-init_diagnostics
| |
| +-merge_fluxes--ice_state
| |
| +-ice_timers
| |
| +-ice_ocean
| |
| +-ice_timer_start
| |
| +-frzmlt_bottom_lateral
| |
| +-init_thermo_vars
| |
| +-(atmo_boundary_layer)
| |
| +-init_vertical_profile-+-print_state-+-ice_model_size
| |                       |             |
| |                       |             +-ice_kinds_mod
| |                       |             |
| |                       |             +-ice_state
| |                       |             |
| |                       |             +-ice_flux
| |                       |             |
| |                       |             +-(ilyr1)
| |                       |
| |                       +-(abort_ice)
| |
| +-temperature_changes-+-print_state
| |                     |
| |                     +-conductivity
| |                     |
| |                     +-absorbed_solar--ice_albedo
| |                     |
| |                     +-surface_fluxes
| |                     |
| |                     +-tridiag_solver
| |                     |
| |                     +-(abort_ice)
| |
| +-thickness_changes
| |
| +-conservation_check_vthermo-+-print_state
| |                            |
| |                            +-(abort_ice)
| |
| +-add_new_snow
| |
| +-update_state_vthermo
| |
| +-ice_timer_stop
|
+-scale_fluxes--ice_albedo
|
+-to_coupler
|
+-thermo_itd-+-aggregate-+-ice_domain
|            |           |
|            |           +-ice_flux
|            |           |
|            |           +-ice_grid
|            |
|            +-init_flux_ocn
|            |
|            +-reduce_area--ice_grid
|            |
|            +-rebin (44)-+-ice_grid
|            |            |
|            |            +-shift_ice-+-ice_flux
|            |                        |
|            |                        +-ice_work
|            |                        |
|            |                        +-(abort_ice)
|            |
|            +-zap_small_areas-+-ice_flux
|            |                 |
|            |                 +-ice_calendar
|            |                 |
|            |                 +-(abort_ice)
|            |
|            +-ice_timers
|            |
|            +-ice_itd_linear
|            |
|            +-ice_therm_vertical
|            |
|            +-ice_timer_start
|            |
|            +-ice_timer_stop
|            |
|            +-linear_itd-+-aggregate_area--(abort_ice)
|            |            |
|            |            +-column_sum
|            |            |
|            |            +-column_conservation_check--(abort_ice)
|            |            |
|            |            +-shift_ice
|            |            |
|            |            +-fit_line
|            |
|            +-add_new_ice-+-column_sum
|            |             |
|            |             +-column_conservation_check
|            |
|            +-lateral_melt
|            |
|            +-freeboard
|
+-evp-+-ice_timers
|     |
|     +-ice_timer_start
|     |
|     +-evp_prep-+-ice_flux
|     |          |
|     |          +-ice_calendar
|     |          |
|     |          +-(bound)
|     |          |
|     |          +-(tmask)
|     |          |
|     |          +-(t2ugrid)
|     |          |
|     |          +-(to_ugrid)
|     |          |
|     |          +-(iceumask)
|     |          |
|     |          +-(ice_strength)
|     |          |
|     |          +-(icetmask)
|     |          |
|     |          +-(umask)
|     |
|     +-stress-+-(cyp)
|     |        |
|     |        +-(dyt)
|     |        |
|     |        +-(cxp)
|     |        |
|     |        +-(dxt)
|     |        |
|     |        +-(cym)
|     |        |
|     |        +-(cxm)
|     |        |

|     |        +-(tarear)
|     |        |
|     |        +-(tinyarea)
|     |
|     +-stepu-+-ice_flux
|     |       |
|     |       +-(dxt2)
|     |       |
|     |       +-(dyt2)
|     |       |
|     |       +-(dyt4)
|     |       |
|     |       +-(dxhy)
|     |       |
|     |       +-(dyhx)
|     |       |
|     |       +-(dxt4)
|     |       |
|     |       +-(bound_narr_ne)
|     |       |
|     |       +-(iceumask)
|     |       |
|     |       +-(uarear)
|     |
|     +-(bound_sw)
|     |
|     +-evp_finish-+-ice_flux
|     |            |
|     |            +-(iceumask)
|     |            |
|     |            +-(u2tgrid)
|     |
|     +-ice_timer_stop
|
+-transport_remap-+-bound
|                 |
|                 +-bound_sw--bound_ijn
|                 |
|                 +-bound_state-+-ice_grid
|                 |             |
|                 |             +-bound_narr--bound_ijn
|                 |
|                 +-bound_narr
|                 |
|                 +-ice_timer_start
|                 |
|                 +-ice_timer_stop
|                 |
|                 +-departure_points
|                 |
|                 +-locate_triangles
|                 |
|                 +-triangle_coordinates
|                 |
|                 +-make_masks
|                 |
|                 +-conserved_sums-+-ice_mpi_internal
|                 |                |
|                 |                +-ice_global_real_sum
|                 |
|                 +-construct_fields--limited_gradient
|                 |
|                 +-flux_integrals
|                 |
|                 +-update_fields--(abort_ice)
|                 |
|                 +-global_conservation--(abort_ice)
|                 |
|                 +-load_tracers
|                 |
|                 +-local_max_min--bound
|                 |
|                 +-check_monotonicity--(abort_ice)
|                 |
|                 +-unload_tracers
|
+-transport_mpdata-+-bound_narr
|                  |
|                  +-ice_flux
|                  |
|                  +-ice_timers
|                  |
|                  +-ice_state
|                  |
|                  +-ice_itd
|                  |
|                  +-ice_timer_start
|                  |
|                  +-check_state--ice_flux
|                  |
|                  +-mpdata-+-bound
|                  |        |
|                  |        +-bound_narr
|                  |        |
|                  |        +-ice_calendar
|                  |        |
|                  |        +-ice_state
|                  |        |
|                  |        +-(abort_ice)
|                  |
|                  +-ice_timer_stop
|
+-ridge_ice-+-ice_timers
|           |
|           +-ice_flux
|           |
|           +-ice_timer_start
|           |
|           +-ridge_prep--asum_ridging

|           |
|           +-ridge_shift-+-column_sum
|           |             |
|           |             +-column_conservation_check
|           |             |
|           |             +-(abort_ice)
|           |
|           +-asum_ridging
|           |
|           +-(abort_ice)
|           |
|           +-ice_timer_stop
|
+-zap_small_areas
|
+-rebin
|
+-aggregate
|
+-scale_hist_fluxes
|
+-runtime_diags-+-ice_model_size
|               |
|               +-ice_flux
|               |
|               +-ice_albedo
|               |
|               +-ice_mpi_internal
|               |
|               +-ice_state
|               |
|               +-(mask_n)
|               |
|               +-(mask_s)
|               |
|               +-ice_global_real_maxval--(mpi_allreduce)
|               |
|               +-get_sum
|               |
|               +-(ulat)
|               |
|               +-(ulon)
|               |
|               +-(opening)
|               |
|               +-(dardg1dt)
|               |
|               +-(strintx)
|               |
|               +-(strinty)
|               |
|               +-(athorn)
|               |
|                +-ice_global_real_sum
|                |
|                +-global_gather
|                |
|                +-(shr_sys_flush)
|
+-ice_write_hist-+-ice_flux
|                 |
|                 +-ice_albedo
|                 |
|                 +-ice_grid
|                 |
|                 +-ice_calendar
|                 |
|                 +-ice_state
|                 |
|                 +-ice_dyn_evp
|                 |
|                 +-ice_constants
|                 |
|                 +-(opening)

|                 |
|                 +-(dardg1dt)
|                 |
|                 +-(dardg2dt)
|                 |
|                 +-(dvirdgdt)
|                 |
|                 +-principal_stress
|                 |
|                 +-icecdf-+-global_gather
|                          |
|                          +-ice_model_size
|                          |
|                          +-ice_mpi_internal
|                          |
|                          +-ice_constants
|                          |
|                          +-ice_grid
|                          |
|                          +-ice_calendar
|                          |
|                          +-lenstr
|                          |
|                          +-(nf_create)
|                          |
|                          +-(nf_def_dim)
|                          |
|                          +-(nf_def_var)
|                          |
|                          +-(nf_put_att_text)
|                          |
|                          +-(nf_put_att_real)
|                          |
|                          +-(nf_enddef)
|                          |
|                          +-(nf_inq_varid)
|                          |
|                          +-(nf_put_var_real)
|                          |
|                          +-(nf_put_vara_real)
|                          |
|                          +-(nf_close)
|
+-dumpfile-+-ice_model_size
|          |
|          +-ice_open
|          |
|          +-ice_write--global_gather
|          |
|          +-ice_flux
|          |
|          +-ice_grid
|          |
|          +-ice_calendar
|          |
|          +-ice_state
|          |
|          +-ice_dyn_evp
|          |
|          +-ice_coupling
|          |
|          +-lenstr
|
+-ice_timer_stop
|
+-ice_timer_print-+-ice_domain
|                 |
|                 +-ice_mpi_internal
|                 |
|                 +-ice_fileunits
|                 |
|                 +-ice_global_real_minval--(mpi_allreduce)
|                 |
|                 +-ice_global_real_maxval
|
+-end_run--(mpi_finalize)
|
+-exit_coupler-+-(mpi_abort)
               |
               +-(cpl_interface_finalize)
```

4.2.1.2 Detached Call Trees (for extraneous modules and subroutines)

```
1. ice_exit--ice_kinds_mod


2. atmo_boundary_layer--ice_grid

3. ice_global_real_minmax-+-ice_fileunits
                          |
                          +-ice_global_real_minval
                          |
                          +-ice_global_real_maxval

4. bound_aggregate-+-ice_grid
                   |
                   +-bound

5. bound_narr_ne--bound_ijn

6. u2tgrid-+-bound
           |
           +-to_tgrid

7. ice_atmo-+-ice_domain
            |
            +-ice_constants
            |
            +-ice_flux
            |
            +-ice_state
```

```
8. debug_ice-+-ice_kinds_mod
             |
             +-ice_itd
             |
             +-ice_diagnostics
             |
             +-ice_mpi_internal
             |
             +-ice_grid
             |
             +-print_state

9. ice_strength--ridge_prep

10. t2ugrid-+-bound
            |
            +-to_ugrid

11. abort_ice-+-ice_domain
              |
              +-ice_fileunits
              |
              +-ice_mpi_internal
              |
              +-(shr_sys_abort)
              |
              +-end_run
```

4.2.2 Directory Structure

The present PIPS 3.0 code distribution includes makefiles, input files and scripts. The primary directory is **pips3/**, and a run directory (**rundir**) is created upon the initial run of the **comp_ice** script.

+-**pips3/** - primary directory
  +-**README_V3.1** - basic information
  +-**bld/** - makefiles
    +-**Macros.<OS>** - macro definitions for the given operating system, used by
      Makefile.<OS>
    +-**Makefile.<OS>** - primary makefile for a given operating system
      (<std> works for most systems)
    +-**makedep.c** - perl script that determines module dependencies
  +-**clean_ice** - script that removes files from the compile directory
  +-**comp_ice** - script that sets up the run directory and compiles the code
  +-**doc/** - documentation
    +-**PIPS_SDD.doc**- software design description
    +-**PIPS_UM.doc**- user's manual
    +-**PIPS_VTR.doc**- validation test report
    +-**cicedoc.pdf** - CICE: the Los Alamos Sea Ice Model Doc and User's Manual
    +-**PDF/** - PDF documents of several publications related to CICE
  +-**ice.log.<OS>** - sample diagnostic output files
  +-**source/** - PIPS 3.0 source code.

+-**PIPS.F** - main program
+-**PIPS.F_debug-** debugging version of PIPS.F
+-**ice_albedo.F** - albedo parameterization
+-**ice_atmo.F** - stability-based parameterization for calculation of turbulent ice-atm fluxes
+-**ice_calendar.F** - keeps track of what time it is
+-**ice_constants.F** - physical and numerical constants and parameters
+-**ice_coupling.F** - interface with the flux coupler
+-**ice_diagnostics.F** - miscellaneous diagnostic and debugging routines
+-**ice_domain.F** - MPI subdomain sizes and related parallel processing info
+-**ice_dyn_evp.F** - elastic-viscous-plastic dynamics component
+-**ice_exit.F** - aborts the model, printing an error message
+-**ice_fileunits.F** - unit numbers for I/O
+-**ice_flux.F** -fluxes needed/produced by the model
+-**ice_flux_in.F** - Reads and interpolates forcing data for stand-alone ice model runs
+-**ice_grid.F** - grid and land masks
+-**ice_history.F** - netCDF output routines and restart read/write
+-**ice_init.F** - namelist and initializations
+-**ice_itd.F** - utilities for managing ice thickness distribution
+-**ice_itd_linear.F** - linear remapping for transport in thickness space
+-**ice_kinds_mod.F** - basic defnitions of reals, integers, etc.
+-**ice_mechred.F** - mechanical redistribution component (ridging)
+-**ice_model_size.F** - grid size and number of thickness categories and vertical layers
+-**ice_model_size.F.gx1**- specific ice_model_size.F for use by scripts with $<1^o>$ grid
+-**ice_model_size.Fgx3**- specific ice_model_size.F for use by scripts with $<3^o>$ grid
+-**ice_mpi_internal.F** - utilities for internal MPI parallelization
+-**ice_ocean.F** - mixed layer ocean model
+-**ice_read_write.F** - utilities for reading and writing files
+-**ice_scaling.F** - ice-area scaling of variables for the coupler
+-**ice_state.F** - essential arrays to describe the state of the ice
+-**ice_therm_itd.F** - thermodynamic changes related to ice thickness distribution (post-coupling)
+-**ice_therm_vertical.F** - vertical growth rates and fluxes (pre-coupling thermodynamics)
+-**ice_timers.F** - timing routines
+-**ice_transport_mpdata.F** - horizontal advection via MPDATA or upwind
+-**ice_transport_remap.F** - horizontal advection via incremental remapping
+-**ice_work.F** - globally accessible work arrays
+-**rundir/** - execution or "run" directory generated when the code is compiled using the comp_ice script
+-**cice** - code executable
+-**compile/** - directory containing object files, etc.
+-**grid** - horizontal grid file from pips/input_templates/
+-**ice.log.[ID]** - diagnostic output file
+-**ice_in** - namelist input data from pips/input_templates
+-**hist/iceh_mavg.[timeID].nc** - monthly average output history file
+-**kmt** - land mask file from pips/input_templates/

+-**run_ice** - batch run script file from pips/input_templates/

5.0 PIPS 3.0 DETAILED DESIGN

The following sections give a detailed description of the purpose, variables, logic, and constraints for the software elements in the model.

5.1 Constraints and Limitations

1. Fluxes sent to the coupler could have incorrect values in grid cells that transform from an ice-free state to having ice during the given time step, or vice versa, due to scaling by the ice area. The flux coupler must have area scaling so that the ice and land models are treated reliably in the coupler (but note that the land area does not suddenly become zero in a grid cell, as does the ice area).
2. A significant fraction (more than 10%) of the total shortwave radiation is absorbed at the surface. It should, however, 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. Repairing the problem will require more albedo arrays to be added to the code.
3. The date-of-onset diagnostic variables, `melt_onset` and `frz_onset`, are not included in the restart file. These could therefore be incorrect for the current year if the run is restarted after Jan 1. Also, these variables were created with the Arctic in mind and may be incorrect for the Antarctic.
4. Timers are architecture dependent.
5. Local domains are not padded for uneven division of the global domain.

5.2 Logic and Basic Equations

5.2.1 Coupling with Ocean Model Components

PIPS 3.0 exchanges information with the NCOM ocean model via local file transfer. The fluxes and state variables passed between the PIPS 3.0 model and the ocean model are: ice concentration, SST, heat flux ice to ocean, and the ocean/ice stresses. The state variables passed between the ocean model and PIPS 3.0 are: ocean SST and ocean currents.
By convention, directional fluxes are positive downward.

The ice fraction a_i (`aice;` see Section 6.0 for a description of all typewritten equivalents) is the total fractional ice coverage of a grid cell. That is, in each cell,

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

where a_i is the sum of fractional ice areas for each category of ice. The ice fraction is employed by the flux coupler to merge fluxes from PIPS 3.0 with fluxes from the other components. For instance, 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 get 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 managed 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."

5.2.1.1 Atmosphere

Wind velocity, air density, specific humidity and potential temperature at the given level height z_o are used to calculate transfer coefficients in formulas for the surface wind stress and turbulent heat fluxes $\vec{\tau}_a$, F_s and F_l, as described below. Wind is quite possibly the main forcing mechanism for the ice motion, although the ice–ocean stress, Coriolis force, and slope of the ocean surface are also important [41]. 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 establishes the ice or snow surface temperature. As described in Section 5.2.2.5, these fluxes rely 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 flux of water evaporated to the atmosphere is given in terms of the latent heat, $F_{evap} = F_l / (L_{vap} + L_{ice})$, where L_{vap} and L_{ice} are latent heats of vaporization and fusion.

The snowfall precipitation rate (specified as liquid water equivalent and converted by PIPS 3.0 to snow depth) also contributes to the heat and water mass budgets of the ice layer. Although melt ponds usually develop on the ice surface in the Arctic and refreeze later in the autumn, reducing the total amount of fresh water that reaches the ocean and altering the heat budget of the ice, there currently is no melt pond parameterization. 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 [23]. For clarity, the equations are replicated 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" as

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

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

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

where the wind speed has a minimum value of 1 m/s. Ice motion is ignored in u^*, and T_{sfc} and Q_{sfc} represent the surface temperature and specific humidity, respectively. The latter is calculated by assuming a saturated surface temperature T_{sfc}, as described in Section 5.2.2.5.1.

The exchange coefficients c_u, c_θ and c_q are initialized as

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

and updated during a short iteration, as they depend on the turbulent scales. Here, z_{ref} is a reference height of 10 m and z_{ice} represents the roughness length scale for the given sea ice category. Υ is constrained to have a 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

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

Aside from the parameterization used in [23], profiles are used for the stable case from [22],

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

The coefficients are then updated as

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

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

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

where ρ_a is the density of air and c_p is its specific heat. Using [22] again, a constant has been added to the sensible heat flux coefficient to allow some heat to pass between the atmosphere and the ice surface in calm, stable 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 PIPS 3.0 does not use this quantity, it is quite convenient for the ice model to execute this calculation. The atmospheric reference temperature is returned to the flux coupler as a climate diagnostic. The same holds 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 5.2.2.5.1.

5.2.1.2 Ocean

New sea ice grows when the ocean temperature drops below its freezing temperature, $T_f = -\mu S$, where S is the seawater salinity and $\mu = 0.054$ is the ratio of the freezing temperature of brine to its salinity. The ocean models, either NCOM or HYCOM, perform this calculation; if the freezing/melting potential F_{frzmlt} is positive; its value is a certain quantity of frazil ice that has formed in one or more layers of the ocean and floated to the surface. (The ocean models assume that the quantity of new ice implied by the freezing potential actually forms.) Generally, this ice is added to the thinnest ice category. The new ice is created in the open water area of the grid cell to a specified minimum thickness. Therefore, if the open water area is nearly zero or if there is more new ice than will fit into the thinnest ice category, the new ice is then spread uniformly over the entire grid cell.

If F_{frzmlt} is negative, it is used to heat existing ice from below. Specifically, the sea surface temperature and salinity are used to compute an oceanic heat flux $F_w(|F_w| \le |F_{frzmlt}|)$ which is applied at the bottom of the ice. The fraction of the melting potential actually used to melt ice is returned to the coupler in F_{hnet}. The ocean models adjust their own heat budgets with this quantity, assuming that the rest of the flux stayed in the ocean.

In addition to runoff from rain and melted snow, the fresh water flux F_{water} includes ice meltwater from the top surface of the ice and water melted or frozen (a negative flux) at the bottom surface. This flux is computed as the net change of fresh water in the ice and snow volume over the coupling time step, prohibiting frazil ice formation and newly accumulated snow.

There is a flux of salt into the sea under melting conditions, and a (negative) flux when ocean water is freezing. However, melting sea ice ultimately freshens the top layer of the ocean, since it is much more saline than the ice. The PIPS 3.0 model passes the net flux of salt, F_{salt}, to the flux coupler, based on the net exchange in salt for ice in every category. In the present configuration, `ice_ref_salinity` is used in computing the salt flux, although the ice salinity utilized 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 5.2.2.5.1.

Many ice models compute the sea surface slope $\nabla H_{\circ}$ from geostrophic ocean currents supplied by an ocean model or other data source. In this case, the sea surface height $H_{\circ}$ is a prognostic variable in the NCOM and HYCOM models. The flux coupler provides the surface slope directly, instead of inferring it from the currents (The option of computing surface slope from the currents is provided in subroutine *evp_prep*.). The PIPS 3.0 model uses the surface layer currents $\vec{U}_w$ to derive the stress between the ocean and the ice, and subsequently the ice velocity $\vec{u}$. This stress, in relation to the ice,

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

is then transferred to the flux coupler (relative to the ocean) for use by the ocean model. Here, θ is the turning angle between geostrophic and surface currents, c_w is the ocean drag coefficient, ρ_w is the density of seawater (`dragw`$= c_w \rho_w$), and $\hat{k}$ represents the vertical unit vector. The turning angle is necessary if the top ocean model layers cannot 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 θ =0 is a good approximation. Here it is assumed that the top layer is sufficiently thin.

5.2.2 Model Components

The Arctic and Antarctic sea ice packs are comprised of combinations of open water, thin first-year ice, thicker multi-year ice, and thick pressure ridges. The thermodynamic and dynamic characteristics of the ice pack depend on the amount of ice lying in each thickness range. Thus the basic challenge in sea ice modeling is to describe the evolution of the ice thickness distribution (ITD) in both time and space.

The fundamental equation solved by PIPS 3.0 is from [43]:

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

where 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 represents the ice thickness distribution function. We define $g(x,h,t)dh$ as the fractional area covered by ice in the thickness range $(h, h+dh)$ at a given time and location.

Equation (2) is solved by sectioning the ice pack at each grid point into discrete thickness categories. The number of categories is set by the user, with a default value N_C =5. (Five categories, plus open water, are sufficient to simulate the yearly cycles of ice thickness, ice strength, and surface fluxes [6], [25]. 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 selected with greater resolution for small h, since the properties of the ice pack are particularly sensitive to the amount of thin ice [28]. The continuous function $g(h)$ is replaced by the discrete variable a_{in}, defined as the fractional area covered by ice in the thickness range (H_{n-1}, H_n). The fractional area of open water is denoted by a_{i0}, giving $\sum_{n=0}^{N_C} a_{in} = 1$ by definition.

Category boundaries are calculated in *init_itd* using one of two formulas. The old formula, from [26], assigns lower boundaries (in meters) of (0.0, 0.64, 1.39, 2.47, and 4.57) for categories 1 to 5 when N_C = 5. A new formula has been created for boundaries that are round numbers. This formula gives boundaries (0.0, 0.60, 1.40, 2.40, and 3.60) for N_C = 5. The old formula (`kcatbound` = 0 in the namelist) is the default. A user may substitute his or her own preferred boundaries in *init_itd.*

Besides the fractional ice area, a_{in}, the following state variables for each category n are defined as:

- v_{in}, the ice volume, equal to the product of a_{in} and the ice thickness h_{in}.
- v_{sn}, the snow volume, equal to the product of a_{in} and the snow thickness h_{sn}.
- e_{ink}, the internal ice energy in layer k, equal to the product of the ice layer volume, v_{in}/N_i, and the ice layer enthalpy, q_{ink}. Here N_i is the total number of ice layers, with a default value N_i =4, and q_{ink} is the negative of the energy needed to melt a unit volume of ice and raise its temperature to 0°C. This is discussed in Section 5.2.2.5. (NOTE: In the current code, $e_i < 0$ and $q_i < 0$ with $e_i = v_i q_i$.)
- e_{sn}, the snow energy, equal to the product of v_{sn} and the snow enthalpy, q_{sn}. At this

writing there is only one snow layer, but future versions of PIPS 3.0 will allow for multiple snow layers. (Similarly, $e_s < 0$ in the code.)

- T_{sfn}, the surface temperature.

Because 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 (2) illustrate 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.

The equation is solved by operator splitting in three stages, with two out of three terms on the right set to zero in each stage. Horizontal transport is computed using the incremental remapping scheme of [9] as adapted for sea ice by [26]. This scheme is discussed in Section 5.2.2.1. Ice is carried in thickness space through the remapping scheme of [25], as described in Section 5.2.2.2. The mechanical redistribution scheme, based on [43], [35], [16], [13], and [27] is outlined in Section 5.2.2.3. To solve the horizontal transport and ridging equations, we need the ice velocity u. To calculate transport in thickness space, ice growth rate f must be known in each thickness category. The elastic-viscous-plastic (EVP) ice dynamics scheme of [18] is used, as modified by [17], [19] and [20] to find the velocity, described in Section 5.2.2.4. Finally, the thermodynamic model of [7], discussed in Section 5.2.2.5, is used to compute f.

5.2.2.1 Horizontal Transport

Solving the continuity or transport equation,

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

for the fractional ice area in each thickness category n. Equation (3) describes the conservation of ice area in horizontal transport. It is taken from Equation (2) by discretizing g and ignoring the second and third terms on the right-hand side, which are handled separately (see Sections 5.2.2.2 and 5.2.2.3).

There are comparable conservation equations for ice volume, snow volume, ice energy, snow energy, and area-weighted surface temperature:

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

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

$$\frac{\partial e_{ink}}{\partial t} + \nabla \cdot (e_{ink}\mathbf{u}) = 0, \tag{6}$$

$$\frac{\partial e_{sn}}{\partial t} + \nabla \cdot (e_{sn}\mathbf{u}) = 0, \tag{7}$$

$$\frac{\partial \left(a_{in}T_{sfn}\right)}{\partial t} + \nabla \cdot (a_{in}T_{sfn}\mathbf{u}) = 0. \tag{8}$$

For simplicity, ice and snow are assumed to have constant densities, so that volume conservation is equal to mass conservation. Also transported is the fractional area of open water, using equation (3) with n =0. Including $N_C = 5$ and $N_i = 4$ there are 46 transport equations to be solved.

Three transport schemes are available, upwind, MPDATA [39] and the incremental remapping scheme of [9] as modified for sea ice by [26]. Because several fields are transported, the transport module is quite computationally expensive (almost half the total computer time) in runs using MPDATA. Although a cheaper first-order upwind scheme is available as an MPDATA option (see Section 4.1.3.1), it is advised that the incremental remapping method be used instead. This scheme has many desirable features:

- It conserves the quantity being transported (area, volume, or energy).
- It is non-oscillatory, meaning it does not create spurious ripples in the transported fields.
- It preserves tracer monotonicity, meaning 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 a great deal less diffusive than first-order schemes. The accuracy can be decreased locally to first order to preserve monotonicity.
- It is efficient for large amounts of categories or tracers. Much of the work is geometrical and is executed 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, thus defining incremental remapping as opposed to general remapping. This requirement leads to a CFL-like condition,

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

For greatly divergent velocity fields the maximum time step must be decreased by a factor of two to ensure that trajectories do not cross. Then again, ice velocity fields in climate models typically have small divergences per time step relative to the grid size.

The remapping algorithm may be summarized as follows:

1. Given mean values of the ice area and tracer fields in each grid cell, generate linear approximations of these fields. Limit the field gradients to conserve 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 these fields across the departure triangles to obtain the area, volume, and energy fluxes across each cell edge.
4. Transfer the fluxes over cell edges and update the state variables.

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

5.2.2.1.1 Reconstructing Area and Tracer Fields

Firstly, using the known values of the state variables, the ice area and tracer fields are reconstructed in every grid cell as linear functions of x and y. For each field the value at the cell center is computed (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 conserve monotonicity. When integrated over a grid cell, the reconstructed fields must have mean values equal to the known state variables, given by $\bar{a}$ for fractional area, $\tilde{h}$ for thickness, and $\hat{q}$ for enthalpy. The mean values are typically not equal to the values at the cell center. For instance, the mean ice area must equal the value at the centroid, which may not be at the cell center.

First consider the fractional ice area, the analog to fluid density ρ in [9]. For each thickness category a field $a(\mathbf{r})$ is constructed whose mean is $\bar{a}$, where $\mathbf{r}=(x, y)$ is the position vector relative to the cell center. That is, we require

$$\int_A a\,dA = \bar{a}A, \tag{9}$$

where $A=\int_A dA$ is the grid cell area. Equation (9) is satisfied if $a(\mathbf{r})$ takes the form

$$a(\mathbf{r}) = \bar{a} + \alpha_a < \nabla a > \cdot (\mathbf{r} - \bar{\mathbf{r}}), \tag{10}$$

where $< \nabla a >$ is the centered estimate of the area gradient in the cell, α_a is a limiting coefficient that implements monotonicity, and $\bar{r}$ is the cell centroid:

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

It follows from (10) that the ice area at the cell center (**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 tested using the triangle integration formulas described in Section 5.2.2.1.3. These means, combined with higher order means such as $\overline{x^2}$, $\overline{xy}$, and $\overline{y^2}$, are computed once and stored.

Next consider the snow and ice thickness and enthalpy fields. Thickness is analogous to the tracer concentration T in [9] but there is no analog in their study to the enthalpy. The reconstructed ice or snow thickness h(**r**) and enthalpy q(**r**) must satisfy

$$\int_A ahdA \quad = \quad \bar{a}\tilde{h}A, \tag{11}$$

$$\int_A ahqdA \quad = \quad \bar{a}\tilde{h}\hat{q}A. \tag{12}$$

Equations (11) and (12) are satisfied when h(**r**) and q(**r**) are denoted by

$$h(\mathbf{r}) = \tilde{h} + \alpha_h < \nabla h > \cdot (\mathbf{r} - \tilde{\mathbf{r}}), \tag{13}$$

$$q(\mathbf{r}) = \hat{q} + \alpha_q < \nabla q > \cdot (\mathbf{r} - \hat{\mathbf{r}}), \tag{14}$$

where α_h and α_q are limiting coefficients, $\tilde{\mathbf{r}}$ is the center of ice area,

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

and $\tilde{\mathbf{r}}$ is the center of ice or snow volume,

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

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

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

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

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

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

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

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 equations (13) and (14), the thickness and enthalpy at the cell center are denoted 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 dealt with the same way as ice or snow thickness, but it has no associated enthalpy.

Monotonicity is preserved by van Leer limiting. If $\overline{\phi}(i,j)$ is the mean value of some field in grid cell (i, j), centered gradients of $\overline{\phi}$ in the x and y directions are computed. They are also checked to see that the gradients provide values of ϕ within cell (i, j) that lie outside the range of $\overline{\phi}$ in the grid cell and its eight neighbors. Let $\overline{\phi}_{\max}$ and $\overline{\phi}_{\min}$ be the maximum and minimum values of $\overline{\phi}$ over the cell and its neighbors, and let $\phi_{\max}$ and $\phi_{\min}$ be the maximum and minimum values of the reconstructed ϕ within the cell. Since the reconstruction is linear, $\phi_{\max}$ and $\phi_{\min}$ are located at cell corners. If $\phi_{\max} > \bar{\phi}_{\max}$ or $\phi_{\min} < \bar{\phi}_{\min}$, the unlimited gradient is multiplied 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 does not need to be limited.

5.2.2.1.2 Locating Departure Triangles

The technique for locating departure triangles is discussed in detail by [9]. The basic idea is seen in Figure 1, which illustrates a shaded quadrilateral departure region whose contents are carried to the target or home grid cell, labeled *H*. The neighboring grid cells are tagged 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 the vectors. Rather than integrating across the entire departure region, we compute fluxes across cell edges. Departure regions are selected 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, over 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 region from the *N* cell is a quadrilateral that can be divided into two triangles with vertices *acd* and *ade*. Focusing on triangle *abc*, the coordinates of vertices *b* and *c* with respect to to the cell corner (vertex *a*) are determined first, using Euclidean geometry, to find vertex *c*. Then these three vertices are translated to a coordinate system located in the *NW* cell center. This translation is necessary for integrating fields (Section 5.2.2.1.3) in the coordinate system where they have been reconstructed (Section 5.2.2.1.1). Repeating this process for the north and east edges of each grid cell, we compute the vertices of all the departure triangles associated with each cell edge.

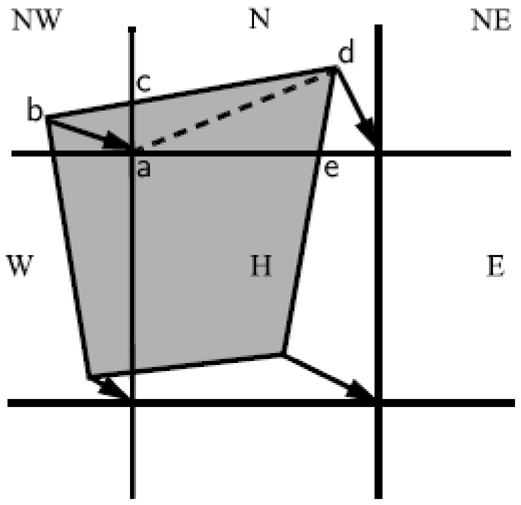

Figure 1: In incremental remapping, conserved quantities are remapped from the shaded departure region, which is a quadrilateral formed by joining the backward trajectories from the four cell corners, to grid cell *H*. The region fluxed over the north edge of cell *H* consists of a triangle (*abc*) in the *NW* cell and a quadrilateral (two triangles, *acd* and *ade*) in the *N* cell.

Figure 2, reproduced from [9], shows all possible triangles that can contribute fluxes across the north edge of a grid cell. There are 20 triangles, which may be divided into five groups of four mutually exclusive triangles, as shown in Table 2. In this table, (x_1,y_1) and (x_2,y_2) are the Cartesian coordinates of the departure points relative to the NW and NE cell corners, respectively. The departure points are connected by a straight line that intersects the west edge at $(0, y_a)$ relative to the NW corner and intersects the east edge at $(0, y_b)$ relative to the NE corner. This line intersects the N edge at $(x_a, 0)$ relative to the NW corner and $(x_b, 0)$ relative to the NE corner.

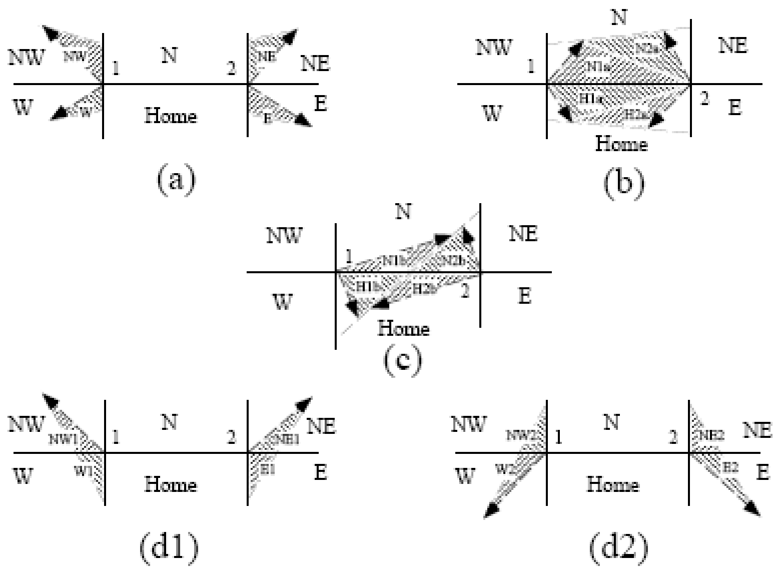

Figure 2: The 20 possible triangles that can contribute fluxes across the north edge of a grid cell.

From Cartesian geometry it can be shown that

$$
\begin{aligned}
y_a &= \frac{y_1\Delta x + (x_2 y_1 - x_1 y_2)}{\Delta x + x_2 - x_1}, \\
y_b &= \frac{y_2\Delta x + (x_2 y_1 - x_1 y_2)}{\Delta x + x_2 - x_1}, \\
x_a &= \frac{y_a \Delta x}{y_a - y_b}, \\
x_b &= \frac{y_b \Delta x}{y_a - y_b},
\end{aligned}
$$

where Δx represents the length of the N edge. The east cell triangles and selecting conditions are alike except for a rotation through 90 degrees.

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

Table 4: 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 $\tilde{y}_2 = y_b$.

This scheme was created for rectangular grids. Grid cells in PIPS 3.0 essentially lie on the surface of a sphere and must be projected onto a plane. Many projections are possible. The projection used in PIPS 3.0, illustrated in Figure 3, approximates spherical grid cells as quadrilaterals in the plane tangent to the sphere at a point inside the cell. The quadrilateral vertices are (*N*/2, *E*/2), (-*N*/2, *W*/2), (-*S*/2, -*W*/2), and (*S*/2, -*E*/2), where *N*, *S*, *E*, and *W* are the lengths of the cell edges on the spherical grid. The quadrilateral area, (*N* + *S*)(*E* + *W*)/4, is an apt approximation to the true spherical area. But cell edges in this projection are not orthogonal (i.e., they do not converge at right angles) as they do on the spherical grid. Therefore, when vectors are translated from cell corners to cell centers, the departure points in the cell-center coordinate system must be inside the grid cell contributing the flux. Otherwise, monotonicity may be violated, because van Leer limiting does not apply outside the grid cell.

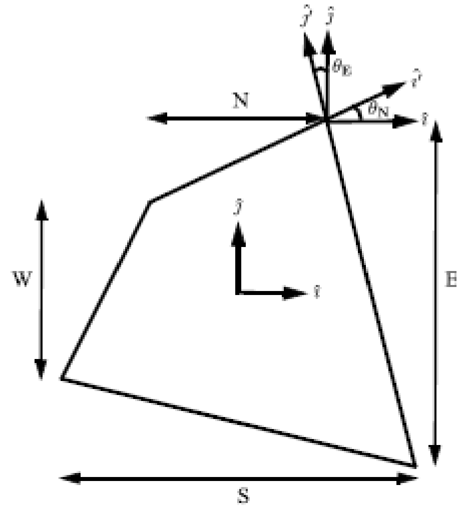

Figure 3: A grid cell on the surface of a sphere with unequal sides of length N, S, E, and W is estimated as a quadrilateral lying in the tangent plane at the cell center. The quadrilateral vertices are (N/2, E/2), (-N/2, W/2), (-S/2, -W/2), and (S/2, -E/2). The basis vectors ($\hat{i}', \hat{j}'$), located at the northeast cell corner, have been projected into the cell-center coordinate system and vary from the cell-center basis vectors ($\hat{i}, \hat{j}$). The angles θ_N and θ_E relating the two bases are defined in the text.

Figure 3 demonstrates the difficulty. At the cell center orthogonal basis vectors $\hat{i}$ and $\hat{j}$ that point to the midpoints of the cell edges are defined. Similarly, at each cell corner is defined a coordinate system whose basis vectors, $\hat{i}'$ and $\hat{j}'$ point along cell edges. The vectors $\hat{i}'$ and $\hat{j}'$ are orthogonal in the cell-corner reference frame, but not when projected into the reference frame of the bordering cell center. Because of this, a simple transformation is used to preserve monotonicity when vectors are translated from corners to centers. Consider a vector $(x'\hat{i}' + y'\hat{j}')$ in the cell-corner basis. An assumption is made that this vector has the same coordinates when $\hat{i}'$ and $\hat{j}'$ are non-orthogonal projections of the cell-corner basis vectors into the cell-center tangent plane, as in Figure 3. Thus a transformation follows from the ($\hat{i}', \hat{j}'$) basis to the ($\hat{i}, \hat{j}$) basis. In the cell-center coordinate system, $\hat{i}'$ is obtained by a rotation of $\hat{i}$ through an angle θ_N, where

$$\theta_N = \arctan\left(\frac{E-W}{2N}\right). \qquad (15)$$

Similarly, $\hat{j}'$ is obtained by a rotation of $\hat{j}$ through θ_E, where

$$\theta_E = \arctan\left(\frac{S-N}{2E}\right). \tag{16}$$

Vectors are transformed between basis sets using

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta_N & -\sin\theta_E \\ \sin\theta_N & \cos\theta_E \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}, \tag{17}$$

which may be verified by inspection, alternately setting $x' = 0$ and $y' = 0$. Similar transformations are used at the three other cell corners. These transformations guarantee that the grid cell in which a given departure point is positioned does not change under a modification in coordinate systems.

Most grids cells are fairly rectangular, unlike the distorted cell shown in Figure 3. On the 1° displaced-pole grid frequently used for PIPS 3.0 runs, the maximum angle in equations (15) and (16) is about 1°. Vector transformations could therefore be left out on most grids with little loss of accuracy. They have been retained, however, because they assure precise monotonicity at little added cost.

Another change should be noted in the scheme of [9] for locating triangles. In that paper, departure points are defined by projecting cell corner velocities directly backward. That is,

$$\mathbf{x}_\mathbf{D}' = -\mathbf{u}'\Delta t, \tag{18}$$

where $\mathbf{x}_D'$ is the location of the departure point relative to the cell corner. The primes represent vectors defined in the cell-corner basis. This approximation is accurate only for first-order. The accuracy may be raised to second-order by rectifying the velocity with a midpoint approximation before finding the departure point.

First, the midpoint of the backward trajectory is estimated using $\mathbf{x}_\mathbf{M}' = \mathbf{x}_\mathbf{D}'/2$, then an equation like equation (17) is used to transform $\mathbf{x}_\mathbf{M}'$ to the appropriate cell-center coordinate system. Next, a bilinear interpolation is employed to estimate the velocity at $\mathbf{x}_\mathbf{M}$. In a square 2 x 2 grid cell with corners $\tilde{\mathbf{x}}_\mathbf{1} = (-1,-1)$, $\tilde{\mathbf{x}}_\mathbf{2} = (1,-1)$, $\tilde{\mathbf{x}}_\mathbf{3} = (1,1)$, and $\tilde{\mathbf{x}}_\mathbf{4} = (-1,1)$, the values of any scalar field ϕ can be paired up at the cell corners with the following bilinear approximation:

$$\phi(\tilde{x},\tilde{y}) = \frac{1}{4}[\phi_1(\tilde{x}-1)(\tilde{y}-1) - \phi_2(\tilde{x}+1)(\tilde{y}-1) + \phi_3(\tilde{x}+1)(\tilde{y}+1) - \phi_4(\tilde{x}-1)(\tilde{y}+1)], \tag{19}$$

where ϕ_1, ϕ_2, ϕ_3, and ϕ_4 are the corner values. To use equation (19), $\mathbf{x_M}$ must be transformed from cell-center coordinates (x, y) into the simpler $(\tilde{x}, \tilde{y})$ coordinate system. Substituting *x* and *y* for ϕ in equation (19), we get

$$\begin{aligned}
x(\tilde{x},\tilde{y}) &= \frac{1}{4}[x_1(\tilde{x}-1)(\tilde{y}-1) - x_2(\tilde{x}+1)(\tilde{y}-1) + x_3(\tilde{x}+1)(\tilde{y}+1) - x_4(\tilde{x}-1)(\tilde{y}+1)],\\
y(\tilde{x},\tilde{y}) &= \frac{1}{4}[y_1(\tilde{x}-1)(\tilde{y}-1) - y_2(\tilde{x}+1)(\tilde{y}-1) + y_3(\tilde{x}+1)(\tilde{y}+1) - y_4(\tilde{x}-1)(\tilde{y}+1)].
\end{aligned}$$

Recalling that the corner coordinates are $\mathbf{x_1} = (-S/2, -W/2)$, $\mathbf{x_2} = (S/2, -E/2)$ $\mathbf{x_3} = (N/2, E/2)$, and $\mathbf{x_4} = (-N/2, W/2)$, expressions are derived for $\tilde{x}$ and $\tilde{y}$:

$$\tilde{x} = \frac{2x\Delta Y}{\Delta X \Delta Y + \delta X(2y - \tilde{x}\tilde{y}\delta Y)}, \tag{20}$$

$$\tilde{y} = \frac{2y\Delta X}{\Delta X \Delta Y + \delta Y(2x - \tilde{x}\tilde{y}\delta X)}, \tag{21}$$

where $\Delta X = (N+S)/2$, $\Delta Y = (E+W)/2$, $\delta X = (N-S)/2$, and $\delta Y = (E-W)/2$. These equations are nonlinear, since $\tilde{x}$ and $\tilde{y}$ emerge on the right-hand side, but are easily iterated to convergence. Given the $(\tilde{x}, \tilde{y})$ coordinates of the midpoint, we relate equation (19) to the components of $\mathbf{u}$ at the cell corners to estimate the velocity at the midpoint. The midpoint velocity is changed back to cell-corner coordinates using the inverse of equation (17), and then the corrected velocity in equation (18) is applied to find the departure point. With this correction, departure points for a velocity field varying linearly in space are nearly precise.

5.2.2.1.3 *Integrating Fluxes*

In the next step, the reconstructed fields are integrated over the departure triangles to find the total fluxes of area, volume, and energy across each cell edge. Area fluxes are simple to compute since the area is linear in *x* and *y*. Given a triangle with vertices $\mathbf{x_i} = (x_i, y_i)$, $i \in \{1,2,3\}$, the triangle area is

$$A_T = \frac{1}{2}\left|(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)\right|. \tag{22}$$

The integral I_1 of any linear function $f(\mathbf{r})$ over a triangle is given by

$$I_1 = A_T f(\mathbf{x_0}), \tag{23}$$

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

$$\mathbf{x}_0 = \frac{1}{3}\sum_{i=1}^{3}\mathbf{x}_i. \tag{24}$$

To compute the area flux, the area is evaluated at the midpoint,

$$a(\mathbf{x}_0) = a_c + a_x x_0 + a_y y_0, \tag{25}$$

and multiplied by A_T. By convention, northward and eastward fluxes are positive, while southward and westward fluxes are negative.

Equation (23) cannot be used for volume fluxes, since 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 necessitates function evaluations at three points,

$$I_2 = \frac{A_T}{3}\sum_{i=1}^{3} f\left(\mathbf{x}_i'\right), \tag{26}$$

where $\mathbf{x}_i' = (\mathbf{x}_0 + \mathbf{x}_i)/2$ are points lying halfway between the midpoint and the three vertices. The authors of [9] use this formula to calculate fluxes of the product *ρT*, which is equivalent to ice volume. Equation (26) 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 [42]:

$$I_3 = A_T\left[-\frac{9}{16} f(\mathbf{x}_0) + \frac{25}{48}\sum_{i=1}^{3} f(\mathbf{x}_i'')\right] \tag{27}$$

where $\mathbf{x}_\mathbf{i}'' = (3\mathbf{x}_0 + 2\mathbf{x}_i)/5$.

To evaluate functions at specific points, products of the form $a(x)h(x)$ and $a(x)h(x)q(x)$ must be computed, where each term in the product is the sum of a cell-center value and two displacement terms. This calculation may be sped up by storing some sums that are used repeatedly. First, weighted ice areas are computed at the four points of the integral formula (27):

$$a_0 = -\frac{9}{16}(a_c + a_x x_0 + a_y y_0),$$

$$a_i = \frac{25}{48}(a_c + a_x x_i + a_y y_i), i \in \{1,2,3\},$$

where the double primes have been dropped from the x_i. Defined next is

$$\begin{aligned}
\sigma_a &= \sum_{i=0}^{3} a_i, \\
\sigma_{ax} &= \sum_{i=0}^{3} a_i x_i, \\
\sigma_{ay} &= \sum_{i=0}^{3} a_i y_i,
\end{aligned}$$

which is used to compute the volume fluxes:

$$\sigma_{ah} = \sigma_a h_c + \sigma_{ax} h_x + \sigma_{ay} h_y .$$

Notice that σ_a, σ_{ax}, and σ_{ay} are used in three different flux computations: ice volume, snow volume, and area-weighted surface temperature. Defined next are

$$\begin{aligned}
\sigma_{axx} &= \sum_{i=0}^{3} a_i x_i^2, \\
\sigma_{axy} &= \sum_{i=0}^{3} a_i x_i y_i, \\
\sigma_{ayy} &= \sum_{i=0}^{3} a_i y_i^2, \\
\sigma_{axh} &= \sigma_{ax} h_c + \sigma_{axx} h_x + \sigma_{axy} h_y, \\
\sigma_{ayh} &= \sigma_{ay} h_c + \sigma_{axy} h_x + \sigma_{ayy} h_y .
\end{aligned}$$

The sums σ_{axh} and σ_{ayh} are calculated separately for ice and snow, whereas the first three sums are independent of the material being transported. Each sum is utilized repeatedly if there are multiple enthalpy layers. From these sums the energy fluxes are computed:

$$\sigma_{ahq} = \sigma_{ah} q_c + \sigma_{axh} q_x + \sigma_{ayh} q_y ,$$

thus finishing the flux integrals for a given triangle. To compute the total flux across a cell edge the contributions from each triangle are added.

5.2.2.1.4 Updating State Variables

Finally, these fluxes are used to compute new values of the state variables in every ice category and grid cell. The new fractional ice areas $a'_{in}(i,j)$ are given by

$$a'_{in}(i,j) = a_{in}(i,j) + \frac{F_{Ea}(i-1,j) - F_{Ea}(i,j) + F_{Na}(i,j-1) - F_{Na}(i,j)}{A(i,j)} \tag{28}$$

where $F_{Ea}(i,j)$ and $F_{Na}(i,j)$ are area fluxes across the east and north edges, respectively, of cell (i,j). $A(i,j)$ is the grid cell area. All fluxes added to one cell are subtracted from a neighboring cell; thus equation (28) conserves total ice area.

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

$$h' = \frac{\int_A ahdA}{\int_A adA},$$

$$q' = \frac{\int_A ahqdA}{\int_A ah\,dA},$$

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

5.2.2.2 Transport in Thickness Space

Now the equation for ice transport in thickness space due to thermodynamic growth and melt is solved by,

$$\frac{\partial g}{\partial t} + \frac{\partial}{\partial h}(fg) = 0, \tag{29}$$

which is obtained from equation (2) by ignoring the first and third terms on the right-hand side. The remapping method of [25] is used, in which thickness categories are represented as Lagrangian grid cells with boundaries that are projected ahead in time. The thickness distribution function *g* is estimated as a linear function of *h* in each displaced category and is then remapped onto the initial thickness categories. This method is numerically smooth (unlike schemes that treat $g(h)$ like a set of delta functions) and is not overly diffusive. It can be seen as a 1D simplification of the 2D incremental remapping scheme described above.

The first computation is 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 the subscript *i* that distinguishes ice from snow must be omitted.) A thermodynamic model (Section 5.2.2.5) is used to calculate 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 growth rate F_n is estimated 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 for $n = 1$ to N - 1 are given by

$$H_n^* = H_n + F_n \Delta t.$$

The boundaries cannot be displaced by more than one category to the left or right; in other words, $H_{n-1} < H_n^* < H_{n+1}$ is required. Without this requirement there would need to be a general remapping rather than an incremental remapping, at the cost of added complexity.

Next $g(h)$ is constructed in the displaced thickness categories. The ice areas in the displaced categories are $a_n^{m+1} = a_n^m$, because 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 brevity, define $H_L = H_{n-1}^*$ and $H_R = H_n^*$ and drop the time index *m* +1. A continuous function *g(h)* is built in to each category so that the total area and volume at time *m* +1 are a_n and v_n, respectively:

$$\int_{H_L}^{H_R} g\,dh = a_n, \tag{30}$$

$$\int_{H_L}^{H_R} hg\,dh = v_n. \tag{31}$$

The simplest polynomial that can satisfy both equations is a line. It is easy 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 (30) and (31) may be written as

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

where $\eta_R = H_R - H_L$ and $\eta_n = h_n - H_L$. The solution to these equations is

$$g_0=\frac{6a_n}{\eta_R^2}\left(\frac{2\eta_R}{3}-\eta_n\right), \quad (32)$$

$$g_1=\frac{12a_n}{\eta_R^3}\left(\eta_n-\frac{\eta_R}{2}\right). \quad (33)$$

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

$$g(0)=\frac{6a_n}{\eta_R^2}\left(\frac{2\eta_R}{3}-\eta_n\right)=g_0, \quad (34)$$

$$g(\eta_R)=\frac{6a_n}{\eta_R^2}\left(\eta_n-\frac{\eta_R}{3}\right). \quad (35)$$

Equation (34) implies that $g(0) < 0$ when $\eta_n > 2\eta_R/3$, i.e., when h_n sits in the right third of the thickness range (H_L,H_R). Similarly, equation (35) means that $g(\eta_R) < 0$ when $\eta_n < \eta_R/3$, i.e., when h_n is in the left third of the range. Because negative values of g are unphysical, a different solution is used when h_n lies outside the central third of the thickness range. If h_n is in the left third of the range, a cutoff thickness is identified as H_C =3h_n - 2H_L, and sets g =0 between H_C and H_R. Equations (32) and (33) are then valid with η_R redefined as H_C - H_L. And if h_n is in the right third of the range, define H_C =3h_n - 2H_R and set g =0 between H_L and H_C. In this case, equations (32) and (33) apply with $\eta_R = H_R - H_C$ and $\eta_n = h_n - H_C$.

Figure 4 demonstrates 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 surpasses the midpoint thickness. For h_n =0.2 and 0.8, g =0 over part of the range.

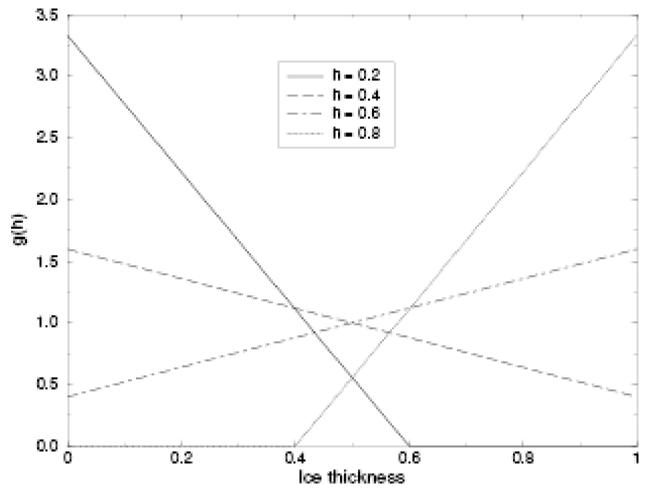

Figure 4: Linear approximation of the thickness distribution function *g(h)* for an ice category with left boundary H_L =0, right boundary H_R =1, fractional area a_n =1, and mean ice thickness h_n = 0.2, 0.4, 0.6, and 0.8.

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

$$\Delta a_n = \int_{H_n}^{H_n^*} g\,dh, \tag{36}$$

$$\Delta v_n = \int_{H_n}^{H_n^*} hg\,dh. \tag{37}$$

If $H_n^* < H_N$, ice area and volume are shifted from category *n* +1 to *n* using equations (36) and (37) with the limits of integration reversed. To evaluate the integrals, coordinates are changed 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 (32) and (33). This is how the new areas a_n and volumes v_n are acquired 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 be within the range (H_{n-1}, H_n). If g =0 in the area of a category that is remapped to a neighboring category, no ice is transferred.

Other conserved quantities are transported in proportion to the ice volume Δv_{in}. (Now using the subscripts *i* and *s* to distinguish ice from snow.) For example, the transferred snow volume is $\Delta v_{sn} = v_{sn}(\Delta v_{in}/v_{in})$, and the transferred ice energy in layer *k* is $\Delta e_{ink} = e_{ink}(\Delta v_{in}/v_{in})$.

The left and right boundaries of the domain necessitate special treatment. If ice is accumulating in open water at a rate F_0, the left boundary H_0 is shifted right by $F_0\Delta t$ before g is constructed in category 1, then reset to zero after the remapping is finished. Then new ice is added to the grid cell, conserving area, volume, and energy. If ice cannot grow in open water (if the sea water is too warm or the net surface energy flux is downward), H_0 is fixed at zero, and the growth rate at the left boundary is approximated as $F_0 = f_1$. If $F_0 < 0$, the area of ice thinner than $\Delta h_0 = -F_0\Delta t$ is added to the open water area a_0, allowing the ice and snow volume and energy to remain unchanged. 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, set $H_N = 3h_N - 2H_{N-1}$, which guarantees 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 big, the thickness remapping scheme will not function. Instead, the thickness categories in that grid cell are treated as delta functions following [6], and categories outside their set 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 rarely needed, if ever.

5.2.2.3 Mechanical Redistribution

The last term on the right-hand side of equation (2) is ψ, which defines the redistribution of ice in thickness space due to ridging and other mechanical processes. The mechanical redistribution scheme in PIPS 3.0 is based on [43], [35], [16], [27] and [13]. This scheme transforms thinner ice to thicker ice and is applied after horizontal transport. When the ice is converging, enough ice ridges to guarantee that the ice area does not surpass the grid cell area.

First specify the participation function: the thickness distribution $a_P(h) = b(h)g(h)$ of the ice participating in ridging. ("Ridging" is used here as shorthand for all modes of mechanical redistribution.) The weighting function $b(h)$ prefers ridging of thin ice and closing of open water to ridging of thicker ice. There are two options for the form of $b(h)$. If $\mathtt{krdg}_{\mathtt{partic}} = 0$ in the namelist, then following [43], we set

$$b(h) = \begin{cases} \frac{2}{G^*}\left(1 - \frac{G(h)}{G^*}\right) & if G(h) < G^* \\ 0 & otherwise \end{cases} \tag{38}$$

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, the mean value of a_P in category *n* is obtained by

$$a_{Pn} = \frac{2}{G^*}(G_n - G_{n-1})\left(1 - \frac{G_{n-1} + G_n}{2G^*}\right), \tag{39}$$

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 (39) applies to categories with $G_n < G^*$. If $G_{n-1} < G^* < G_n$, equation (39) 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 will ridge, because "ridging" simply reduces the area of open water. As in [43], G^* is set to =0.15.

If the spatial resolution is too fine for a given time step Δt, the weighting function (38) can support numerical instability. For $\Delta t = 1\,\text{hour}$, resolutions finer than $\Delta x \sim 10\,\text{km}$ are usually unstable. The instability comes from feedback between the ridging scheme and the dynamics through the ice strength. If the strength changes considerably on time scales less than Δt, the EVP solution becomes inaccurate and sometimes oscillatory. Consequently, the fields of ice area, thickness, velocity, strength, divergence, and shear can become noisy and unphysical.

A more stable weighting function was suggested by [27]:

$$b(h) = \frac{\exp[-G(h)/a^*]}{a^*[1 - \exp(-1/a^*)]} \tag{40}$$

When integrated between category boundaries, equation (40) implies

$$a_{Pn} = \frac{\exp(-G_n/a^*) - \exp(-G_{n-1}/a^*)}{1 - \exp(-1/a^*)} \tag{41}$$

This weighting function is applied if $\mathtt{krdg}_{\mathtt{partic}} = 1$ in the namelist. From equation (40), the mean value of G for ice involved in ridging is a^*, as compared to $G^*/3$ for equation (38). For standard ice thickness distributions, setting $a^* = 0.05$ with $\mathtt{krdg}_{\mathtt{partic}} = 1$ generates participation fractions similar to those given by $G^* = 0.15$ with $\mathtt{krdg}_{\mathtt{partic}} = 0$. See [27] for a detailed comparison of these two participation functions.

Thin ice is converted to thick ridged ice in a way that diminishes the total ice area while conserving ice volume and energy. There are two namelist options for redistributing ice among thickness categories. If $\mathtt{krdg}_{\mathtt{redist}} = 0$, ridging ice of thickness h_n creates ridges whose area is distributed uniformly between $H_{\min} = 2h_n$ and $H_{\max} = 2\sqrt{H^* h_n}$, as in [16]. The default value of H* is 25 m. Observations suggest that $H^* = 50$ m gives a better fit to first-year ridges [4], although the lower value may be appropriate for multiyear ridges [13]. 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 matures 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 allocated amid categories in the thickness range $(H_{\min}, H_{\max})$. The fraction of the new ridge area going to category *m* is

$$f_m^{\text{area}} = \frac{H_R - H_L}{H_{\max} - H_{\min}}, \tag{42}$$

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

$$f_m^{\text{vol}} = \frac{(H_R)^2 - (H_L)^2}{(H_{\max})^2 - (H_{\min})^2}. \tag{43}$$

This uniform redistribution function has a tendency to not produce enough ice in the 3--5 m range and too much ice thicker than 10 m [4]. Observational data show that the ITD of ridges is better approximated by a negative exponential. Setting $\texttt{krdg}_{\texttt{redist}} = 1$ gives ridges with an exponential ITD [27]:

$$g_R(h) \propto \exp[-(h - H_{\min})/\lambda] \tag{44}$$

for $h >= H_{\min}$, with $g_R(h) = 0$ for $h < H_{\min}$. Here, the variable λ is an empirical *e*-folding scale and $H_{\min} = 2h_n$ (where h_n represents the thickness of ridging ice). It is assumed that $\lambda = \mu h_n^{1/2}$, where μ is a tunable parameter with units of $\text{m}^{1/2}$. Therefore the mean ridge thickness increases in proportion to $h_n^{1/2}$, as in [16]. The default value $\mu = 4.0\ \text{m}^{1/2}$ assigns λ in the range 1--4 m for most ridged ice. Ice strengths with $\mu = 4.0\ \text{m}^{1/2}$ and $\texttt{krdg}_{\texttt{redist}} = 1$ are nearly comparable to the strengths with $H^* = 50$ and $\texttt{krdg}_{\texttt{redist}} = 0$.

From equation (44) it can be shown that the fractional area going to category m as a result of ridging is

$$f_m^{\text{area}} = \exp[-(H_{m-1} - H_{\min})/\lambda] - \exp[-(H_m - H_{\min})/\lambda]. \tag{45}$$

The fractional volume available to category m is

$$f_m^{\text{vol}} = \frac{(H_{m-1}+\lambda)\exp[-(H_{m-1}-H_{\text{min}})/\lambda] - (H_m+\lambda)\exp[-(H_m-H_{\text{min}})/\lambda]}{H_{min}+\lambda}. \tag{46}$$

Equations (45) and (46) replace equations (42) and (43) when $\texttt{krdg}_{\texttt{redist}} = 1$.

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

The net area removed through ridging and closing is a function of the strain rates. Let R_{net} equal the net rate of area loss for the ice pack (i.e., the rate of open water area closing, in addition to the net rate of ice area loss due to ridging). Following [13], R_{net} is given by

$$R_{\text{net}} = \frac{C_s}{2}(\Delta - |D_D|) - \min(D_D, 0), \tag{47}$$

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

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

$$R_{\text{net}} = \left[a_{P0} + \sum_{n=1}^{N} a_{Pn}\left(1 - 1k_n\right) \right] R_{\text{tot}}. \tag{48}$$

Given R_{net} from equation (47), equation (48) is used to compute R_{tot}. Next the area ridged in category n is shown 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 $a_{rn}h_n$ represents the volume of new ridges (because volume is conserved during ridging). The ridging ice from category n is removed and we use equations (42) and (43) or equations (45) and (46) to redistribute the ice amid thicker categories.

In some instances the ridging rate in a given thickness category n may be substantial enough to ridge the total area in that category during a time interval less than Δt. In this instance R_{tot} is reduced to the value that accurately ridges an area a_n during Δt. Following 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 $a_i = 1$.

The ice strength P may be computed in either of two ways. If the namelist parameter $\texttt{krdg}_{\texttt{redist}} = 0$, the strength is given by [15]:

$$P = P^{*} h \exp[-C(1-a_i)], \tag{49}$$

where P* = 27, 500 N/m and C = 20 are empirical constants and h is the mean ice thickness. Alternatively, setting $\mathtt{krdg}_{\mathtt{redist}} = 1$ provides an ice strength closely related to the ridging scheme. Following [35], the strength is known to be proportional to the change in ice potential energy ΔE_p per unit area of compressive deformation. Given the uniform ridge ITDs ($\mathtt{krdg}_{\mathtt{redist}} = 0$), notice that

$$P = C_f C_p \beta \sum_{n=1}^{N_C} \left[-a_{Pn} h_n^2 + \frac{a_{Pn}}{k_n} \left(\frac{(H_n^{\max})^3 - (H_n^{\min})^3}{3(H_n^{\max} - H_n^{\min})} \right) \right], \tag{50}$$

where $C_P = (g/2)(\rho_i/\rho_w)(\rho_w - \rho_i)$, $\beta = R_{\text{tot}}/R_{\text{net}} > 1$ from equation (48), and C_f represents an empirical parameter that accounts for frictional energy dissipation. Following [13], we set C_f = 17. The primary term in the summation is the potential energy of ridging ice, and the second, greater term is the potential energy of the resulting ridges. The factor of β is included since a_{Pn} is normalized with respect to the total area of ice ridging, not the net area removed. Remember that more than one unit area of ice must be ridged to reduce the net ice area by one unit. For exponential ridge ITDs ($\mathtt{krdg}_{\mathtt{redist}} = 1$), the ridge potential energy is adapted:

$$P = C_f C_p \beta \sum_{n=1}^{N_C} \left[-a_{Pn} h_n^2 + \frac{a_{Pn}}{k_n} \left(H_{\min}^2 + 2H_{\min}\lambda + 2\lambda^2 \right) \right] \tag{51}$$

The energy-based ice strength given by equations (50) or (51) is more physically realistic than the strength produced by equation (49). But the use of equation (49) is less likely to permit numerical instability at a given resolution and time step. See [27] for more details.

5.2.2.4 Dynamics

The elastic-viscous-plastic (EVP) model represents an alteration of the standard viscous-plastic (VP) model for sea ice dynamics by [15]. It reduces to the VP model at temporal scales associated with the wind forcing, while at shorter time scales the modification process takes place by a numerically more efficient elastic wave mechanism. While retaining the basic physics, this elastic wave version leads to a fully explicit numerical scheme that greatly enhances the model's computational efficiency.

The EVP sea ice dynamics model is well documented in [18], [17], [19] and [20]. Simulation results and performance of the EVP model have been evaluated against the VP model in realistic simulations of the Arctic [21]. The equations are summarized below but it is recommended that the reader refer back to the above references for details. The numerical implementation in this code release is that of [19], [20].

The force balance per unit area in the ice pack is set by a two-dimensional momentum equation [15], obtained by integrating the 3D equation through the thickness of the ice in the vertical direction:

$$m\frac{\partial u}{\partial t} = \nabla \cdot \sigma + \vec{\tau}_a + \vec{\tau}_w - \hat{k} \times mfu - mg\nabla H_\circ, \tag{52}$$

where *m* is the combined mass of ice and snow per unit area and $\vec{\tau}_a$ and $\vec{\tau}_w$ represent wind and ocean stresses, respectively. The strength of the ice is given by the internal stress tensor σ_{ij}, and the other two terms on the right side are stresses due to Coriolis effects and the sea surface slope. The parameterization for the wind and ice-ocean stress terms must have the ice concentration as a multiplicative factor to be consistent with the formal theory of free drift in low ice concentration areas. A thorough explanation of the issue and its continuum solution is given in [20] and [8].

For simplicity the stress tensor σ is formulated in terms of $\sigma_1 = \sigma_{11} + \sigma_{22}$, $\sigma_2 = \sigma_{11} - \sigma_{22}$ and the divergence, D_D. The stress tensor is also formulated in terms of the horizontal tension and shearing strain rates, D_T and D_S respectively. The internal stress tensor is determined from a standardized version of the VP constitutive law,

$$\frac{1}{E}\frac{\partial \sigma_1}{\partial t} + \frac{\sigma_1}{2\zeta} + \frac{P}{2\zeta} = D_D, \tag{53}$$

$$\frac{1}{E}\frac{\partial \sigma_2}{\partial t} + \frac{\sigma_2}{2\eta} = D_T, \tag{54}$$

$$\frac{1}{E}\frac{\partial \sigma_{12}}{\partial t} + \frac{\sigma_{12}}{2\eta} = \frac{1}{2}D_S, \tag{55}$$

where

$$
\begin{aligned}
D_D &= \dot{\varepsilon}_{11} + \dot{\varepsilon}_{22}, && (56)\\
D_T &= \dot{\varepsilon}_{11} - \dot{\varepsilon}_{22}, && (57)\\
D_S &= 2\dot{\varepsilon}_{12}, && (58)\\
\dot{\varepsilon}_{ij} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j{}_j} + \frac{\partial u_j}{\partial x_i{}_i}\right), &&\\
\zeta &= \frac{P}{2\Delta}, &&\\
\eta &= \frac{P}{2\Delta e^2}, &&\\
\Delta &= \left[D_D^2 + 1e^2\left(D_T^2 + D_S^2\right)\right]^{1/2}, &&
\end{aligned}
$$

and P is a function of the ice thickness and concentration, explained in Section 5.2.2.3. The dynamics component utilizes a “replacement pressure” (see [14], for example), which functions to prevent residual ice motion due to spatial variations of P when the rates of strain are exactly zero.

Many adjustments have been made to the EVP model since its original release. In the previous version, the viscosities were held fixed while the stress and momentum equations were subcycled with the smaller time step `dte`. The reason for applying the EVP model in this way was to reproduce the results of the original VP model as closely as possible. When solved with time steps of several hours or more, the VP model goes through a linearization error associated with the viscosities, which are lagged over the time step [17]. This led to principal stress states that were widely spread outside the elliptical yield curve in both models [21]. This problem has been addressed by updating the viscosities during the subcycling, so that the complete dynamics component is subcycled within the time step. Taken alone, this modification would require an increased number of operations to compute the viscosities.

Nonetheless, the new dynamics component is roughly as efficient as the earlier version due to a change in the definition of the elastic parameter *E*. *E* is now described in terms of a damping timescale *T* for elastic waves, $\Delta t_e < T < \Delta t$, as

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

where $T = E_\circ \Delta t$ and $E_\circ$ (eyc) is a tunable parameter less than one, as before. The stress equations (53-55) 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 side are constant with the exception of *P*, which changes only on the longer time step Δ*t*. This alteration compensates for the diminished efficiency of including the viscosity terms in the subcycling. (Note that the viscosities do not appear explicitly.) Choices of the parameters used to describe *E*, *T* and Δt_e are discussed in the PIPS 3.0 User's Manual [2].

A different discretization of the stress terms $\partial\sigma_{ij}/\partial x_j$ in the momentum equation is now used, which allows the discrete equations to be derived from the continuous equations written in curvilinear coordinates. In this way, metric terms associated with the curvature of the grid were integrated into the discretization explicitly. No longer in use is the “triangle discretization,” in which the strain rates and stresses were constant over each of four subtriangles in each grid cell, and instead a bilinear approximation for the velocities and stresses is used. Details pertaining to the spatial discretization are given in [19].

The momentum equation is discretized in time as follows. First, to clarify, the two components of equation (52) are

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

In the code, $vrel=a_ic_w\rho_w\left|U_w-u^k\right|$, where *k* denotes the subcycling step. The following equations show the time discretization and illustrate some of the other variables used in the code.

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

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

`vrel·waterx(y) = taux(y)`. This system of equations is solved analytically for u^{k+1} and v^{k+1}. When the subcycling is complete for each (thermodynamic) time step, the ice-ocean stress must be constructed from `taux(y)` and the terms containing `vrel` on the left side of the equations. This is done in subroutine *evp_finish*.

5.2.2.5 Thermodynamics

The thermodynamic sea ice model is based on [31] and [7], and is described more fully in [24]. For each thickness category the model calculates 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 reproduce the effect of brine pocket melting and freezing.

Each thickness category *n* in each grid cell is handled as a horizontally uniform column with ice thickness $h_{in} = v_{in}/a_{in}$ and snow thickness $h_{sn} = v_{sn}/a_{in}$. (Henceforth the category index *n* is omitted.) Each column is divided into N_i ice layers of thickness $\Delta h_i = h_i/N_i$ and, if snow is present, a single snow layer. (Allowing for multiple snow layers is possible in future versions of PIPS 3.0.) The surface temperature (i.e., the temperature of ice or snow at the interface with the atmosphere) is T_{sf}, which cannot exceed 0°C. The temperature at the midpoint of the snow layer is T_s, and the midpoint ice layer temperatures are T_{ik}, where *k* ranges from 1 to N_i. The temperature at the ice bottom is held at T_f, the freezing temperature of the ocean mixed layer. All temperatures are in degrees Celsius unless otherwise noted.

The vertical salinity profile is set and is unchanging in time. The snow is assumed to be fresh, and the midpoint salinity S_{ik} in each ice layer is given by

$$S_{ik} = \frac{1}{2} S_{\max}[1 - \cos(\pi z^{(\frac{a}{z+b})})], \tag{59}$$

where $z \equiv (k - 1/2)/N_i$, $S_{\max} = 3.2$ psu. The variables a =0.407 and b =0.573 are determined from a least-squares fit to the salinity profile seen in multiyear sea ice by [36]. 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 [31]. Equation (59) is quite 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, though, which has a more vertically uniform salinity since it has not yet drained. However, the effects of salinity on heat capacity are negligible for temperatures well below freezing, so the salinity error does not lead to significant temperature errors.

Each ice layer has an enthalpy q_{ik}, defined as the negative of the energy necessary to melt a unit volume of ice and raise its temperature to 0°C. Due to internal melting and freezing in brine pockets, the enthalpy of the ice depends on the brine pocket volume and is a function of temperature and salinity. Since the salinity is prescribed, there is a one-to-one relationship between temperature and enthalpy. Snow enthalpy q_s is also defined, which depends on temperature alone. Enthalpy equations are derived in Section 5.2.2.5.3.

Given surface forcing at the atmosphere-ice and ice-ocean interfaces in addition to the ice and snow thicknesses and temperatures/enthalpies at time *m*, the thermodynamic model advances these quantities to time *m* +1. The computation proceeds in two steps. First to be solved is a set of equations for the new temperatures, as discussed in Section 5.2.2.5.2. Next is the calculation of 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 5.2.2.5.3. We begin by defining the surface forcing parameterizations, which are closely related to the ice and snow surface temperatures.

5.2.2.5.1 Thermodynamic Surface Forcing

The net energy flux from atmosphere to 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 represents the shortwave albedo, and i_0 is the fraction of absorbed shortwave flux that penetrates the ice.

The albedo depends on the thickness and temperature of ice and snow and on the spectral distribution of the incoming solar radiation. Albedo parameters have been selected to fit observations from the Surface Heat Budget of the Arctic Ocean (SHEBA) field experiment. For $T_{sf} < -1$°C and $h_i > 0.5$m, the bare ice albedo is 0.78 for visible wavelengths (< 700 nm) and 0.36 for near IR wavelengths (> 700 nm). As h_i decreases from 0.5 m to zero, the ice albedo decreases efficiently (through use of an arctangent function) to the ocean albedo, 0.06. The ice albedo in both spectral bands drops by 0.075 as T_{sf} rises from -1°C to 0°C. The albedo of cold snow (T_{sf} < -1°C) is 0.98 for visible wavelengths and 0.70 for near IR wavelengths. The visible snow albedo falls by 0.10 and the near IR albedo by 0.15 as T_{sf} increases from -1°C to 0°C. The total albedo is an area-weighted average of the ice and snow albedos, where the fractional snow-covered area is

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

and $h_{snowpatch}$ =0.02 m. The envelope of albedo values is shown in Figure 5.

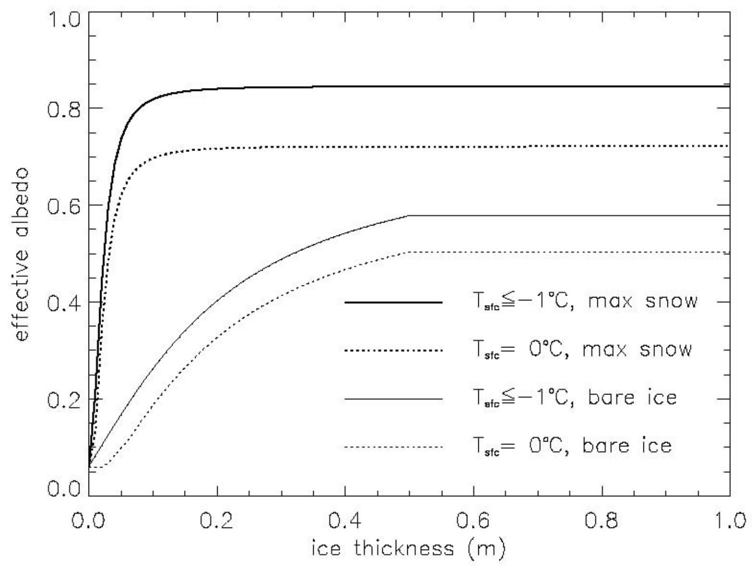

Figure 5: Albedo as a function of ice thickness and temperature for the two extrema in snow depth. Maximum snow depth is calculated based on Archimedes' Principle for the given ice thickness. These curves symbolize the envelope of potential albedo values.

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

$$I(z) = I_0 \exp(-\kappa_i z), \tag{60}$$

where $I(z)$ is the shortwave flux that extends to depth z beneath the surface without being absorbed, and κ_i is the bulk extinction coefficient for solar radiation in ice, set at 1.4 m^{-1} for visible wavelengths [12]. A fraction exp $(-\kappa_i h_i)$ of the penetrating solar radiation passes through the ice to the ocean ($F_{sw\Downarrow}$). Although incoming shortwave and longwave radiation are received from the atmosphere, outgoing long-wave radiation and the turbulent heat fluxes are derived quantities. Outgoing longwave assumes the standard blackbody form, $F_{L\uparrow} = \varepsilon\sigma\left(T_{sf}^K\right)^4$, where $\varepsilon = 0.95$ is the emissivity of snow or ice, σ is the Stefan-Boltzmann constant and T_{sf}^K is the surface temperature in Kelvin.

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

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

C_s and C_l (below) are nonlinear turbulent heat transfer coefficients described in Section 5.2.1.1. Likewise, the latent heat flux is proportional to the difference between Q_a and the surface saturation specific humidity Q_{sf}:

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

where q_1 =1.16378 x 10^7 kg/m^3 , q_2 = 5897.8K, 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 provided by [29]:

$$F_{bot} = -\rho_w c_w c_h u_*(T_w - T_f), \tag{61}$$

where ρ_w is the density of seawater, c_w is the specific heat of seawater, c_h =0.006 represents a heat transfer coefficient, $u_* = \sqrt{|\vec{\tau}_w|/\rho_w}$ is the friction velocity, and T_w is the sea surface temperature. The minimum value of u_* depends on if the model is run coupled; lack of currents in uncoupled runs means there was insufficient heat available to melt ice in the standard formulation. In this release we have $u_{*\min} = 5\times10^{-3}$ for coupled runs and 5×10^{-2} for uncoupled runs.

5.2.2.5.2 New Temperatures

Given the temperatures T_{sf}^m , T_s^m and T_{ik}^m at time *m*, we solve a series of finite-difference equations to obtain the new temperatures at time *m* +1. Each temperature is coupled to the temperatures of the layers directly 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. Therefore, we have a set of equations of the form

$$\mathbf{A}\mathrm{x} = \mathbf{b}, \tag{62}$$

where **A** is a tridiagonal matrix, x is a column vector whose components are the unknown new temperatures, and **b** represents another column vector. Given **A** and **b**, x can be computed with a standard tridiagonal solver.

There are four general cases:
(1) $T_{sf} < 0$°C, snow present;
(2) $T_{sf} = 0$°C, snow present;
(3) $T_{sf} < 0$°C, snow absent; and

(4) T_{sf}=0°C, snow absent.

Case 1 has one equation (the top row of the matrix) for the new surface temperature, one equation (the second row) for the new snow temperature, and N_i equations (the remaining rows) for the new ice temperatures. For cases 2 and 4 the equation for the surface temperature is omitted, which is held at 0°C, and for cases 3 and 4 the snow temperature equation is omitted.

The rate of temperature change in the ice interior is given by [31]:

$$\rho_i c_i \frac{\partial T_i}{\partial t} = \frac{\partial}{\partial z}\left(k_i \frac{\partial T_i}{\partial z}\right) - \frac{\partial}{\partial z}[I_0 \exp(-\kappa_i z)], \tag{63}$$

where ρ_i = 917 kg/m^3 represents 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_0 is the flux of penetrating solar radiation, attenuated with extinction coefficient κ_i (see previous section), and z represents the vertical coordinate, set to be positive downward with z =0 at the top surface. The specific heat of sea ice is given to an excellent approximation by [33],

$$c_i(T,S) = c_0 + \frac{L_0 \mu S}{T^2}, \tag{64}$$

where c_0 = 2106 J/kg/deg is the specific heat of fresh ice at 0°C, L_0 =3.34 x 10^5 J/kg is the latent heat of fusion of fresh ice at 0°C, and μ =0.054 deg/psu represents the ratio between freezing temperature and the salinity of brine. Following [44], the thermal conductivity is shown by

$$k_i(T,S) = k_0 + \frac{\beta S}{T}, \tag{65}$$

where k_0 =2.03 W/m/deg is the conductivity of fresh ice and β =0.13 W/m/psu is an empirical constant. The analogous equation for the temperature change in snow is

$$\rho_s c_s \frac{\partial T_s}{\partial t} = \frac{\partial}{\partial z}\left(k_s \frac{\partial T_s}{\partial z}\right), \tag{66}$$

where ρ_s = 330 kg/m^3 is the snow density (also assumed uniform), c_s = c_0 is the specific heat of snow, and k_s = 0.30 W/m/deg denotes the thermal conductivity of snow. Penetrating solar radiation is neglected in equation (66) because the majority of the incoming sunlight is absorbed near the top surface when snow exists.

Now Equations (63) and (66) are converted to finite-difference form. The resulting equations are second-order accurate in space (except maybe at material boundaries) and first-order accurate in time. Before writing the equations in full the finite-difference expressions are provided for some of the terms.

First consider the terms on the left-hand side of equations (63) and (66). The time derivatives are written as

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

where T is the temperature of either ice or snow. The specific heat of ice layer *k* is approximated as

$$c_{ik} = c_0 + \frac{L_0 \mu S_{ik}}{T_{ik}^{m} T_{ik}^{m+1}}, \tag{67}$$

which guarantees that energy is conserved during a change in temperature. This can be shown by using equation (64) 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 (67). Unfortunately, the specific heat is a nonlinear function of T_{ik}^{m+1}, the unknown new temperature. A set of linear equations is retained, however, by initially guessing $T_{ik}^{m+1} = T_{ik}^{m}$ and then iterating the solution, updating T_{ik}^{m+1} in equation (67) with each iteration until the solution converges.

Next to consider is the heat diffusion term, the first term on the right side of equation (63). In the ice interior (layers 2 to N_i - 1) this term is discretized as

$$\frac{\partial}{\partial z}\left(k_i \frac{\partial T_i}{\partial z}\right) = \frac{1}{\Delta h_i}\left[\frac{k_{i,k}(T_{i,k-1}^{m+1} - T_{ik}^{m+1})}{\Delta h_i} - \frac{k_{i,k+1}(T_{ik}^{m+1} - T_{i,k+1}^{m+1})}{\Delta h_i}\right], \tag{68}$$

where k_{ik} represents the thermal conductivity at the upper boundary of layer *k*. The approximation in equation (68) is spatially centered and second-order accurate. Similar expressions can be written for heat diffusion in the top and bottom ice layers and the snow layer, as shown below. Note that the conduction terms are handled implicitly; that is, they depend on the temperatures at the new time *m*+1. The resulting scheme is first-order accurate in time and unconditionally stable.

Using equation (65), k_{ik} is approximated in the ice interior (at the upper boundary of layers 2 to N_i) as

$$k_{ik} = k_0 + \frac{\beta(S_{i,k-1} + S_{ik})}{T_{i,k-1}^{m} + T_{ik}^{m}}.$$

Because the conductivity does not depend as much on temperature as does the specific heat, k_{ik} is defined in terms of the ice temperatures at time m. Thus the conductivity does not need to be updated with each iteration. At the bottom surface there is

$$k_{i,N_i+1} = k_0 + \frac{\beta S_{\max}}{T_f}.$$

The conductivity at the top ice surface, k_{i1}, depends on whether snow exists. If there is no snow, we set

$$k_{i1} = k_0 + \frac{\beta S_{i1}}{T_{i1}^m}.$$

(Defining k_{i1} in terms of T_{sf} is avoided since then it would be undefined for T_{sf} =0.) If snow is present, a continuous heat flux across the ice-snow interface is assumed:

$$k_{i1} \frac{T_{i1}^m - T_{int}^m}{\Delta h_i/2} = k_s \frac{T_{int}^m - T_s^m}{h_s/2},$$

where T_{int} is the interface temperature. Solving for T_{int}^m, it is evident that this heat flux is equivalent to

$$k_{int} \frac{T_{i1}^m - T_s^m}{(\Delta h_i + h_s)/2},$$

where k_{int} , the equivalent conductivity at the interface, is defined as

$$k_{int} = \frac{k_{i1} k_s (\Delta h_i + h_s)}{h_s k_{i1} + \Delta h_i k_s}.$$

Finally, the second term on the right in equation (63) is taken into account. From equation (60), the fraction of the penetrating solar radiation I_0 transmitted through layer k without being absorbed is

$$\tau_k = \exp(-\kappa_i k \Delta h_i).$$

Thus the flux absorbed in layer k is

$$Q_k = I_0 (\tau_{k-1} - \tau_k).$$

The flux absorbed per unit ice thickness is $Q_k/\Delta h_i$, the desired finite-difference approximation to

$$-\frac{\partial}{\partial z}[I_0 \exp(-\kappa_i z)].$$

Now a system of equations for the new temperatures is constructed. (The reader uninterested in algebraic details may want to go to the next section.) Beginning at the surface and working down for case 1 ($T_{sf} < 0$◦C and snow present), we require

$$F_0 = F_{ct}, \tag{69}$$

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}, there is 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 rewrite equation (69) as

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

where $K_s \equiv 2k_s/h_s$. Rearranging terms, we get

$$\left[\left(\frac{dF_0}{dT_{sf}}\right)^* - K_s\right] T_{sf}^{m+1} + K_s T_s^{m+1} = \left(\frac{dF_0}{dT_{sf}}\right)^* T_{sf}^* - F_0^*, \tag{70}$$

the first equation in the set of equations (62).

Continuing with case 1, we write the equation for the change in T_s:

$$\rho_s c_s \frac{(T_s^{m+1} - T_s^m)}{\Delta t} = \frac{1}{h_s}[K_s (T_{sf}^{m+1} - T_s^{m+1}) - K_{int}(T_s^{m+1} - T_{i1}^{m+1})] \tag{71}$$

where $K_{int} \equiv 2k_{int}/(\Delta h_i + h_s)$. In tridiagonal matrix form equation (71) becomes

$$-\eta_s K_s T_{sf}^{m+1} + [1 + \eta_s (K_{int} + K_s)] T_s^{m+1} - \eta_s K_{int} T_{i1}^{m+1} = T_s^m, \tag{72}$$

where $\eta_s \equiv \Delta t/(\rho_s c_s h_s)$.

The ice equations for the top layer, the interior layers (2 to N_i - 1), and the bottom layer, respectively, are

$$\begin{aligned}
\rho_i c_{i1}\frac{(T_{i1}^{m+1}-T_{i1}^{m})}{\Delta t} &= \frac{1}{\Delta h_i}[K_{int}(T_s^{m+1}-T_{i1}^{m+1})-K_{i2}(T_{i1}^{m+1}-T_{i2}^{m+1})+Q_1],\\
\rho_i c_{ik}\frac{(T_{ik}^{m+1}-T_{ik}^{m})}{\Delta t} &= \frac{1}{\Delta h_i}[K_{ik}(T_{i,k-1}^{m+1}-T_{ik}^{m+1})-K_{i,k+1}(T_{ik}^{m+1}-T_{i,k+1}^{m+1})+Q_k],\\
\rho_i c_{iN}\frac{(T_{iN}^{m+1}-T_{iN}^{m})}{\Delta t} &= \frac{1}{\Delta h_i}\{K_{iN}(T_{i,N-1}^{m+1}-T_{iN}^{m+1})-\\
&\qquad K_{i,N+1}[\gamma_1(T_{iN}^{m+1}-T_f)+\gamma_2(T_{i,N-1}^{m+1}-T_f)]+Q_1\},
\end{aligned}$$

where $K_{ik} \equiv k_{ik}/\Delta h_i$ and $N \equiv N_i$. The coefficients γ_1 =3 and γ_2 = -1/3 provide one-sided second-order spatial accuracy at the bottom surface; they are derived from a Taylor series expansion of *dT/dz* at z = h_i. Rearranging terms, we have

$$-\eta_{i1}K_{int}T_s^{m+1}+[1+\eta_{i1}(K_{int}+K_{i2})]T_{i1}^{m+1}-\eta_{i1}K_{i2}T_{i2}^{m+1}=T_{i1}^{m}+\eta_{i1}Q_1, \tag{73}$$

$$-\eta_{ik}K_{ik}T_{i,k-1}^{m+1}+[1+\eta_{ik}(K_{ik}+K_{i,k+1})]T_{ik}^{m+1}-\eta_{ik}K_{i,k+1}T_{i,k+1}^{m+1}=T_{ik}^{m}+\eta_{ik}Q_k, \tag{74}$$

$$\begin{aligned}
-\eta_{iN}(K_{i,N}-\gamma_2K_{i,N+1})T_{i,N-1}^{m+1}+[1+\eta_{iN}(K_{iN}+\gamma_1K_{i,N+1})]T_{iN}^{m+1} &=\\
\eta_{iN}K_{i,N+1}(\gamma_1+\gamma_2)T_f+T_{iN}^{m}+\eta_{iN}Q_N&,
\end{aligned} \tag{75}$$

where $\eta_{ik} \equiv \Delta t/(\rho_i c_{ik}\Delta h_i)$.

Next consider case 2 (T_{sf} =0°C and snow present). Since T_{sf} is fixed, there is no surface flux equation. The new snow temperature is given by equation (71), but with the unknown T_{sf}^{m+1} replaced by T_{sf}=0°C. In matrix form we have

$$[1+\eta_s(K_{int}+K_s)]T_s^{m+1}-\eta_sK_{int}T_{i1}^{m+1}=\eta_sK_sT_{sf}+T_s^{m}. \tag{76}$$

The ice equations for case 2 are the same as for case 1: (73), (74), and (75).

For case 3 (T_{sf}< 0°C and snow absent) the surface temperature equation is like equation (69), but we use a second-order accurate expression for *dT/dz* at z =0:

$$F_0^* + \left(\frac{dF_0}{dT_{sf}}\right)^* (T_{sf}^{m+1} - T_{sf}^*) = K_{i1}[\gamma_1(T_{sf}^{m+1} - T_{i1}^{m+1}) + \gamma_2(T_{sf}^{m+1} - T_{i2}^{m+1})].$$

This gives the matrix equation

$$\left[\left(\frac{dF_0}{dT_{sf}}\right)^* - K_{i1}(\gamma_1 + \gamma_2)\right] T_{sf}^{m+1} + \gamma_1 K_{i1} T_s^{m+1} + \gamma_2 K_{i1} T_{i2}^{m+1} = \left(\frac{dF_0}{dT_{sf}}\right)^* T_{sf}^* - F_0^*. \qquad (77)$$

The equation for T_{i1} ty'in case 3 is

$$\rho_i c_{i1} \frac{(T_{i1}^{m+1} - T_{i1}^m)}{\Delta t} = \frac{1}{\Delta h_i} K_{i1}[\gamma_1(T_{sf}^{m+1} - T_{i1}^{m+1}) + \gamma_2(T_{sf}^{m+1} - T_{i2}^{m+1})] - K_{i2}(T_{i1}^{m+1} - T_{i2}^{m+1}) + Q_1.$$

Rearranging terms, we find

$$-\eta_{i1} K_{i1}(\gamma_1 + \gamma_2) T_{sf}^{m+1} + [1 + \eta_{i1}(K_{i2} + \gamma_1 K_{i1})] T_{i1}^{m+1} - \eta_{i1}(K_{i2} - \gamma_2 K_{i1}) T_{i2}^{m+1} = T_{i1}^m + \eta_{i1} Q_1. \qquad (78)$$

Equation (77) includes T_{i2}^{m+1} and therefore provides an unwanted matrix term two places to the right of the main diagonal. This term is eliminated by making the substitution

$$R_1 \to c_2 R_1 - c_1 R_2,$$

where *R₁* is the first matrix row, *R₂* is the second row, and $c_1 = \gamma_2 K_{i1}$ and $c_2 = -\eta_{i1}(K_{i2} - \gamma_2 K_{i1})$ are the coefficients multiplying T_{i2}^{m+1} in rows 1 and 2, respectively. The other ice layer equations for case 3 are equations (74) and (75).

Finally, for case 4 (T_{sf}=0°C and snow absent) we have the top ice layer equation

$$\rho_i c_{i1} \frac{(T_{i1}^{m+1} - T_{i1}^m)}{\Delta t} = \frac{1}{\Delta h_i} K_{i1}[\gamma_1(T_{sf} - T_{i1}^{m+1}) + \gamma_2(T_{sf} - T_{i2}^{m+1})] - K_{i2}(T_{i1}^{m+1} - T_{i2}^{m+1}) + Q_1.$$

which can be rewritten as

$$[1 + \eta_{i1}(\gamma_1 K_{i1} + K_{i2})] T_{i1}^{m+1} + \eta_{i1}(\gamma_2 K_{i1} - K_{i2}) T_{i2}^{m+1} = \eta_{i1} K_{i1}(\gamma_1 + \gamma_2) T_{sf} + T_{i1}^m + \eta_{i1} Q_1. \qquad (79)$$

The remaining ice layer equations are (74) and (75), as with the other three cases.

This completes the specification of the matrix equations for the four cases. The new temperatures are computed using a tridiagonal solver. After each iteration there is a check to see whether the following conditions hold:

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

The convergence rate of T_{sf} is also checked. If T_{sf} is oscillating and failing to converge, temperatures from successive iterations are averaged to improve convergence. When all these conditions are satisfied, typically within two to four iterations for $\Delta T_{\max} \approx 0.01^\circ\mathrm{C}$ and $\Delta F_{max} \approx 0.01\ \mathrm{W/m^2}$, the computation is complete.

5.2.2.5.3 Growth and Melting

First the expressions are derived for the enthalpy q. The enthalpy of snow (or fresh ice) is given by

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

Sea ice enthalpy is more complex, due to brine pockets whose salinity varies inversely with temperature. The specific heat of sea ice, shown by equation (64), includes the energy needed to warm or cool ice as well as the energy used to freeze or melt ice adjacent to brine pockets. Equation (64) 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 transform the units from J/kg to J/m^3 and adding a term for the energy required to raise the meltwater temperature to 0°C, we get the sea ice enthalpy:

$$q_i(T,S) = -\rho_i \left[c_0(T_m - T) + L_0\left(1 - \frac{T_m}{T}\right) - c_w T_m. \right] \qquad (80)$$

Note that equation (80) is a quadratic equation in *T*. Given the layer enthalpies, the temperatures can be computed using the quadratic formula:

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

where

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

The other root is unphysical.

Melting at the top surface is demonstrated by

$$q\delta h = \begin{cases} (F_0 - F_{ct})\Delta t & if F_0 > F_{ct} \\ 0 & otherwise \end{cases} \qquad (81)$$

where q is the enthalpy of the surface ice or snow layer (recall that $q < 0$) and δh is the change in thickness. If the layer melts completely, the residual flux is used to melt the layers below. Any energy remaining when the ice and snow have melted is added to the ocean mixed layer. Ice cannot grow at the top surface, but new snow can fall. Snowfall is added at the end of the thermodynamic time step.

Growth and melting at the bottom ice surface are given by

$$q\delta h = (F_{cb} - F_{bot})\Delta t, \qquad (82)$$

where F_{bot} is given by equation (61) and F_{cb} is the conductive heat flux at the bottom surface:

$$F_{cb} = \frac{k_{i,N+1}}{\Delta h_i}[\gamma_1(T_{iN} - T_f) + \gamma_2(T_{i,N-1} - T_f)].$$

If ice is melting at the bottom surface, then q in equation (82) is the enthalpy of the bottom ice layer. If ice is growing, q is the enthalpy of new ice with temperature T_f and salinity S_{max}. This ice is added to the bottom layer.

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

$$(\rho L_v - q)\delta h = F_l \Delta t, \tag{83}$$

where ρ is the density of the surface material (snow or ice), and $L_v = 2.501 \times 10^6$ J/kg is the latent heat of vaporization of liquid water at 0°C. Notice that ρL_v is close to 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 the same thicknesses. Therefore the layer interfaces are adjusted, conserving energy, so as to re-establish layers of equal thickness $\Delta h_i = h_i/N_i$. This is accomplished by computing the overlap η_{km} of each new layer k with each old layer m:

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

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

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

At the end of the time step there is a check of whether the snow is deep enough to be partially below freeboard (i.e., below the surface of the ocean). Using Archimedes' principle, the base of the snow is at freeboard when

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

So then the snow base is below freeboard when

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

In this case the snow base is raised to freeboard 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 exceptional cases this process may increase the ice thickness substantially. Therefore we postpone snow-ice conversions until after the remapping in thickness space (Section 5.2.2.2), which assumes that ice growth during a single time step is quite small. Lateral melting is accomplished by multiplying the state variables by 1-r_{side}, where r_{side} is the fraction of ice melted laterally, and adjusting the ice energy and fluxes as appropriate.

5.3 Primary PIPS 3.0 FORTRAN Routines

5.3.1 PIPS 3.0 Modules

Module	**Description**
ice_albedo	Snow and ice albedo parameterization and aggregation. **I/O:** None **Calls:** ice_kinds_mod, ice_domain **Called by:** icemodel, ice_coupling, input_data, mixed_layer, absorbed_solar, scale_fluxes, runtime_diags, ice_write_hist **I/O Parameters:** None
ice_atmo	Atmospheric boundary interface (stability based flux calculations). **I/O:** None **Calls:** ice_domain, ice_constants, ice_flux, ice_state **Called by:** None **I/O Parameters:** None
ice_calendar	Calendar routines for managing time. **I/O:** None **Calls:** ice_constants **Called by:** evp_prep, dumpfile, icecdf, icemodel, ice_coupling, ice_diagnostics, ice_flux_in, ice_itd_linear, ice_mechred, ice_therm_vertical, ice_therm_itd, ice_transport_remap, ice_write_hist, init_hist, input_data, init_evp, mixed_layer, mpdata, restartfile, zap_small_areas **I/O Parameters:** None
ice_constants	This module defines a variety of physical and numerical constants used throughout the ice model. **I/O:** None **Calls:** If coupled, shr_const_mod. Otherwise, ice_kinds_mod, ice_domain **Called by:** albedos, global_gather, global_scatter, ice_atmo, icecdf, ice_calendar, ice_coupling, ice_diagnostics, ice_dyn_evp, ice_flux, ice_flux_in, ice_grid, ice_itd, ice_itd_linear, ice_mechred, ice_ocean, ice_scaling, ice_therm_itd, ice_therm_vertical, ice_timers, ice_transport_mpdata, ice_transport_remap, ice_write_hist, init_flux, init_hist, init_state **I/O Parameters:** None
ice_coupling	Message passing to and from the coupler. **I/O:** None **Calls:** ice_kinds_mod, ice_model_size, ice_constants, ice_grid, ice_state, ice_flux, ice_albedo, ice_mpi_internal, ice_timers, ice_fileunits, ice_work (only: *worka, work_l1*). If coupled, shr_sys_mod (only: *shr_sys_flush*), cpl_contract_mod, cpl_interface_mod, cpl_fields_mod **Called by:** icemodel, dumpfile, input_data, restartfile, setup_mpi **I/O Parameters:** None
ice_diagnostics	Diagnostic information output during a model run. **I/O:** None **Calls:** ice_domain, ice_constants, ice_calendar, ice_fileunits, ice_work (only: *work_g2, work_11, work 12, worka, workb*)

Module	**Description**
	Called by: icemodel, debug_ice, ice_therm_itd, ice_therm_vertical, input_data, read_clim_data, read_data **I/O Parameters:** None
ice_domain	Sets array sizes for the local subdomain and related parallel processing information. This code was originally based on *domain.F* in the POP model. **I/O:** None **Calls:** ice_kinds_mod, ice_model_size **Called by:** abort_ice, aggregate, icemodel, ice_atmo, ice_dyn_evp, ice_flux_in, ice_grid, ice_history, ice_init, ice_itd_linear, ice_mechred, ice_read_write, ice_scaling, ice_therm_itd, ice_therm_vertical, ice_timer_print, ice_transport_mpdata, ice_transport_remap **I/O Parameters:** None
ice_dyn_evp	This is the elastic-viscous-plastic sea ice dynamics model. It computes ice velocity and deformation. References: [18], [17], [19] and [20]. **I/O:** None **Calls:** ice_kinds_mod, ice_domain, ice_grid, ice_constants, ice_state, ice_work (only: *worka, workb*) **Called by:** icemodel, dumpfile, ice_write_hist, input_data, restartfile **I/O Parameters:** None
ice_exit	Exits the model. Logically, this routine should be used for "normal" exit of the ice model, but there would be only one such call, and creating a subroutine here to accomplish that causes circular dependencies due to the coupler exit strategy. **I/O:** None **Calls:** ice_kinds_mod **Called by:** None **I/O Parameters:** None
ice_fileunits	Defines unit numbers for files opened for reading or writing. **I/O:** None **Calls:** ice_kinds_mod **Called by:** abort_ice, calendar, icemodel, ice_coupling, ice_diagnostics, ice_flux_in, ice_global_real_minmax, ice_grid, ice_history, ice_itd, ice_itd_linear, ice_mechred, ice_read_write, ice_therm_vertical, ice_timer_print, ice_transport_mpdata, ice_transport_remap, init_evp **I/O Parameters:** None
ice_flux	Flux variable declarations. These include fields sent from the coupler ("in"), sent to the coupler ("out"), written to diagnostic history files ("diagnostic"), and used internally ("internal"). **I/O:** None **Calls:** ice_kinds_mod, ice_domain, ice_constants **Called by:** aggregate, check_state, dumpfile, evp_finish, evp_prep, ice_atmo, ice_coupling, ice_flux_in, ice_scaling, ice_therm_itd, ice_therm_vertical, ice_write_hist, init_evp, init_flux, init_hist, init_state, mixed_layer, print_state,

Module	**Description**
	ridge_ice, runtime_diags, shift_ice, stepu, transport_mpdata, zap_small_areas **I/O Parameters:** None
ice_flux_in	Reads and interpolates forcing data for atmospheric and oceanic quantities. **I/O:** None **Calls:** ice_kinds_mod, ice_domain, ice_constants, ice_flux, ice_calendar ice_read_write, ice_fileunits **Called by:** icemodel, input_data **I/O Parameters:** None
ice_grid	This module contains spatial grids, masks, and boundary conditions. **I/O:** None **Calls:** ice_kinds_mod, ice_constants, ice_domain, ice_fileunits, ice_mpi_internal, ice_work (only: *work_g1, work_g2, work_11, worka*) **Called by:** aggregate, atmo_boundary_layer, bound_aggregate, bound_state, icemodel, debug_ice, dumpfile, icecdf, ice_mechred, ice_scaling, ice_therm_itd, ice_therm_vertical, ice_transport_mpdata, ice_transport_remap, ice_write_hist, init_state, mixed_layer, rebin, reduce_area, restartfile **I/O Parameters:** None
ice_history	Reads/Writes ice model history and restart files. Output files include netCDF data and Fortran unformatted dumps. **I/O:** None **Calls:** ice_kinds_mod, ice_domain, ice_read_write, ice_fileunits, ice_work (only: *work_g1, work_gr, worka*) **Called by:** icemodel, input_data **I/O Parameters:** None
ice_init	Parameter and variable initializations. **I/O:** None **Calls:** ice_domain **Called by:** icemodel, ice_transport_mpdata **I/O Parameters:** None
ice_itd	Initializes and redistributes ice in the ITD. *Ice_itd* contains routines to initialize the ice thickness distribution and utilities to redistribute ice among categories. These routines are not specific to a particular numerical implementation. References: [7], [6]. **I/O:** None **Calls:** abort_ice, ice_kinds_mod, ice_model_size, ice_constants, ice_state, ice_fileunits, **Called by:** icemodel, debug_ice, ice_itd_linear, ice_mechred, ice_therm_itd, ice_therm_vertical, ice_transport_remap, init_thermo_vertical, thermo_itd, transport_mpdata **I/O Parameters:** None
ice_itd_linear	Linear remapping scheme for the ITD. **I/O:** None **Calls:** ice_model_size ice_kinds_mod, ice_domain, ice_constants ice_state, ice_itd, ice_calendar, ice_fileunits

Module	**Description**
	Called by: icemodel, thermo_itd **I/O Parameters:** None
ice_kinds_mod	Defines variable precision for all common data types. **I/O:** None **Calls:** None **Called by:** icemodel, debug_ice, ice_albedo, ice_constants, ice_coupling, ice_domain, ice_exit, ice_dyn_evp, ice_fileunits, ice_flux, ice_flux_in, ice_grid, ice_history, ice_itd, ice_itd_linear, ice_mpi_internal, ice_ocean, ice_scaling, ice_state, ice_therm_itd, ice_therm_vertical, ice_timers, ice_transport_remap, ice_work, print_state **I/O Parameters:** None
ice_mechred	Ice mechanical redistribution (ridging) and strength computations. **I/O:** None **Calls:** ice_model_size, ice_constants, ice_state, ice_itd, ice_grid, ice_fileunits, ice_domain, ice_calendar, ice_work (only: *worka, workb*) **Called by**: icemodel **I/O Parameters:** None
icemodel	Main driver routine for PIPS 3.0. Initializes and steps through the model. **I/O:** None **Calls:** ice_albedo, ice_calendar, ice_coupling, ice_diagnostics, ice_domain, ice_dyn_evp, ice_fileunits, ice_flux_in, ice_grid, ice_history, ice_init, ice_itd, ice_itd_linear, ice_kinds_mod, ice_mechred, ice_mpi_internal, ice_ocean, ice_scaling, ice_therm_vertical, ice_therm_itd, ice_timers, ice_transport_mpdata, ice_transport_remap **Called by:** None **I/O Parameters:** None
ice_model_size	Defines the global domain size and number of categories and layers. The code is based on **model_size.F** in the POP model. **I/O:** None **Calls:** ice_kinds_mod **Called by:** columngrid, dumpfile, global_gather, global_scatter, icecdf, ice_coupling, ice_domain, ice_itd, ice_itd_linear, ice_mechred ice_read_write, ice_state, ice_therm_itd, ice_therm_vertical, ice_transport_mpdata, ice_transport_remap, init_state, print_state, rectgrid, restartfile, runtime_diags, tlatlon **I/O Parameters:** None
ice_mpi_internal	Parameters and common blocks for MPI parallelization internal to the ice model. **I/O:** None **Calls:** ice_kinds_mod, ice_domain **Called by:** abort_ice, conserved_sums, debug_ice, icecdf, icemodel, ice_coupling, ice_grid, ice_read_write, ice_timer_print, init_diags, init_mass_diags, restartfile, runtime_diags, setup_mpi **I/O Parameters:** None
ice_ocean	Ocean mixed layer calculation (internal to sea ice model). It allows heat storage

Module	**Description**

in the ocean for uncoupled runs.
I/O: None
Calls: ice_kinds_mod, ice_constants
Called by: icemodel, thermo_vertical
I/O Parameters: None

ice_read_write

Routines for opening, reading and writing external files.
I/O: None
Calls: ice_model_size, ice_domain, ice_mpi_internal, ice_fileunits, ice_work (only: *work_g1, work_gr*)
Called by: ice_flux_in, ice_history, pipsgrid, popgrid
I/O Parameters: None

ice_scaling

Scales ice fluxes by ice area.
I/O: None
Calls: ice_domain, ice_kinds_mod, ice_constants, ice_state, ice_flux, ice_grid (only: *tmask*)
Called by: icemodel
I/O Parameters: None

ice_state

Primary state variables in various configurations.

The primary state variable names are:

For Each Category	Aggregated Over Units	Categories
aicen(i,j,n)	aice(i,j)	---
vicen(i,j,n)	vice(i,j)	m
vsnon(i,j,n)	vsno(i,j)	m
eicen(i,j,k)	eice(i,j)	J/m^2
esnon(i,j,n)	esno(i,j)	J/m^2
Tsfcn(i,j,n)	Tsfc(i,j)	deg

Area is dimensionless because *aice* is the fractional area (normalized so that the sum over all categories, including open water, is 1.0). That explains why *vice/vsno* has units of m instead of m^3, and *eice/esno* have units of J/m^2 instead of J.

I/O: None
Calls: ice_kinds_mod, ice_model_size, ice_domain
Called by: albedos, dumpfile, ice_atmo, ice_coupling, ice_dyn_evp, ice_itd, ice_itd_linear, ice_mechred, ice_scaling, ice_therm_vertical, ice_therm_itd, ice_transport_remap, ice_write_hist, init_diagnostics, init_flux_atm, init_mass_diags, init_state, merge_fluxes, mixed_layer, mpdata, prepare_forcing, print_state, restartfile, runtime_diags, transport_mpdata
I/O Parameters: None

ice_therm_itd

Thermodynamic calculations after call to coupler, mostly related to ITD: ice thickness redistribution, lateral growth and melting, and freeboard adjustment.

Module	**Description**
	NOTE: The thermodynamic calculation is split in two for load balancing. First *ice_therm_vertical* computes vertical growth rates and coupler fluxes. Then *ice_therm_itd* does thermodynamic calculations not needed for coupling. **I/O:** None **Calls:** ice_kinds_mod, ice_model_size, ice_constants, ice_domain, ice_state, ice_flux, ice_diagnostics, ice_calendar, ice_grid, ice_itd **Called by**: icemodel **I/O Parameters:** real hicen- ice thickness (m)
ice_therm_vertical	Thermodynamic calculations before calls to the coupler. This module updates the ice and snow internal temperatures and computes thermodynamic growth rates and atmospheric fluxes. References: [7] NOTE: The thermodynamic calculation is split in two for load balancing. First *ice_therm_vertical* computes vertical growth rates and coupler fluxes. Then *ice_therm_itd* does thermodynamic calculations not needed for coupling. **I/O:** None **Calls:** ice_model_size, ice_kinds_mod, ice_domain, ice_fileunits, ice_constants, ice_calendar, ice_grid, ice_state, ice_flux, ice_itd, ice_diagnostics (only: *print_state*) **Called by:** icemodel, thermo_itd **I/O Parameters:** None
ice_timers	Timing routines for the PIPS 3.0 model. **I/O:** None **Calls:** ice_kinds_mod, ice_constants **Called by:** bound_ijn, evp, icemodel, ice_coupling, ice_transport_remap, ridge_ice, thermo_itd, thermo_vertical, transport_mpdata **I/O Parameters:** None
ice_transport_mpdata	Calculates horizontal advection using MPDATA (Multidimensional Positive Definite Advection Transport Algorithm). **I/O:** None **Calls:** ice_model_size, ice_domain, ice constants, ice_grid, ice_fileunits, ice_init (only: *advection*), ice_work (only: *work_l1*, *work_l2*) **Called by:** icemodel **I/O Parameters:** None
ice_transport_remap	Transports quantities using the second-order conservative remapping scheme developed by John Dukowicz and John Baumgardner (DB) and modified for sea ice by William Lipscomb and Elizabeth Hunke. **I/O:** None **Calls:** ice_calendar, ice_constants, ice_domain, ice_fileunits, ice_grid, ice_itd, ice_kinds_mod, ice_model_size, ice_state, ice_timers, ice_work (only: *work_l1*) **Called by:** icemodel **I/O Parameters:** None
ice_work	The intent of this subroutine is to have one global work array available all the time, and another available that can be allocated when necessary. Globally

Module	Description
	accessible, local (i.e., on-processor) work arrays are also available to conserve memory. These arrays should be used only within a single subroutine. **I/O:** None **Calls:** ice_kinds_mod, ice_domain **Called by:** ice_coupling, ice_diagnostics, ice_dyn_evp, ice_grid, ice_history, ice_mechred, ice_read_write, ice_transport_mpdata, ice_transport_remap, sss_clim, thermo_vertical, shift_ice **I/O Parameters:** None

5.3.2 PIPS 3.0 Subroutines

Subroutine	Description
abort_ice	This routine aborts the ice model and prints an error message. **I/O:** stdout **Calls:** ice_domain, ice_fileunits, ice_mpi_internal. If coupled, shr_sys_mod **Called by:** aggregate_area, check_monotonicity, columngrid, column_conservation_check, conservation_check_vthermo, global_conservation, init_grid, init_hist, init_itd, init_remap, init_vertical_profile, input_data, mpdata, ridge_ice, ridge_shift, setup_mpi shift_ice, temperature_changes, update_fields, zap_small_areas **I/O Parameters:** character error_message (input)
absorbed_solar	Computes solar radiation absorbed in ice and penetrating to the ocean. **I/O:** None **Calls:** ice_albedo **Called by:** temperature_changes **I/O Parameters:** integer n- thickness category index integer icells- no. of cells with *aicen* > puny integer indxi, indxj- compressed indices for cells with *aicen* > puny real hlyr- ice layer thickness real hsn- snow thickness (m) real fswsfc- SW absorbed at ice/snow surface (W/m^2) real fswint- SW absorbed in ice interior, below surface (W/m^2)

Subroutine	Description
	real fswthrun- SW through ice to ocean (W/m^2) real Iabs- SW absorbed in particular layer (W/m^2)
add_new_ice	Given the volume of new ice grown in open water, this subroutine computes its area and thickness and adds it to the appropriate category or categories. NOTE: Usually all the new ice is added to category 1. An exception is made if there is no open water or if the new ice is too thick for category 1, in which case ice is distributed evenly over the entire cell. Subroutine *rebin* should be called in case the ice thickness lies outside category bounds after new ice formation. **I/O:** None **Calls:** column_conservation_check, column_sum **Called by**: thermo_itd **I/O Parameters:** None
add_new_snow	Adds new snow at top surface. **I/O:** None **Calls:** None **Called by:** thermo_vertical **I/O Parameters:** integer n - thickness category index integer icells- number of cells with *aicen* > puny integer indxi, indxj- compressed indices for cells with *aicen* > puny real Tsf- ice/snow surface temperature, T_{sfcn} real hsn- snow thickness (m) real qsn- snow enthalpy real hsn_new- thickness of new snow (m)
aggregate	Aggregates ice state variables over thickness categories. **I/O:** None **Calls:** ice_domain, ice_flux (only: *Tf*), ice_grid **Called by**: icemodel, init_state, restartfile, thermo_itd **I/O Parameters:** None
aggregate_area	Aggregates ice area (but not other state variables) over thickness categories. **I/O:** stdout **Calls:** abort_ice **Called by**: linear_itd **I/O Parameters:** None

Subroutine	Description
albedos	Computes snow/ice albedos and aggregate. Ice albedo is zero if no ice is present. **Calls:** ice_constants, ice_grid, ice_state **Called by:** icemodel **I/O Parameters:** None
asum_ridging	Finds the total area of ice plus open water in each grid cell. This is similar to the *aggregate_area* subroutine except that the total area can be greater than 1, so the open water area is included in the sum instead of being computed as a residual. **I/O:** None **Calls:** None **Called by:** ridge_ice, ridge_prep **I/O Parameters:** None
atmo_boundary_layer	Computes coefficients for atmosphere-ice fluxes, stress and 2 meter reference temperature. NOTE: (1) all fluxes are positive downward. (2) net heat flux = *fswabs* + *flwup* + (1-emissivity)*flwdn* + *fsh* + *flh*. (3) here, *tstar* = (WT)/U*, and *qstar* = (WQ)/U*. (4) wind speeds should all be above a minimum speed (eg. 1.0 m/s). **I/O:** None **Calls:** ice_grid **Called by**: mixed_layer, thermo_vertical **Assumptions:** The saturation humidity of air at T(K): qsat(T) (kg/m^3). **I/O Parameters:** integer n- thickness category index character sfctype- ice or ocean real Tsf- surface temp of ice or ocean real strx- x surface stress real stry- y surface stress real Trf- reference height temp (K) real Qrf- reference height specific humidity (kg/kg) real delt- potential T difference (K) real delq- humidity difference (kg/kg)

Subroutine	Description
bound	Fills ghost cells with boundary information. **I/O:** None **Calls:** bound_ijn **Called by:** bound_aggregate, evp_prep, from_coupler, init_evp, init_grid, init_remap, init_state, local_max_min, makemask, mpdata, restartfile, transport_remap, t2ugrid, tlatlon, to_coupler, u2tgrid **I/O Parameters:** real work1
bound_aggregate	Bound calls for aggregate ice state. Gets ghost cell values for aggregate ice state variables. NOTE: This subroutine is called only at initialization. **I/O:** None **Calls:** ice_grid, bound **Called by**: init_state, restartfile **I/O Parameters:** None
bound_ijn	Periodic/Neumann conditions for global domain boundaries. Assumptions: A "single" row of ghost cells (*num-ghost-cells=1*); *work1* array has form (*i-index,j-index,number-arrays*). **I/O:** None **Calls:** ice_timers **Called by**: bound, bound_narr, bound_narr_ne, bound_sw **I/O Parameters:** integer nd, real work1, logical north, south, east, west
bound_narr	Fills neighboring ghost cells with boundary information and fills several arrays at once (for performance). **I/O:** None **Calls:** bound_ijn **Called by:** bound_state, mpdata, transport_remap, transport_mpdata **I/O Parameters:** integer narrays, real work1
bound_narr_ne	Fills north and east ghost cells with boundary information and fills several arrays at once (for performance). **I/O:** None **Calls:** bound_ijn **Called by:** stepu **I/O Parameters:** integer narrays, real work1

Subroutine	Description
bound_state	Bound calls for ice state variables. Gets ghost cell values for ice state variables in each thickness category. **I/O:** None **Calls:** ice_grid, bound_narr **Called by**: restartfile, transport_remap **I/O Parameters:** None
bound_sw	Fills south and west ghost cells with boundary information. **I/O:** None **Calls:** bound_ijn **Called by**: evp, transport_remap **I/O Parameters:** real work1
calendar	Determines the date at the end of the time step. **I/O:** stdout **Calls:** ice_fileunits **Called by**: icemodel, init_cpl **I/O Parameters:** real ttime- time variable
check_monotonicity	At each grid point, this subroutine assures that the new tracer values fall between the local maximum and minimum values before transport. **I/O:** stdout **Calls:** abort_ice **Called by:** transport_remap **I/O Parameters:** real aim- new ice area real trm- new tracers real tmin- local min tracer real tmax- local max tracer
check_state	This subroutine requires certain fields to be monotone. NOTE: This should not be necessary if all is well, but it is best to keep going. The model will not conserve energy and water if fields are zeroed here. **I/O:** None **Calls:** ice_flux **Called by**: transport_mpdata **I/O Parameters:** None

Subroutine	Description
column_conservation_check	For each physical grid cell, the routine checks that initial and final values of a conserved field are equal to within a small value. **I/O:** stdout **Calls:** abort_ice **Called by:** add_new_ice, linear_itd, ridge_shift **I/O Parameters:** real x1- initial field real x2- final field real max_err- max allowed error character fielded- field identifier
column_sum	For each grid cell, the subroutine sums the field over all ice categories. **I/O:** None **Calls:** None **Called by:** add_new_ice, linear_itd, ridge_shift **I/O Parameters:** integer nsum- no. of categories/layers real xin- input field real xout- output field
columngrid	Constructs column grid and mask. **I/O:** stdout **Calls:** global_scatter, ice_model_size, abort_ice **Called by**: init_grid **I/O Parameters:** None
complete_getflux_ocn	Computes remaining ocean forcing fields. **I/O:** None **Calls:** None **Called by**: sss_clim, sss_sst_restore **I/O Parameters:** None

Subroutine	Description
conductivity	Computes thermal conductivity at interfaces (held fixed during the subsequent iteration). NOTE: Ice conductivity must be >= *kimin* **I/O:** None **Calls:** None **Called by:** temperature_changes **I/O Parameters:** integer icells- no. of cells with *aicen* > puny integer indxi, indxj- compressed indices for cells with *aicen* > puny real hlyr- ice layer thickness real hsn- snow layer thickness real Tbot- ice bottom surface temp (°C) real Tin- internal ice layer temperatures real khi- *ki/hlyr* real khs- *ksno/hsn*
conservation_check_vthermo	Checks for energy conservation by comparing the change in energy to the net energy input. **I/O:** stdout **Calls:** print_state, abort_ice **Called by:** thermo_vertical **I/O Parameters:** integer n- thickness category index (diagnostic only) integer icells- number of cells with *aicen* > puny integer indxi, indxj- compressed indices for cells with *aicen* > puny real fsurf - net flux to top surface, not including *fcondtop* real flatn - surface downward latent heat (W/m^2) real fhnetn- *fbot*, corrected for any surplus energy real fswint- SW absorbed in ice interior, below surface (W/m^2) real einit - initial energy of melting (J/m^2) real efinal- final energy of melting (J/m^2)

Subroutine	Description
conserved_sums	Computes global sums of conserved variables over the physical grid. **I/O:** None **Calls:** ice_mpi_internal, ice_global_real_sum **Called by:** transport_remap **I/O Parameters:** real aim- mean ice area real asum- global sum of area real trm- mean tracer real atsum- global sum of area*tracer
construct_fields	Constructs fields of ice area and tracers. **I/O:** None **Calls:** limited_gradient **Called by:** transport_remap **I/O Parameters:** real aim- mean ice area real aimask- = 1. if ice is present, = 0. otherwise real trm- mean tracer real trmask- = 1. if tracer is present, =0. otherwise. real aic- ice area at geometric center of cell real aix, aiy- limited derivative of ice area with respect to *x* and *y* real trc- tracer at geometric center of cell real trx, try- limited derivative of tracer with respect to *x* and *y*
debug_ice	Wrapper for the print_state debugging routine. It is useful for debugging in the main driver. **I/O:** None **Calls:** ice_kinds_mod, ice_itd, ice_diagnostics, ice_mpi_internal, ice_grid, print_state **Called by:** None **I/O Parameters:** character (char_len), intent(in) :: plabeld
departure_points	Given velocity fields on cell corners, this subroutine computes departure points of trajectories using a midpoint approximation. **I/O:** stdout **Calls:** None **Called by:** transport_remap **I/O Parameters:** real dpx, dpy- *x* and *y* coordinates of departure points at cell corners

Subroutine	Description
dumpfile	Dumps all values needed for a restart. **I/O:** stdout, write filename, nu_dump **Calls:** ice_model_size, ice_flux, ice_grid, ice_calendar ice_state, ice_dyn_evp, ice_ocean (only: *oceanmixed_ice*), ice_open, ice_coupling (only: *l_coupled*), ice_write **Called by:** icemodel **I/O Parameters:** None
end_run	Ends run by calling *mpi_finalize*. **I/O:** None **Calls:** mpi_finalize **Called by:** icemodel, abort_ice **I/O Parameters:** None
evp	Elastic-viscous-plastic dynamics driver. **I/O:** None **Calls:** ice-timers, ice_timer_start, evp_prep, stepu, stress, bound_sw, evp_finish, ice_timer_stop **Called by:** icemodel **I/O Parameters:** integer kstrngth- input
evp_finish	Calculates ice-ocean stress. **I/O:** None **Calls:** ice_flux, iceumask, u2tgrid **Called by:** evp **I/O Parameters:** None
evp_prep	Computes quantities needed in the stress tensor (sigma) and momentum (u) equations, but the following quantities do not change during the thermodynamics/transport time step: wind stress shift to U grid, ice mass and ice extent masks, pressure (strength), and part of the forcing stresses, initializes ice velocity for new points to ocean surface current. **I/O:** None **Calls:** bound, tmask, t2ugrid, to_ugrid, iceumask, ice_strength, ice_flux, ice_calendar, ice_mechred **Called by:** evp **I/O Parameters:** integer kstrngth- input
exit_coupler	Exits from coupled/MPI environment. **I/O:** stdout **Calls:** cpl_interface_finalize, mpi_abort **Called by**: icemodel **I/O Parameters:** None

Subroutine	Description
file_year	Constructs the correct name of the atmospheric data file to be read, given the year and assuming the naming convention of filenames ending with 'yyyy.dat'. **I/O:** stdout (a, 14.4, a) **Calls:** None **Called by**: ncar_files, read_data **I/O Parameters:** character data file
fit_line	Fits *g(h)* with a line, satisfying area and volume constraints. To reduce round off errors caused by large values of *g0* and *g1*, it computes *g(eta)*, where *eta = h - hL*, and *hL* is the left boundary. **I/O:** None **Calls:** None **Called by:** linear_itd **I/O Parameters:** integer n- category index real HbL, HbR- left and right cat boundaries real hice- ice thickness real g0, g1- coefficients in linear equation for *g(eta)* real hL-min value of range over which *g(h)* > 0 real hR- max value of range over which *g(h)* > 0 logical reamp_flag
flux_integrals	Computes the fluxes across each edge by integrating the ice area and tracers over each flux triangle. Input variables have the same meanings as in the main subroutine. Repeated use of certain sums makes the calculation more efficient. **I/O:** None **Calls:** None **Called by:** transport_remap **I/O Parameters:** real triarea real xp0, yp0 real xp1, yp1 real xp2, yp2 real xp3, yp3 integer iflux, jflux real aic, aix, aiy real aiflx real trc, trx, try real atflx

Subroutine	Description
freeboard	If there is enough snow to lower the ice/snow interface below sea level, this subroutine converts enough snow to ice to bring the interface back to sea level. NOTE: Subroutine *rebin* should be called after *freeboard* to make sure ice thicknesses are within category bounds. **I/O:** None **Calls:** None **Called by**: thermo_itd **I/O Parameters:** None
from_coupler	Reads input data from coupler to sea ice model. **I/O:** 100 (write) **Calls:** get_sum, ice_timer_start, ice_timer_stop, cpl_interface_contractrecv, tarea, hm, bound, anglet, t2ugrid **Called by**: icemodel **I/O Parameters:** None
frzmlt_bottom_lateral	Adjusts *frzmlt* to account for changes in *fhnet* since *from_coupler*. Computes heat flux to the bottom surface. Computes the fraction of ice that melts laterally. **I/O:** None **Calls:** None **Called by:** thermo_vertical **I/O Parameters:** real Tbot- ice bottom surface temp (°C) real fbot- heat flux to ice bottom (W/m^2) real rside- fraction of ice that melts laterally
get_sum	Computes a (weighted) sum over the global grid. If *flag* = 1 then *work1* is weighted by *work2* before being added to *work3*. **I/O:** None **Calls:** ice_global_real_sum **Called by:** runtime_diags, to_coupler **I/O Parameters:** integer flag real work1, work2, work3 real gsum
getflux	Gets forcing data and interpolates as necessary. **I/O:** None **Calls:** sss_sst_restore **Called by:** icemodel **I/O Parameters:** None

Subroutine	Description
global_conservation	Checks whether values of conserved quantities have changed. An error probably means that ghost cells are treated incorrectly. **I/O:** stdout **Calls:** abort_ice **Called by:** transport_remap **I/O Parameters:** real asum_init- initial global ice area real asum_final- final global ice area real atsum_init- initial global ice area*tracer real atsum_final- final global ice area*tracer
global_gather	Gathers a distributed array and strips off ghost cells to create a local array with global dimensions. **I/O:** None **Calls:** ice_model_size, ice_constants **Called by:** icecdf, ice_write, init_diags, init_mass_diags, runtime_diags, tlatlon **I/O Parameters:** real work
global_scatter	Scatters a global array and adds ghost cells to create a distributed array. **I/O:** None **Calls:** ice_model_size, ice_constants **Called by:** columngrid, init_grid, ice_read, rectgrid, tlatlon **I/O Parameters:** real work
ice_bcast_char	Broadcasts a scalar character value to all processors. **I/O:** None **Calls:** None **Called by:** input_data **I/O Parameters:** character charval
ice_bcast_iscalar	Broadcasts an integer scalar character value to all processors. **I/O:** None **Calls:** None **Called by:** init_cpl, init_hist, restartfile **I/O Parameters:** integer ival
ice_bcast_logical	Broadcasts a scalar logical value to all processors. **I/O:** None **Calls:** None **Called by**: ice_read, init_hist, input_data **I/O Parameters:** logical logval

Subroutine	Description
ice_bcast_rscalar	Broadcasts a real scalar character value to all processors. **I/O:** None **Calls:** None **Called by**: init_cpl, input_data, restartfile **I/O Parameters:** real val
ice_coupling_setup	This routine sets the model MPI communicators and task IDs from CCSM share code. **I/O:** stdout **Calls:** cpl_interface_init, shr_sys_flush **Called by:** setup_mpi **I/O Parameters:** character in_model_name- input integer model_comm - communicator for model (output)
ice_global_real_minmax	Determines and writes both minimum and maximum over global grid and then prints. **I/O:** stdout **Calls:** ice_fileunits, ice_global_real_minval, ice_global_real_maxval **Called by:** None **I/O Parameters:** integer nc real work(nc) character string
ice_global_real_maxval	Computes the maximum over the global grid. **I/O:** None **Calls:** None **Called by:** ice_global_real_minmax, ice_timer_print, runtime_diags **I/O Parameters:** integer nc real work(nc)
ice_global_real_minval	Computes the minimum over the global grid. **I/O:** None **Calls:** None **Called by:** ice_global_real_minmax, ice_timer_print **I/O Parameters:** integer nc real work(nc)

Subroutine	Description
ice_global_real_sum	Sums a given array over the global grid. **I/O:** None **Calls:** mpi_allreduce **Called by:** conserved_sums, get_sum, runtime_diags **I/O Parameters:** integer nc real work(nc)
ice_open	Opens an unformatted file for reading. The indication for whether the file is sequential or direct access is found in *nbits*. **I/O:** open nu (unformatted direct access open) **Calls:** None **Called by**: dumpfile, pipsgrid, popgrid, restartfile, read_clim_data, read_data, sss_clim, sst_ic **I/O Parameters:** integer nu- unit number integer nbits- no. of bits/variable (0 for sequential access) character filename
ice_read	Reads an unformatted file. *Work* is a real array, *atype* indicates the format of the data. **I/O:** stdout, read nu (unformatted read) **Calls:** None **Called by:** pipsgrid, popgrid, read_clim_data, read_data, restartfile, sss_clim, sst_ic **I/O Parameters:** integer nu- unit number integer nrec- record number (0 for sequential access) real work- output array (real, 8-byte) character atype- format for input array (real/int, 4-byte/8-byte) logical scatter- if true, scatter the data logical diag- if true, write diagnostic output logical ignore_eof, hit_eof

Subroutine	Description
ice_strength	Computes the strength of the ice pack, defined as the energy (J/m^2) dissipated per unit area removed from the ice pack under compression and assumed proportional to the change in potential energy caused by ridging. References: [35] and [16]. For simpler strength parameterization, see [15]. **I/O:** None **Calls:** ridge_prep **Called by:** evp_prep **I/O Parameters:** integer kstrngth- =1 for Rothrock formulation [35], 0 for Hibler formation [15]
ice_timer_clear	Initializes timer *n* to 0. If *n* = -1, all timers are initialized. **I/O:** None **Calls:** None **Called by:** icemodel **I/O Parameters:** integer n- timer number
ice_timer_print	Prints timing results of timer *n*. If *n* = -1 the timing results of all timers are printed. **I/O:** write 100 **Calls:** ice_domain, ice_mpi_internal, ice_fileunits, ice_global_real_maxval, ice_global_real_minval, **Called by**: icemodel **I/O Parameters:** integer n- timer number
ice_timer_start	Begin timing with timer *n*. **I/O:** None **Calls:** timers **Called by**: bound_ijn, evp, icemodel, from_coupler, ridge_ice, thermo_itd, thermo_vertical, to_coupler, transport_mpdata, transport_remap **I/O Parameters:** integer n- timer number
ice_timer_stop	Ends (or pauses) timing with timer *n*. **I/O:** None **Calls:** timers **Called by**: bound_ijn, evp, icemodel, from_coupler, ridge_ice, thermo_itd, thermo_vertical, to_coupler, transport_mpdata, transport_remap **I/O Parameters:** integer n- timer number

Subroutine	Description
ice_write	Writes an unformatted file. *Work* is a real array, *atype* indicates the format of the data. **I/O:** stdout, write nu (unformatted write) **Calls:** global_gather **Called by**: dumpfile **I/O Parameters:** integer nu- unit number integer nrec- record number (0 for sequential access) real work- input array character atype- format for output array logical gather- if true, gather the data
ice_write_hist	Writes average ice quantities or snapshots. **I/O:** None **Calls:** icecdf, ice_flux, ice_albedo, ice_mechred, ice_grid, ice_calendar, ice_state, ice_dyn_evp, ice_constants, principal_stress **Called by**: icemodel **I/O Parameters:** None
icecdf	Writes netCDF history file. **I/O:** stdout, write ncfile **Calls:** global_gather, ice_model_size, ice_constants, ice_mpi_internal, ice_grid, ice_calendar **Called by**: ice_write_hist **I/O Parameters:** None
init_calendar	Initializes calendar variables. **Calls:** None **Called by:** icemodel **I/O Parameters:** None
init_constants	Initializes constants that are best defined at run time (e.g. pi). **Calls:** None **Called by:** icemodel **I/O Parameters:** None
init_cpl	Initializes message passing between ice and coupler. **I/O:** stdout **Calls:** None **Called by**: icemodel **I/O Parameters:** None

Subroutine	Description
init_diagnostics	Initializes diagnostic fields written to history files. **I/O:** None **Calls:** ice_state **Called by**: icemodel, thermo_vertical **I/O Parameters:** None
init_diags	Finds tasks for requested points. **I/O:** stdout, 2234 (write) **Calls:** global_gather, ice_grid, ice_mpi_internal **Called by**: icemodel **I/O Parameters:** None
init_evp	Initializes parameters needed for EVP dynamics. **I/O:** stdout (a8, f12.4) **Calls:** ice_calendar, ice_fileunits, ice_flux **Called by**: icemodel **I/O Parameters:** None
init_flux	Initializes all of the fluxes exchanged with the flux coupler and some data derived fields. **I/O:** None **Calls:** ice_constants, ice_flux, init_flux_atm, init_flux_ocn **Called by**: icemodel **I/O Parameters:** None
init_flux_atm	Initializes all fluxes sent to the coupler for use by the atmospheric model and a few state quantities. **I/O:** None **Calls:** ice_state (only: *aice*) **Called by**: init_flux **I/O Parameters:** None
init_flux_ocn	Initializes fluxes sent to the coupler for use by the ocean model. **I/O:** None **Calls:** None **Called by:** init_flux, thermo_itd **I/O Parameters:** None

Subroutine	Description
init_getflux	Determines the final year of the forcing cycle based on namelist input. It initializes the forcing data filenames and initializes surface temperature and salinity from data. **I/O:** stdout **Calls:** ice_history (only: *restart*), ncar_files, sss_clim, sst_ic **Called by:** icemodel **I/O Parameters:** None
init_grid	Horizontal grid initialization routine. HT{N,E} = cell widths on {N,E} sides of T cell; U{LAT,LONG} = true {latitude, longitude} of U points; D{X,Y}{T,U} = {x,y} spacing centered at {T,U} points. **I/O:** None **Calls:** None **Called by**: icemodel **I/O Parameters:** None
init_hist	Initializes history files. **I/O:** stdout, open nu_nml, read nu_nml **Calls:** abort_ice, ice_bcast_iscalar, ice_bcast_logical, ice_constants, ice_calendar, ice_flux (only: *mlt_onset*, *frz_onset*),. If coupled, shr_sys_mod (only: *shr_sys_flush*) **Called by**: icemodel **I/O Parameters:** None
init_itd	Initializes area fraction and thickness boundaries for the ITD model. **I/O:** stdout **Calls:** abort_ice **Called by:** icemodel **I/O Parameters:** None
init_mass_diags	Computes the global combined ice and snow mass sum. **I/O:** None **Calls:** get_sum, global_gather, ice_mpi_internal, ice_state **Called by:** icemodel **I/O Parameters:** None
init_mechred	Initializes some variables written to the history file. **I/O:** None **Calls:** None **Called by:** icemodel **I/O Parameters:** None

Subroutine	Description
init_remap	Initializes grid quantities used by remapping. **I/O:** stdout **Calls:** abort_ice, bound **Called by:** icemodel **I/O Parameters:** None
init_state	Initializes state for the ITD model. **I/O:** None **Calls:** aggregate, bound, bound_aggregate, ice_constants, ice_flux, ice_grid, ice_model_size, ice_state, ice_therm_vertical **Called by:** icemodel **I/O Parameters:** None
init_thermo_vars	For current thickness category, this routine initializes the thermodynamic variables that are aggregated and sent to the coupler, along with other fluxes passed among subroutines. **I/O:** None **Calls:** None **Called by:** thermo_vertical **I/O Parameters:** real strxn- air/ice zonal stress (N/m^2) real stryn- air/ice meridional stress (N/m^2) real Trefn- air temp reference level (K) real Qrefn- air speed reference level (kg/kg) real fsensn- surface downward sensible heat (W/m^2) real flatn- surface downward latent heat (W/m^2) real fswabsn- SW absorbed by ice (W/m^2) real flwoutn- upward LW at surface (W/m^2) real evapn- flux of vapor, atmosphere to ice (kg/m^2/s) real freshn- flux of water, ice to ocean (kg/m^2/s) real fsaltn- flux of salt, ice to ocean (kg/m^2/s) real fhnetn- fbot, corrected for surplus energy (W/m^2) real fswthrun- SW through ice to ocean (W/m^2) real fsurf- net flux to top surface, not inc. *fcondtop* real fcondtop- downward cond flux at top surface (W/m^2) real fcondbot- downward cond flux at bottom surf (W/m^2) real fswint- SW absorbed in ice interior, below surf (W/m^2) real einit- initial energy of melting (J/m^2) real efinal- final energy of melting (J/m^2) real mvap- ice/ snow mass sublimated/condensed (kg/m^2)

Subroutine	Description
init_thermo_vertical	Initializes the vertical profile of ice salinity and melting temperature. **I/O:** None **Calls:** ice_itd **Called by:** icemodel **I/O Parameters:** None
init_vertical_profile	Given the state variables (*vicen, vsnon, eicen, esnon, Tsfcn*), this subroutine computes variables needed for the vertical thermodynamics (*hin, hsn, qin, qsn, Tin, Tsn, Tsf*). **I/O:** stdout **Calls:** abort_ice, print_state **Called by:** thermo_vertical **I/O Parameters:** integer n- thickness category index integer icells- number of cells with *aicen* > puny integer indexi, indxj- compressed indices for cells with *aicen* > puny real hin- ice thickness (m) real hsn- snow thickness (m) real hlyr- ice layer thickness real hin_init- initial value of *hin* real hsn_init- initial value of *hsn* real qsn- snow enthalpy real Tsn- snow temperature real Tsf- ice/snow surface temperature, T_{sfcn} real qin- ice layer enthalpy (J/m^3) real Tin- internal ice layer temperatures real einit- initial energy of melting (J/m^2)
input_data	Namelist variables that are set to default values. They may be altered at run time. **I/O:** stdout, read nu_nml, write 1000, 1010, 1020, 1030, 1050, 1060, 1070, 1080, 1090, 1095 **Calls:** abort_ice, ice_albedo, ice_bcast_char, ice_bcast_iscalar, ice_bcast_logical, ice_bcast_rscalar, ice_diagnostics, ice_mechred (only: *kstrength, krdg_partic, krdg_redist*), ice_history, ice_calendar, ice_coupling (only: *l_coupled, runtype*), ice_dyn_evp, ice_itd (only: *kitd, kcatbound*), ice_ocean (only: *oceanmixed_ice*), ice_flux_in (only: *ycycle, fyear_init, atm_data_dir, ocn_data_dir*) **Called by:** icemodel **I/O Parameters:** None

Subroutine	Description
interp_coeff	Computes coefficients for interpolating data to the current time step. It works for any data interval that divides evenly into a 365-day year (daily, 6-hourly, etc.). Uses *interp_coef_monthly* for monthly data. **I/O:** None **Calls:** None **Called by:** ncar_bulk_dat **I/O Parameters:** integer recnum- record number for current data value integer recslot- spline slot for current record real secant- seconds in data interval
interp_coeff_monthly	Computes coefficients for interpolating monthly data to the current time step. **I/O:** None **Calls:** None **Called by:** ncar_bulk_dat, sss_sst_restore **I/O Parameters:** integer recslot- slot (1 or 2) for current record
interpolate_data	Linear interpolation subroutine. **I/O:** None **Calls:** None **Called by:** ncar_bulk_dat, sss_sst_restore **I/O Parameters:** real field_data- two values used for interpolation real field – interpolated field
lateral_melt	Given the fraction of ice melting laterally in each grid cell (computed in subroutine *frzmlt_bottom_lateral.f*), melt the ice. **I/O:** None **Calls:** None **Called by:** thermo_itd **I/O Parameters:** real rside- fraction of ice that melts laterally
function lenstr (label)	Computes the length string by finding the first non-blank character from the right. **I/O:** None **Calls:** None **Called by:** dumpfile, icecdf **I/O Parameters:** character label

Subroutine	Description
limited_gradient	Computes a limited gradient of the scalar field *phi*. "Limited" means that we do not create new extrema in *phi*. For instance, field values at the cell corners can neither exceed the maximum of *phi(i,j)* in the cell and its eight neighbors, nor fall below the minimum. **I/O:** None **Calls:** None **Called by:** construct_fields **I/O Parameters:** real phi- input tracer field (mean values in each grid cell) real cnx- x-coordinate of *phi* relative to geometric center of cell real cny- y-coordinate of *phi* relative to geometric center of cell real phimask- *phimask(i,j)* = 1 if *phi(i,j)* has physical meaning, = 0 otherwise. For instance, *aice* has no physical meaning on land points, and *hice* no physical meaning where *aice* = 0 real gx- limited x-direction gradient real gy- limited y-direction gradient
linear_itd	Ice thickness distribution scheme that shifts ice among categories. The default scheme is linear remapping, which works as follows: Using the thermodynamic "velocities", it interpolates to find the velocities in thickness space at the category boundaries and computes the new locations of the boundaries. Then for each category, the scheme computes the thickness distribution function, *g(h)*, between *hL* and *hR*, the left and right boundaries of the category. **I/O:** stdout **Calls:** aggregate_area, column_sum, column_conservation_check, fit_line, shift_ice **Called by:** thermo_itd **I/O Parameters:** real hicen_old- starting value of *hicen* real hicen- ice thickness for each category (m)

Subroutine	Description
load_tracers	Loads tracer array and computes tracer dependency vectors. The default version assumes that the advected tracers are *hice, hsno, Tsfc, qice*(1:*nilyr*), and *qsno*. This subroutine must be modified if a different set of tracers is to be transported. The rule for ordering tracers is that a dependent tracer (such as *qice*) must have a larger tracer index than the tracer it depends on (i.e., *hice*). **I/O:** None **Calls:** None **Called by:** transport_remap **I/O Parameters:** integer n- ice category index real trm- mean tracer values in each grid cell
local_max_min	At each grid point, this subroutine computes the local max and min of a scalar field *phi*: i.e., the max and min values in the nine-cell region consisting of the home cell and its eight neighbors, plus the neighbors of the neighbors (25 cells in all). **I/O:** None **Calls:** bound **Called by:** transport_remap **I/O Parameters:** real aimask real trm real trmask real tmin- local min tracer real tmax- local max tracer

Subroutine	Description
locate_triangles	Computes areas and vertices of flux triangles for east and north cell edges. **I/O:** None **Calls:** None **Called by**: transport_remap **I/O Parameters:** real dpx,dpy- x, y coordinates of departure points at cell corners real triarea_e- area of east-edge flux triangle triarea_n- area of north-edge flux triangle real xp0_e, yp0_e- coordinates of special triangle points real xp0_e, yp0_e- e for east edges, n for north edges real xp1_e, yp1_e real xp2_e, yp2_e real xp3_e, yp3_e real xp0_n, yp0_n real xp1_n, yp1_n real xp2_n, yp2_n real xp3_n, yp3_n
make_masks	Creates area and tracer masks. If an area is masked out (*aim* < puny), then the values of tracers in that grid cell are assumed to have no physical meaning. Similarly, if a tracer with dependents is masked out (*abs(trm)* < puny), then the values of its dependent tracers in that grid cell are assumed to have no physical meaning. For example, the enthalpy value has no meaning if the thickness is zero. **I/O:** None **Calls:** None **Called by:** transport_remap **I/O Parameters:** real aim- mean ice area in each grid cell real aimask- 1. if ice is present, otherwise = 0 real trm- mean tracer values in each grid cell real trmask- 1. if tracer is present, else = 0
makemask	Sets the boundary values for the T cell land mask (*hm*) and makes the logical land masks for T and U cells (*tmask, umask*). This routine also creates hemisphere masks (*mask-n* Northern, *mask-s* Southern). **I/O:** None **Calls:** bound **Called by:** init_grid **I/O Parameters:** None

Subroutine	Description
merge_fluxes	Aggregates flux information from all ice thickness categories. **I/O:** None **Calls:** ice_state **Called by:** thermo_vertical **I/O Parameters:** integer n- thickness category index real strxn- air/ice zonal stress (N/m^2) real stryn- air/ice meridional stress (N/m^2) real fsensn- sensible heat flux (W/m^2) real flatn- latent heat flux (W/m^2) real fswabsn- shortwave absorbed heat flux (W/m^2) real flwoutn- upward low emitted heat flux (W/m^2) real evapn- evaporation (kg/m^2/s) real Trefn- air temp reference level (K) real Qrefn- air speed humidity reference level (kg/kg) real freshn- fresh water flux to ocean (kg/m^2/s) real fsaltn- salt flux to ocean (kg/m^2/s) real fhnetn- actual ocean/ice heat flux (W/m^2) real fswthrun- SW radiation through ice bottom (W/m^2)
mixed_layer	Computes the mixed layer heat balance and updates the SST. This routine also computes the energy available to freeze or melt ice. NOTE: SST changes due to fluxes through the ice are computed in *ice_therm_vertical*. **I/O:** None **Calls:** atmo_boundary_layer, ice_flux, ice_calendar (only: *dt*), ice_grid (only: *tmask*), ice_state, ice_albedo **Called by:** icemodel **I/O Parameters:** None
mpdata (narrays.phi)	Advection according to *mpdata*. Reference: [39]. **I/O:** stdout **Calls:** bound, bound_narr, ice_calendar, ice_state (only: *uvel, vvel*), abort_ice **Called by:** transport_mpdata **I/O Parameters:** integer narrays real phi

Subroutine	Description
NCAR_bulk_dat	Reads NCAR_bulk atmospheric data. **I/O:** readm, read6, read **Calls:** interp_coeff_monthly, read_clim_data **Called by:** getflux **I/O Parameters:** None
NCAR_files	This subroutine is based on the LANL file naming conventions. The user must edit it for other directory structures or filenames. NOTE: The year number in these filenames does not matter, because subroutine *file_year* will insert the correct year. **I/O:** stdout **Calls:** file_year **Called by:** init_getflux **I/O Parameters:** integer yr- current forcing year
pipsgrid	PIPS 3.0 rotated spherical grid and land mask.

Rec No.	Field	Units
Land Mask 1	KMT	1 grid file with real KMT's grid
2	ULAT	radians
3	ULON	radians
4	HTN	cm
5	HTE	cm
6	HUS	cm
7	HUW	cm
8	ANGLE	radians

Subroutine	Description
	I/O: stdout **Calls:** ice_read_write **Called by:** init_grid **I/O Parameters:** None
popgrid	Reads and sets POP displaced pole grid and land mask. See Subroutine *pipsgrid* description above for record number, field and units, as they are identical. **I/O:** None **Calls:** ice_read_write, ice_open, ice_read **Called by:** init_grid **I/O Parameters:** None

Subroutine	Description
prepare_forcing	Finishes the task of manipulating forcing. **I/O:** None **Calls:** ice_state (only: *aice*)- lipscombtune **Called by:** getflux **I/O Parameters:** None
principal_stress	Computes principal stresses for comparison with the theoretical yield curve; northeast values. **I/O:** None **Calls:** None **Called by:** ice_write_hist **I/O Parameters:** None
print_state	Prints ice state for a specified grid point. This routine is useful for debugging. **I/O:** stdout **Calls:** ice_model_size, ice_kinds_mod, ice_state, ice_itd, ice_flux **Called by:** conservation_check_vthermo, debug_ice, init_vertical_profile, temperature_changes **I/O Parameters:** character plabel- input character i,j- input
read_clim_data	Reads annual climatological data needed for interpolation, as in *read_data*. It assumes a one-year cycle of climatological data, so that there is no need to get data from other years or to extrapolate data beyond the forcing time period. **I/O:** stdout **Calls:** ice_diagnostics (for debugging), ice_open, ice_read **Called by:** ncar_bulk_dat, sss_sst_restore **I/O Parameters:** logical readflag integer recd- baseline record number integer imx, ixx, ipx- record numbers of three data values relative to recd character data_file real field_data- two values needed for interpolation

Subroutine	Description
read_data	Reads data for interpolation. If data is at the beginning of a one-year record, this subroutine gets data from the previous year. If data is at the end of a one-year record, it gets data from the following year. If no earlier data exists (beginning of *fyear_init*), then: (1) For monthly data, get data from the end of *year_final*. (2) For more frequent data, let the *imx* value equal the first value of the year. If no later data exists (end of *fyear_final*), then: (1) For monthly data, get data from the beginning of *fyear_init*. (2) For more frequent data, let the *ipx* value equal the last value of the year. In other words, assume persistence when daily or 6-hourly data is missing, and assume periodicity when monthly data is missing. **I/O:** stdout **Calls:** file_year, ice_diagnostics, ice_open, ice_read **Called by:** ncar_bulk_dat **I/O Parameters:** logical flag integer recd- baseline record number integer yr- year of forcing data integer imx, ixx, ipx- record numbers of three data values relative to recd integer maxrec- maximum record value real field data- two values needed for interpolation
rebin	Rebins thicknesses into defined categories. **I/O:** None **Calls:** ice_grid, shift_ice **Called by:** thermo_itd **I/O Parameters:** None
rectgrid	Regular rectangular grid and mask. **I/O:** None **Calls:** global_scatter, ice_model_size **Called by:** init_grid **I/O Parameters:** None

Subroutine	Description
reduce_area	Reduces area when ice melts for special case *ncat=1*. Use a CSM 1.0-like method of reducing ice area when melting occurs. Assumes only half the ice volume change goes to thickness decrease and the other half to reduction in ice fraction. **I/O:** None **Calls:** ice_grid **Called by:** thermo_itd **I/O Parameters:** real hice1_old- old ice thickness for category 1 (m) real hice1- new ice thickness for category 1 (m)
restartfile	Restarts from a dumpfile. **I/O:** read, write nu_rst_pointer, write filename, read nu_restart, stdout **Calls:** aggregate, bound, bound_aggregate, bound_state, ice_bcast_iscalar, ice_bcast_rscalar, ice_model_size, ice_flux, ice_mpi_internal, ice_grid, ice_calendar, ice_state, ice_dyn_evp, ice_itd, ice_ocean, ice_open, ice_read, (only: *oceanmixed_ice*), ice_coupling (only: *1_coupled*), lenstr **Called by:** icemodel **I/O Parameters:** None
ridge_ice	Computes changes in the ice thickness distribution due to divergence and shear. References: [13], [16], [35], [43]. **I/O:** stdout **Calls:** abort_ice, asum_ridging, ice_timers, ice_timer_start, ice_timer_stop, ice_flux, ridge_prep, ridge_shift **Called by:** icemodel **I/O Parameters:** real Delta- in the denominator of zeta, eta (1/s) real divu- strain rate I component, velocity divergence (1/s)

Subroutine	Description
ridge_prep	Preparation for ridging and strength calculations. Computes the thickness distribution of the ice and open water participating in ridging and of the resulting ridges. This version includes new options for ridging participation and redistribution. The new participation scheme improves model stability by increasing the time scale for large changes in ice strength. The new redistribution scheme improves the agreement between ITDs of modeled and observed ridges. **I/O:** None **Calls:** asum_ridging **Called by:** ice_strength, ridge_ice **I/O Parameters:** None
ridge_shift	Shifts ridging ice among thickness categories. It removes area, volume, and energy from each ridging category and adds them to thicker ice categories. **I/O:** stdout **Calls:** abort_ice, column_conservation_check, column_sum **Called by:** ridge_ice **I/O Parameters:** real opening- rate of opening due to divergence/sheer real closing_gross- rate at which area is removed, not counting area of new ridges real msnow_mlt- mass of snow added to ocean (kg/m^2) real esnow_mlt- energy needed to melt snow in ocean (J/m^2)
runtime_diags	Writes diagnostic information, such as max, min, global sums, etc., to standard out. **I/O:** stdout, 799, 800, 801, 899, 900, 901, 902, 903 (write) **Calls:** global_gather, ice_global_real_maxval, ice_model_size, ice_flux, ice_albedo, ice_global_real_sum, ice_mpi_internal, ice_state, ice_itd, get_sum. If coupled, shr_sys_mod (only: *shr_sys_flush*) **Called by:** icemodel **I/O Parameters:** None
scale_fluxes	Divides ice fluxes by ice area before sending them to the coupler, since the coupler multiplies by ice area. This is the ice area at the beginning of the timestep, i.e. the value sent to the coupler. **I/O:** None **Calls:** ice_albedo **Called by:** icemodel **I/O Parameters:** None

Subroutine	Description
scale_hist_fluxes	Divides ice fluxes by ice area used by the coupler before writing out diagnostics. *Aice_init* is the ice area saved from coupling. This makes the fluxes written to the history file consistent with those sent to the coupler. **I/O:** None **Calls:** None **Called by:** icemodel **I/O Parameters:** None
setup_mpi	This routine initializes MPI for either internal parallel processing or for message passing with the coupler. **I/O:** stdout **Calls:** abort_ice, ice_mpi_internal, ice_coupling, ice_coupling_setup **Called by:** icemodel **I/O Parameters:** None
shift_ice	Shifts ice across category boundaries, conserving area, volume, and energy. **I/O:** stdout **Calls:** abort_ice, ice_flux, ice_work (only: *worka*) **Called by:** linear_itd, rebin **I/O Parameters:** integer donor- donor category index real daice- ice area transferred across boundary real dvice- ice volume transferred across boundary real hicen- ice thickness for each category (m) NOTE: Third index of *donor, daice, dvice* should be *ncat-1*, except that compilers would have trouble when *ncat = 1.*
sss_clim	Creates an annual mean climatology for Levitus SSS from a 12-month climatology. **I/O:** stdout **Calls:** ice_open, ice_read, ice_work (only: *worka*) **Called by:** init_getflux **I/O Parameters:** None
sss_sst_restore	Interpolates monthly SSS and SST data to timestep. This subroutine restores SST computed by the ice model to data. **I/O:** stdout, readm **Calls:** complete_getflux_ocn, interp_coeff_monthly, interpolate_data, read_clim_data **Called by:** getflux **I/O Parameters:** None

Subroutine	Description
sst_ic	Reads SST data for current month, and adjusts SST based on freezing temperature. This routine does not interpolate. **I/O:** stdout **Calls:** ice_open, ice_read **Called by:** init_getflux **I/O Parameters:** None
stepu	Calculation of the surface stresses and integration of the momentum equation to find velocity (*u,v*). **I/O:** None **Calls:** ice_flux **Called by:** evp **I/O Parameters:** None
stress	Computes the rates of strain and internal stress components for each of the four corners on each T-grid cell. **I/O:** None **Calls:** None **Called by:** evp **I/O Parameters:** integer ksub- subcycling step input)
surface_fluxes	Computes radiative and turbulent fluxes and their derivatives with respect to T_{sf}. **I/O:** None **Calls:** None **Called by:** temperature_changes **I/O Parameters:** integer isolve- no. of cells with temps not converged integer indxii, indxjj- compressed indices for cells not converged real Tsf- ice/snow surface temperature, T_{sfcn} real fswsfc- SW absorbed at ice/ snow surface (W/m^2) real fsensn- surface downward sensible heat (W/m^2) real flatn- surface downward latent heat (W/m^2) real flwoutn- upward LW at surface (W/m^2) real fsurf- net flux to top surface, not incl. *fcondtop* real dfsens_dT- derivative of *fsens* with respect to T_{sf} (W/m^2 /deg) real dflat_dT- deriv of *flat* with respect to T_{sf} (W/m^2 /deg) real dflwout_dT- deriv of *flwout* with respect to T_{sf}(W/m^2 /deg) real dfsurf_dT- derivative of *fsurf* with respect to T_{sf}

Subroutine	Description
t2ugrid	Transfers from T-cell centers to U-cell centers. Writes work into another array that has ghost cells. **I/O:** None **Calls:** bound, to_ugrid **Called by:** evp_prep, from_coupler **I/O Parameters:** real work
temperature_changes	Computes new surface temperature and internal ice and snow temperatures. It includes effects of salinity on sea ice heat capacity in a way that conserves energy [7]. New temperatures are computed iteratively by solving a tridiagonal system of equations. Heat capacity is updated with each iteration. Finite differencing is backward implicit. **I/O:** stdout **Calls:** abort_ice, absorbed_solar, conductivity, print_state, surface_fluxes, tridiag_solver **Called by:** thermo_vertical **I/O Parameters:** integer n- thickness category index integer icells- number of cells with *aicen* > puny integer indxi, indxj- compressed indices for cells with *aicen* > puny real hlyr- ice layer thickness real hsn- snow thickness (m) real Tbot- ice bottom surface temp (° C) real fbot- ice-ocean heat flux at bottom surface (W/m^2) real qsn- snow enthalpy real Tsn- internal snow temperature real Tsf- ice/snow surface temperature, T_{sfcn} real fsensn- surface downward sensible heat (W/m^2) real flatn- surface downward latent heat (W/m^2) real fswabsn- shortwave absorbed by ice (W/m^2) real flwoutn- upward LW at surface (W/m^2) real fswthrun- SW through ice to ocean (W/m^2) real fhnetn- *fbot*, corrected for any surplus energy real fsurf- net flux to top surface, not including *fcondtop* real fcondtop- downward cond flux at top surface (W/m^2) real fcondbot- downward cond flux at bottom surface (W/m^2) real fswint- SW absorbed in ice interior, below surface (W/m^2) real qin- ice layer enthalpy (J/m^3) real Tin- internal ice layer temperatures real einit- initial energy of melting (J/m^2)

Subroutine	Description
thermo_itd	Driver for post-coupler thermodynamic changes not needed for coupling: transport in thickness space, lateral growth and melting, and freeboard adjustment. NOTE: Ocean fluxes are initialized here. **I/O:** None **Calls:** add_new_ice, aggregate, freeboard, ice_timers, ice_itd_linear, ice_therm_vertical, ice_timer_start, ice_timer_stop, init_flux_ocn, lateral_melt, linear_itd, reduce_area, rebin, zap_small_areas **Called by:** icemodel **I/O Parameters:** None
thermo_vertical	Driver for updating ice and snow internal temperatures and computing thermodynamic growth rates and atmospheric fluxes. **I/O:** None **Calls:** add_new_snow, atmo_boundary_layer, conservation_check_vthermo, frzmlt_bottom_lateral, ice_atmo, ice_timers, ice_timer_start, ice_timer_stop, ice_ocean, ice_work (only: *worka, workb*), init_diagnostics, init_flux_atm, init_thermo_vars, init_vertical_profile, merge_fluxes, temperature_changes, thickness_changes, update_state_vthermo **Called by:** icemodel **I/O Parameters:** None

Subroutine	Description
thickness_changes	Computes growth and/or melting at the top and bottom surfaces. **I/O:** None **Calls:** None **Called by:** thermo_vertical **I/O Parameters:** integer n- thickness category index integer icells- number of cells with *aicen* > puny integer indxi, indxj- compressed indices for cells with *aicen* > puny real fbot - ice-ocean heat flux at bottom surface (W/m^2) real Tbot - ice bottom surface temperature (° C) real flatn - surface downward latent heat (W/m^2) real fsurf - net flux to top surface, not including fcondtop real fcondtop- downward cond flux at top surface (W/m^2) real fcondbot- downward cond flux at bottom surface (W/m^2) real qin- ice layer enthalpy (J/m^3) real fhnetn- *fbot*, corrected for any surplus energy real hlyr - ice layer thickness real hsn - snow thickness (m) real qsn - snow enthalpy real efinal- final energy of melting (J/m^2) real mvap - ice/snow mass sublimated/condensed (kg/m^2) real hin- total ice thickness
timers	Does the work. **I/O:** None **Calls:** None **Called by:** ice_timer_start, ice_timer_stop **I/O Parameters:** real t1
tlatlon	Initializes latitude and longitude on T grid. **I/O:** stdout **Calls:** bound, global_gather, global_scatter, ice_model_size, ice_read_write (if reading ULAT, ULON directly from file) **Called by:** init_grid **I/O Parameters:** None
to_coupler	Sends data from PIPS 3.0 model to coupler. **I/O:** stdout, 100 (write) **Calls:** get_sum, ice_timer_start, ice_timer_stop **Called by:** init_cpl **I/O Parameters:** None

Subroutine	Description
to_tgrid	Shifts quantities from the U-cell midpoint (*work1*) to the T-cell midpoint (*work2*). **I/O:** None **Calls:** None **Called by:** u2tgrid **I/O Parameters:** real work1, real work2
to_ugrid	Shifts quantities from the T-cell midpoint (*work1*) to the U-cell midpoint (*work2*). **I/O:** None **Calls:** None **Called by:** evp_prep, t2ugrid **I/O Parameters:** real work1, real work2
transport_mpdata	Computes the transport equations for one timestep using *mpdata*. Sets several fields into a work array and passes it to *mpdata* routine. **I/O:** stdout **Calls:** bound_narr, check_state, ice_flux, ice_timers, ice_state, ice_itd (check_state), ice_timer_start, ice_timer_stop, mpdata **Called by:** icemodel **I/O Parameters:** None
transport_remap	This subroutine solves the transport equations for one timestep using the conservative remapping scheme developed by John Dukowicz and John Baumgardner (DB) and modified for sea ice by William Lipscomb and Elizabeth Hunke. This scheme is second-order accurate, except where gradients are limited to preserve monotonicity. It is compatible for tracers; that is, it does not produce new extrema in thickness or enthalpy. References: [9], [26]. **I/O:** None **Calls:** bound, bound_narr, bound_state, bound_sw, check_monotonicity, conserved_sums, construct_fields, departure_points, flux_integrals, global_conservation, ice_timer_start, ice_timer_stop, load_tracers, local_max_min, locate_triangles, make_masks, unload_tracers, update_fields **Called by:** icemodel **I/O Parameters:** None

Subroutine	Description
triangle_coordinates	For each triangle, this subroutine finds the coordinates of the four points needed to compute integrals of cubic polynomials, using a formula from [35]. (Section 8.8, formula 3.1.) Quadratic functions can be integrated using 3-point formulas, but it is more efficient to use a single formula for both quadratics and cubics. The formula is as follows: $I3$ = integral of f(x,y)*dA = AR * [-9/16 * f($x0,y0$) + 25/48 * (f($x1,y1$) + f($x2,y2$) + f($x3,y3$))] where $I3$ is the integral of a polynomial of 3rd order or lower, AR is the area of the triangle, ($x0,y0$) is the midpoint, and the other three points are located 2/5 of the way from the midpoint to each of the three vertices. **I/O:** None **Calls:** None **Called by:** transport_remap **I/O Parameters:** real triarea- areas of flux triangles real xp0, yp0- coordinates of triangle points xp1, yp1 xp2, yp2 xp3, yp3
tridiag_solver	Tridiagonal matrix solver. It is used to solve the implicit vertical heat equation in ice and snow. **I/O:** None **Calls:** None **Called by:** temperature_changes **I/O Parameters:** integer isolve- number of cells with temps not converged integer indxii, indxjj- compressed indices for cells not converged integer nmat- matrix dimension real diag- diagonal matrix elements real sbdiag- sub-diagonal matrix elements real spdiag- super-diagonal matrix elements real rhs- *rhs* of tri-diagonal matrix equation real xout- solution vector

Subroutine	Description
u2tgrid	Transfers from U-cell centers to T-cell centers. Writes work into another array that has ghost cells. **I/O:** None **Calls:** bound, to_tgrid **Called by:** evp_finish **I/O Parameters:** real work
unload_tracers	Converts from tracer array back to state variables. Like *load_tracers*, this subroutine must be modified if a different set of tracers is to be transported. **I/O:** None **Calls:** None **Called by:** transport_remap **I/O Parameters:** integer n- ice category index real trm- mean tracer values in each grid cell
update_fields	Given fluxes through cell edges, this subroutine computes new area and tracers. **I/O:** None **Calls:** abort_ice **Called by:** transport_remap **I/O Parameters:** real aiflxe, aiflxn- flux of *aice* through east and north cell edges real aim- mean ice area real atflxe, atflxn- flux of *aice**tracer through E and N cell edges real trm- mean tracers integer n- ice category index (for error diagnostics)

Subroutine	Description
update_state_vthermo	Given the vertical thermo state variables (*hin, hsn, qin, qsn, Tsf*), this subroutine computes the new ice state variables (*vicen, vsnon, eicen, esnon, Tsfcn*). State variables are zeroed out if ice has melted entirely. **I/O:** None **Calls:** None **Called by:** thermo_vertical **I/O Parameters:** integer n- thickness category index integer icells- number of cells with *aicen* > puny integer indxi, indxj- compressed indices for cells with *aicen* > puny real hin- ice thickness (m) real hsn- snow thickness (m) real qsn- snow enthalpy real Tsf- ice/snow surface temperature, T_{sfcn} real qin- ice layer enthalpy (J/m^3)
zap_small_areas	Eliminates very small ice areas. **I/O:** stdout **Calls: abort_ice,** ice_flux, ice_calendar (only: *dt*) **Called by:** icemodel, thermo_itd **I/O Parameters:** None

6.0 PIPS 3.0 Primary Variables and Parameters

The following table defines many of the symbols frequently used in the PIPS 3.0 code. Values appearing in this list are either fixed or recommended; most namelist parameters are indicated (*) with their default values. For other namelist options, see Table 2. All quantities in the code are expressed in MKS units (temperatures may take either Celsius or Kelvin units).

Name	Description	Default Values
A		
advection	type of advection algorithm used	'remap'
ahmax	thickness above which ice albedo is constant	0.5 m
aice0	fractional open water area	
aice(n)	total concentration of ice in grid cell (in category *n*)	
aice_init	concentration of ice at beginning of *dt* (for diagnostics)	
ain_min	minimum fractional ice area allowed in each category	
albicei	*near infrared ice albedo for thicker ice	0.36
albicev	*visible ice albedo for thicker ice	0.78
albsnowi	*near infrared, cold snow albedo	0.70
albsnowv	*visible, cold snow albedo	0.98
albocn	ocean albedo	0.06
alpha	floe shape constant for lateral melt	0.66
astar	e-folding scale for participation function	0.05
awtidf	weighting factor for near-ir, diffuse albedo	0.16
awtidr	weighting factor for near-ir, direct albedo	0.31
awtvdf	weighting factor for visible, diffuse albedo	0.24
awtvdr	weighing factor for visible, direct albedo	0.29
ANGLE	for conversions between the ocean grid and lat-lon grids	
ANGLET	ANGLE converted to T-cells	
atm_data_dir	*directory for atmospheric forcing data	
avgsiz	number of fields that may be written to history file	91
C		

Name	Description	Default Values
Cf	ratio of ridging work to PE change in ridging	17.
char_len	length of character variable strings	80
char_len_long	length of longer character variable strings	128
check_step	time step for writing debugging data	
cldf	cloud fraction	
congel	basal ice growth	m
cosw	cosine of the turning angle in water	1.
cp_air	specific heat of air	1005.0 J/kg/K
cp_wv	specific heat of water vapor	1.81 x 10^3 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
cm_to_m	cm to meters conversion	0.01
c<*n*>	real(*n*)	
Cs	fraction of shear energy contributing to ridging	0.5
Cstar	constant in Hibler ice strength formula	20
D		
daidtd	ice area tendency due to dynamics/transport	1/s
daidtt	ice area tendency due to thermodynamics	1/s
dalb_mlt	[see **ice_albedo.F**]	-0.075
dalb_mlti	[see **ice_albedo.F**]	-0.100
dalb_mltv	[see **ice_albedo.F**]	-0.150
dardg1dt	rate of fractional area loss by ridging ice	1/s
dardg2dt	rate of fractional area gain by new ridges	1/s
dvirdgdt	ice volume ridging rate	m/s
dbl_kind	definition of double precision	selected_real_kind (13)
dbug	*write forcing data diagnostics	.false.
Delta	function of strain rates (see Section 5.2.2.4)	
depressT	ratio of freezing temps to salinity of brine	0.054 deg/psu
diag_file	*diagnostic output file (alternative to stdout)	
diag_type	*where diagnostic output is written	stdout

Name	Description	Default Values
diagfreq	*how often diagnostic output is written (10 = once/10 *dt*	24
divu	strain rate I component, velocity divergence	1/s
divu_adv	divergence associated with advection	1/s
dt	*thermo/transport time step	3600.s
dt_dyn	dynamics/transport time step (Δt_{dyn})	
dte	subcycling time step for elastic dynamics (Δt_e)	s
dtei	1/*dte*, where *dte* is the EVP subcycling time step	1/s
dT_mlt	[**see ice_albedo.F**]	1. deg
dump_file	* output file for restart dump	
dumpfreq	* dump frequency for restarts, y, m, or d	y
dumpfreq_n	*restart output frequency	1
dragw	drag coefficient for water on ice*ρ_w	0.00536*rhow kg/m^3
dxt	width of T cell (Δx) through the middle	m
dxu	width of U cell (Δx) through the middle	m
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
E		
ecc	yield curve major/minor axis ratio, squared	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
eps04	a small number	10 $^{-4}$
eps11	a small number	10 $^{-11}$
eps12	a small number	10 $^{-12}$
eps13	a small number	10 $^{-13}$
eps15	a small number	10 $^{-15}$
esno(n)	energy of melting of snow per unit area (in category *n*)	J/m^2
evap	evaporative water flux	kg/m^2/s

Name	Description	Default Values
evp_damping	*if true, use evp damping procedure [17]	F
eyc	coefficient for calculating the parameter E, 0<*eyc*<1	0.36
F		
fcor	Coriolis parameter	1/s
ferrmax	max allowed energy flux error (thermodynamics)	1×10^{-3} W/m^2
fhnet	net heat flux	W/m^2
fhnet_hist	net heat flux to ocean (*fhnet*) for history	W/m^2
flat	latent heat flux	W/m^2
floediam	effective flue diameter for lateral melt	300. m
flw	incoming longwave radiation	W/m^2
flwout	outgoing longwave radiation	W/m^2
frain	rainfall rate	kg/m^2/s
frazil	frazil ice growth	m
fresh	fresh water flux to ocean	kg/m^2/s
fresh_hist	fresh water flux (*fresh*) for history	kg/m^2/s
frzmlt	freezing/melting potential	W/m^2
frz_onset	day of year that freezing begins	
fsalt	net salt flux to ocean	kg/m^2/s
fsalt_hist	salt flux to ocean (*fsalt*) for history	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
fsw	incoming shortwave radiation	W/m^2
fswabs	absorbed shortwave radiation	W/m^2
fswthru	shortwave penetrating to ocean	W/m^2
fswthru_hist	shortwave penetrating to ocean (*fswthru*) for history	W/m^2
fyear	current data year	
fyear_final	last data year	
fyear_init	*initial data year	
G		

Name	Description	Default Values
gravit	gravitational acceleration	9.80616 m/s^2
grid_file	*input file for grid info	
grid_type	*'rectangular' or 'displace_pole' or 'column'	displaced_pole
Gstar	used to compute ridging participation function	0.15
H		
hfrazilmin	minimum thickness of new frazil ice	0.05 m
hi_min	minimum ice thickness for thinnest ice category	0.01 m
hicen	ice thickness in category *n*	m
hin_max	category limits	m
hist_avg	*if true, write averaged data instead of snapshots	T
histfreq	*history output frequency; y, m, w, d or 1	m
history_dir	*path to history output files	
history_file	*output file for history	
hm	land/boundary mask, thickness (T-cell)	
hmix	ocean mixed layer depth	20 m
hsnomin	minimum thickness for which T_s is computed	1. x 10^{-6} 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) of T-cell	m
I		
i0vis	fraction of penetrating visible solar radiation	0.70
icells	number of grid cells with specified property (for vectorization)	
ice_ref_salinity	reference salinity for ice-ocean exchanges	4. psu
iceruf	ice surface roughness	5. x 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)	
ierr	general-use error flag	

Name	Description	Default Values
i(j)hi	last *i(j)* index of physical domain (local)	
i(j)lo	first *i(j)* index of physical domain (*local*)	
ilyr1	index of the top layer in each cat (for *eicen*)	
ilyrn	index of the bottom layer in each cat (for *eicen*)	
i(j)mt_global	number of physical gridpts in *x(y)* direction, local domain	
i(j)mt_local	total no. of gridpoints in *x(y)* direction, local domain	
int_kind	definition of an integer	kind(1)
ip, jp	local processor coordinates for writing 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	
K		
kappav	visible extinction coeff. In ice, wavelength < 700 nm	1.4/m
kappan	visible extinction coeff. In ice, wavelength > 700 nm	17.6/m
kcatbound	*category boundary formula	0
kdyn	*type of dynamics (1 = EVP, 0 = off)	1
kg_to_g	kg to g conversion factor	1000.
kice	thermal conductivity of fresh ice	2.03 W/m/deg
kimin	minimum conductivity of saline ice	W/m/deg
kitd	*type of ITD conversions (1 = delta fxn, 1 = linear remap)	1
kmt_file	*input file for land mask info	
krdg_partic	*ridging participation function	1
krdg_redist	*ridging redistribution function	1
ksmooth	*1 = smooth the ice strength	0
ksno	thermal conductivity of snow	0.30 W/m/deg
kstrength	*ice strength formulation (1 = [35], 0 = [15])	1
L		
l_conservation_check	if true, check conservation	

Name	Description	Default Values
Lfresh	latent heat of melting of fresh ice = *Lsub*=*Lvap*	J/kg
lhcoef	transfer coefficient for latent heat	
log_kind	definition of a logical variable	kind(.true.)
Lsub	latent heat of sublimation for fresh water	2.835 x 10^6 J/kg
Lvap	latent heat of vaporization for fresh water	2.501 x 10^6 J/kg
M		
m_to_cm	meters to cm conversion	100.
m1	constant for lateral melt rate	1.6x10^{-6} m/s deg^{-m2}
m2	constant for lateral melt rate	1.36
m2_to_km2	m^2 to km^2 conversion	1 x10^{-6}
mask_n(s)	northern (southern) hemisphere mask	
master_task	task ID for the controlling processor	
mday	day of the month	
meltb	basal ice melt	m
meltl	lateral ice melt	m
meltt	top ice melt	m
melt_onset	day of year that surface melt begins	
month	the month number	
MPI_COMM_ICE	communicator for ice model internal communications (MPI)	
mps_to_cmpdy	m per s to cm per day conversion	8.64 x 10^6
mps_to_compyr	m per s to cm per year conversion	
mtask	local processor number that writes debugging data	
my_task	task ID for the current processor	
N		
nbr_<*dir*>	processor numbers for the N, S, E, W neighbor processors	
ncat	number of ice categories	5
ndte	*number of subcycles	120
ndyn_dt	*number of dynamics/advection steps under thermo	1
new_day	flag for beginning new day	

Name	Description	Default Values
new_month	flag for beginning new month	
new_week	flag for beginning new week	
new_year	flag for beginning new year	
ngroups	number of groups of flux triangles in remapping	5
nilyr	number of ice layers	4
npt	*total number of time steps (*dt*)	24
ntilay	sum of number of layers in all categories	
ntracer	number of tracers transported in remapping	
nu_diag	unit number for diagnostics output file	6
nu_dump	unit number for dump file for restarting	50
nu_forcing	unit number for forcing data file	49
nu_grid	unit number for grid file	11
nu_kmt	unit number for land mask file	12
nu_nml	unit number for namelist input file	21
nu_restart	unit number for restart input file	50
nu_rst_pointer	unit number for pointer to latest restart file	52
num_ghost_cells	no. of rows of ghost cells surrounding each subdomain	1
nyr	year number	
O		
oceanmixed_file	*data file containing ocean forcing data	
oceanmixed_ice	*if true, use internal ocean mixed layer	T
ocn_data_dir	*directory for ocean forcing data	
omega	angular velocity of Earth	7.292 x 10^{-5} rad/s
one	array of ones which is often useful	1.
opening	rate of ice opening due to divergence and shear	1/s
P		
p001	1/1000	
p01	1/100	
p027	1/36	
p055	1/18	

Name	Description	Default Values
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	½	
p52083	25/48	
p5625m	-9/16	
p6	3/5	
p666	2/3	
pi	π	
pih	π/2	
pi2	2π	
pointer_file	*input file for restarting	
potT	atmospheric potential temperature	K
precip_units	*liquid precipitation data units	
print_global	*if true, print global data	F
print_points	*if true, print point data	F
Pstar	ice strength parameter	$2.75\text{x}10^4$ N/m
puny	a small positive number	$1 \text{ x } 10^{-11}$
Q		
Qa	specific humidity at 10 m	kg/kg
qdp	deep ocean heat flux	W/m ^{2}
qqqice	for saturated specific humidity over ice	$1.16378 \text{ x } 10^7$ kg/m^3
qqqocn	for saturated specific humidity over ocean	$6.275724 \text{ x } 10^6$ kg/m^3
Qref	2 m atmospheric reference specific humidity	kg/kg
R		

Name	Description	Default Values
rad_to_deg	degree-radian conversion	180/π
radius	earth radius	6.37 x 10^6 m
real_kind	definition of single precision real	selected_real_kind(6)
restart	*if true, initialize using restart file instead of defaults	T
restart_dir	*path to restart/dump files	
restore_sst	*restore SST to data	
rhoa	air density	kg/m^3
rhofresh	density of fresh water	1000.0 kg/m^3
rhoi	density of ice	917. kg/m^3
rhos	density of snow	330. kg/m^3
rhow	density of seawater	1026 kg/m^3
rnilyr	real(*nlyr*)	
rside	fraction of ice that melts laterally	
S		
saltmax	max salinity, at ice base	3.2 ppm
sec	seconds elapsed into idate	
secday	number of seconds in a day	86400.
shear	strain rate II component	1/s
shcoef	transfer coefficient for sensible heat	
sig1(2)	principal stress components (diagnostic)	
sinw	sine of the turning angle in water	0.
snoice	snow-ice formation	m
snowpatch	length scale for parameterizing nonuniform snow coverage	0.02 m
spval	special value (generally over land or undefined regions, in place of 0)	$10^3$0
ss_tltx(y)	sea surface slope in the *x(y)* direction	m/m
sss	sea surface salinity	psu
sss_data_type	*source of surface salinity data	
sst	sea surface temperature	C
sst_data_type	*source of surface temperature data	

Name	Description	Default Values
stefan-boltzmann	Stefan-Boltzmann constant	5.67 x 10^{-8} W/m^2K^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
strength	ice strength (pressure)	N/m
stressp	internal ice stress, $\sigma_{11} + \sigma_{22}$	N/m
stressm	internal ice stress, $\sigma_{11} - \sigma_{22}$	N/m
stress12	internal ice stress, σ_{12}	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 in the *x(y)*-dir. (T-cell)	N/ m^2
strtlx(y)	surface stress due to sea surface slope	N/ m^2
swv(n)dr(f)	incoming shortwave radiation, visible (near IR), direct (diffuse)	W/ m^2
T		
Tair	air temperature at 10 m	K
tarea	area of T-cell	m^2
tarean	area of northern hemisphere T-cells	m^2
tarear	1/*tarea*	1/ m^2
tareas	area of southern hemisphere T-cells	m^2
Tf	freezing temperature	C
Tffresh	freezing temp of fresh ice	273.15K
time	total elapsed time	s
time_forc	time of last forcing update	s
Timelt	melting temperature of ice top surface	0. C
tinyarea	puny * *tarea*	m^2
TLAT	latitude of cell center	radians
TLON	longitude of cell center	radians
tmask	land/boundary mask, thickness (T-cell)	
tmass	total mass of ice and snow	kg/m^2

Name	Description	Default Values
Tmin	minimum allowed internal temperature	-100° C
Tref	2m atmospheric reference temperature	K
trestore	*SST restoring time scale	days
Tsfc(n)	temperature of ice/snow top surface (in category *n*)	C
Tsf_errmax	max allowed T_{sfc} error (thermodynamics)	5. x 10^{-4} deg
Tsmelt	melting temperature of snow top surface	0. C
TTTice	for saturated specific humidity over ice	5897.8 K
TTTocn	for saturated specific humidity over ocean	5107.4 K
U		
uarea	area of U-cell	m^2
uarear	1/*uarea*	
u(v)atm	wind velocity, *x(y)*	m/s
ULAT	latitude of U-cell centers	radians
ULON	longitude of U-cell centers	radians
umask	land/boundary mask, velocity (U-cell)	
umin	min wind speed for turbulent fluxes	1. m/s
u(v)ocn	ocean current, *x(y)* direction	m/s
uvel(vvel)	*x(y)*-component of velocity	m/s
uvm	land/boundary mask, velocity (U-cell)	
V		
vice(n)	volume per unit area of ice (in category *n*)	m
vonkar	von Karman constant	0.4
vsno(n)	volume per unit area of snow (in category *n*)	m
W		
week	week of the year	
wind	wind speed	m/s
work_g1	allocatable, dbl_kind work array	
work_g2	allocatable, dbl_kind work array	
work_gr	allocatable, real_kind work array	

Name	Description	Default Values
write_history	if true, write history now	
write_ic	if true, write initial conditions now	
work_l1	(imt_local, jmt_local) work array	
work_l2	(imt_local, jmt_local) work array	
work_a	(ilo:ihi, jlo:jhi) work array	
work_b	(ilo:ihi, jlo:jhi) work array	
write_restart	if 1, write restart now	
Y		
ycycle	*number of years in forcing data cycle	
yday	day of the year	
year_init	*the initial year	
Z		
zlvl	atmospheric level height	m
zref	reference height for stability	10. m
zTrf	reference height for T_{ref}, Q_{ref}	2. m
zvir	gas constant (water vapor)/gas constant (air) -1	0.606

7.0 NOTES

7.1 Acronyms and Abbreviations

Acronym	Definition
CCSM	Community Climate System Model
CICE	Los Alamos Sea-Ice Model
DMSP	Defense Meteorological Satellite Program
DTG	Date-Time-Group
EVP	Elastic-Viscous-Plastic
FNMOC	Fleet Numerical Meteorology and Oceanography Center
HYCOM	HYbrid Coordinate Ocean Model
I/O	Input/Output
ITD	Ice Thickness Distribution model
LANL	Los Alamos National Laboratory
MPDATA	Multidimensional Positive Definite Advection Transport Algorithm
MPI	Message Passing Interface
NAVOCEANO	Naval Oceanographic Office
NCAR	National Center for Atmospheric Research
NCOM	Navy Coastal Ocean Model
NOGAPS	Navy Operational Global Atmospheric Prediction System
NRL	Navy Research Laboratory
PIPS	Polar Ice Prediction System
POP	Parallel Ocean Program model
PSI	Planning Systems, Incorporated
S	Salinity

SDD	Software Design Description
SGI	Silicon Graphics Incorporated
SHEBA	Surface Heat Budget of the Arctic Ocean
SSC	Stennis Space Center
SSM/I	Special Sensor Microwave/Imager
SSS	Sea Surface Salinity
SST	Sea Surface Temperature
SUM	Software Users Manual
T	Temperature

Appendix A

Table of Namelist Options

Name	Type/Options	Description	Default Values / Directory Location
albicei	$0 < \alpha < 1$	near infrared ice albedo for thicker ice	
albicev	$0 < \alpha < 1$	visible ice albedo for thicker ice	
albsnowi	$0 < \alpha < 1$	near infrared, cold snow albedo	
albsnowv	$0 < \alpha < 1$	visible, cold snow albedo	
advection	remap	linear remapping advection	’remap’
	mpdata	2nd order MPDATA	
	upwind	1st order MPDATA	
atm_data_dir	path/	path to atmospheric forcing data directory	
atm_data_type	default	constant values defined in the code	
	ncar	NCAR bulk forcing data	
	LYq	AOMIP/Large-Yeager forcing data	
dbug	true/false	if true, write atm/ocn data diagnostics	.false.
diag_file	filename	diagnostic output file	
diag_type	stdout	write diagnostic output to stdout	‘stdout’ (if uncoupled)
	file	write diagnostic output to file	
diagfreq	integer	frequency of diagnostic output in *dt*	30
	eg., 10	once every 10 time steps	
dt	seconds	thermo/transport time step length	2800.
dump_file	filename prefix	output file for restart dump	’iced’
dumpfreq	y, m, d	write restart every *dumpfreq_n* for years,	'd'

Name	Type/Options	Description	Default Values / Directory Location
		months, days	
dumpfreq_n	integer	frequency restart data is written	
evp_damping	true/false	if true, damp elastic waves [6]	.false.
fyear_init	yyyy	first year of atmospheric forcing data	
f_<*var*>	true/false	write <*var*> to history	
grid_file	filename	name of grid file to be read	‘grid’
grid_type	rectangular, displaced_pole	rectangular: defined in *rectgrid* displaced_pole: read from file in *popgrid* pipsgrid: read from file in *pipsgrid*	‘pipsgrid’
hist_avg	true/false	write time-averaged data if true write snapshots of data if false	.false.
hist_dir	path/	path to history output directory	
histfreq	y, m, w, d, l	write history output once a year, month, week, day, or every time step	'h'
history_file	filename prefix	output file for history	'iceh'
ice_ic	default	latitude and SST dependent	‘default’
	none	no ice	
istep0	integer	initial time step number	0
kcatbound	0/1	if 0, original category boundary formula if 1, new category boundary formula	1

kdyn	0 /1	if 0, EVP dynamics OFF
		if 1, EVP dynamics ON
			1

kitd	0 /1	if 0, delta function ITD approx.
		if 1, linear remapping ITD approx.
			1

kmt_file	filename	name of land mask file to be	'kmt'

Name	Type/Options	Description	Default Values / Directory Location
		read	
krdg_partic	0/1	if 0, old ridging participation function if 1, new ridging participation function	1
krdg_redist	0/1	if 0, old ridging redistribution function if 1, new ridging redistribution function	0

kstrength	0 /1	if 0, [15] ice strength formulation
		if 1, [35] ice strength formulation
			1

ndte	integer	number of EVP subcycles	120
ndyn_dt	integer	number of dynamics/advection/ridging steps per thermo timestep	1
npt	integer	total number of time steps to take	
oceanmixed_file	filename	data file containing ocean forcing data	
oceanmixed_ice	true/false	active ocean mixed layer calculation	.true. (if uncoupled)
ocn_data_dir	path/	path to oceanic forcing data directory	'/scr/posey/pips3c/data_in/'
print_points	true/false	print diagnostic data for two grid points	.true.
precip_units	mm_per_month	liquid precipitation data units	
	mm_per_sec	(default; MKS units)	
print_global	true/false	print diagnostic data, global sums	.true.
pointer_file	pointer filename	contains restart filename	
restart	true/false	initialize using restart file	.true.
restart_dir	path/	path to restart directory	
restore_sst	true/false	restore SST to data	
sss_data_type	default	constant values defined in the code	
	clim	climatological data	

Name	Type/Options	Description	Default Values / Directory Location
	ncar	POP ocean forcing data	
sst_data_type	default	constant values defined in the code	
	clim	climatological data	
	ncar	POP ocean forcing data	
trestore	integer	SST restoring time scale (days)	
ycycle	integer	no. of years in forcing data cycle	
year_init	yyyy	the initial year, if not using restart	