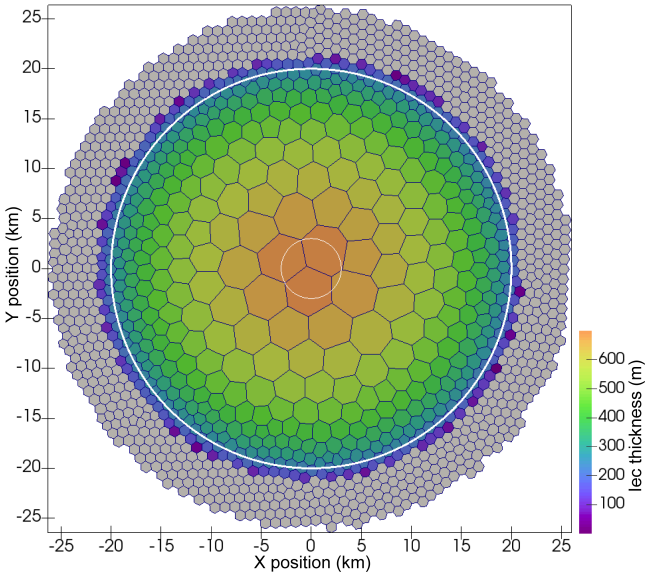


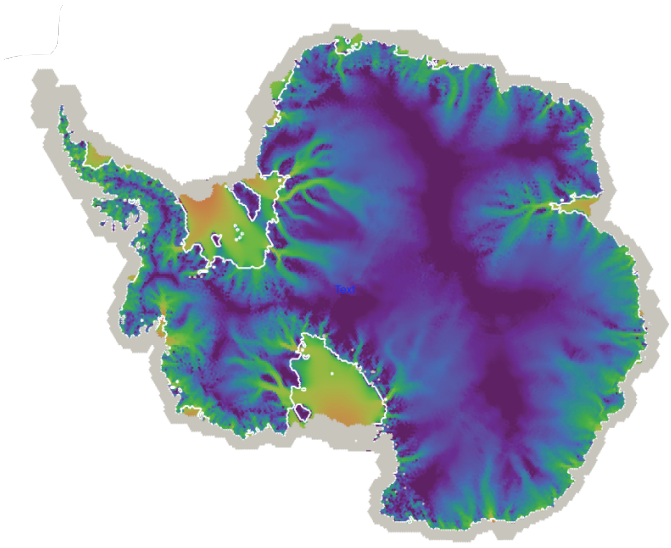
MPAS-Albany Land Ice Model User's Guide

Version: 6.0

Climate, Ocean, Sea-Ice Modeling Team

Los Alamos National Laboratory

April 17, 2018



Foreword

The MPAS-Albany Land Ice (MALI) is an unstructured-mesh land ice model (ice sheets or glaciers) capable of using enhanced horizontal resolution in selected regions of the land ice domain. MALI is built using the Model for Prediction Across Scales (MPAS) framework for developing variable resolution Earth System Model components and the Albany multi-physics code base for solution of coupled systems of partial-differential equations, which itself makes use of Trilinos solver libraries. MALI includes a three-dimensional, first-order momentum balance solver ("Blatter-Pattyn") by linking to the Albany-LI ice sheet velocity solver, as well as an explicit shallow ice velocity solver. Evolution of ice geometry and tracers is handled through an explicit first-order horizontal advection scheme with vertical remapping. Evolution of ice temperature is treated using operator splitting of vertical diffusion and horizontal advection and can be configured to use either a temperature or enthalpy formulation. MALI includes a mass-conserving subglacial hydrology model that supports distributed and/or channelized drainage and can optionally be coupled to ice dynamics. Options for calving include "eigencalving", which assumes calving rate is proportional to extensional strain rates. MALI has been evaluated against commonly used exact solutions and community benchmark experiments and shows the expected accuracy. It has been used in land ice evolution experiments estimating potential for future sea-level rise from ice sheets (e.g., Ice2Sea international assessment project (Shannon et al., 2013; Edwards et al., 2014b)). MALI is the glacier component of the Energy Exascale Earth System Model (E3SM) version 1.0 (https://climatemodeling.science.energy.gov/projects/energy-exascale-earth-system-model).

MPAS-Albany Land Ice is one component within the Model for Prediction Across Scales (MPAS) framework of climate model components that is developed in cooperation between Los Alamos National Laboratory (LANL) and the National Center for Atmospheric Research (NCAR). Functionality that is required by all cores, such as i/o, time management, block decomposition, etc, is developed collaboratively, and this code is shared across cores within the same repository. Each core then solves its own differential equations and physical parameterizations within this framework. This user's guide reflects the spirit of this collaborative process, where Part I, "The MPAS Framework", applies to all cores, and the remaining parts apply specifically to MPAS-Albany Land Ice.

MPAS-Albany Land Ice also makes use of the Albany/LI velocity solver (formally Albany/FELIX) for implementation of the first-order velocity solver. Not all details of running and configuring that velocity solver are covered in this User's Guide. We refer the user to Albany's website for more information about installing and running Albany: https://github.com/gahansen/Albany

MPAS-Albany Land Ice is described by a model description paper currently in review in Geoscientific Model Descriptions at: https://www.geosci-model-dev-discuss.net/gmd-2018-78/ Much of the information in this User's Guide is derived from the text of that paper.

0.1 Support Policy

The Department of Energy support for the MALI model fully realizes and embraces the importance of making the model source code, the data and the application software tools publicly available, and of communicating and informing the scientific community and the public about all stages of the project, its research and future plans. While MALI has become an open development project, we cannot commit ourselves to increased support to cover developmental versions. We are committed though to provide limited support for the scientifically validated configurations of the model.

0.2 Release History

A history of releases of the Land Ice core within the MPAS version numbering scheme is as follows:

version	date	description
6.0	April 17, 2018	Addition of Albany FO velocity solver, thermal solver, subglacial hydrology model, calving, analysis members, coupling to E3SM.
3.0	November 18, 2014	Fix bug in SIA slope calculation. Introduction of run-time I/O streams.
2.0	November 15, 2013	Initial public release of Land Ice core (SIA velocity solver only)

0.3 Additional Information

Information about MPAS-Albany Land Ice, including the most recent code, user's guide, and test cases, may be found at http://mpas-dev.github.com. This user's guide refers to version 6.0.

Contributors to this guide:
Matt Hoffman, Stephen Price, Mauro Perego
Additional contributors to MPAS Framework sections:
Michael Duda, Douglas Jacobsen

Funding for the development of MPAS-Albany Land Ice was provided by the United States Department of Energy, Office of Science.

Contents

Chapter 1

MPAS-Land Ice Quick Start Guide

This chapter provides MPAS-Land Ice users with a quick start description of how to build and run the model. It is meant merely as a brief overview of the process, while the more detailed descriptions of each step are provided in later sections.

In general, the build process follows the following steps.

1. Build MPI Layer (OpenMPI, MVAPICH2, etc.)
2. Build serial NetCDF library (v3.6.3, v4.1.3, etc.)
3. Build Parallel-NetCDF library (v1.2.1, v1.3.0, etc.)
4. Build Parallel I/O library (v1.4.1, v1.6.1, etc.)
5. (Optional) Build METIS library and executables (v4.0, v5.0.2, etc.)
6. Clone MPAS-Land Ice from repository
7. Build Land Ice core

After step 7, an executable should be created called `landice_model.exe`. Once the executable is built, one can begin the run process as follows:

1. Create run directory.
2. Copy executable to run directory.
3. Copy `namelist.landice` and `streams.landice` from the `default_inputs` directory into run directory.
4. (Optional) Copy input and graph files into run directory.
5. Edit `namelist.input` to have the proper parameters.
   If step 4 was skipped, ensure paths to input and graph files are appropriately set.
6. (Optional) Create graph files, using METIS executable (pmetis or gpmetis depending on version).
   A graph file is required for each processor count you want to use greater than one processor.
7. Run MPAS-Land Ice.
8. Visualize output file, and perform analysis.

Part I

The MPAS Framework

Chapter 2

MPAS Framework Overview

The MPAS Framework provides the foundation for a generalized geophysical fluid dynamics model on unstructured spherical and planar meshes. On top of the framework, implementations specific to the modeling of a particular physical system (e.g., land ice, ocean) are created as MPAS *cores*. To date, MPAS cores for atmosphere (Skamarock et al., 2012), ocean (Ringler et al., 2013; Petersen et al., 2015, 2018), shallow water (Ringler et al., 2011), sea ice (Turner et al., 2018), and land ice (Hoffman et al., 2018) have been implemented. The MPAS design philosophy is to leverage the efforts of developers from the various MPAS cores to provide common framework functionality with minimal effort, allowing MPAS core developers to focus on development of the physics and features relevant to their application.

The framework code includes shared modules for fundamental model operation. Significant capabilities include:

- *Description of model data types.* MPAS uses a handful of fundamental Fortran derived types for basic model functionality. Core-specific model variables are handled through custom groupings of model fields called *pools*, for which custom accessor routines exist. Core-specific variables are easily defined in XML syntax in a *Registry*, and the framework parses the Registry, defines variables, and allocates memory as needed.

- *Description of the mesh specification.* MPAS requires 36 fields to fully describe the mesh used in a simulation. These include the position, area, orientation, and connectivity of all cells, edges, and vertices in the mesh. The mesh specification can flexibly describe both spherical and planar meshes. More details are provided in the next section.

- *Distributed memory parallelization and domain decomposition.* The MPAS Framework provides needed routines for exchanging information between processors in a parallel environment using Message Passing Interface (MPI). This includes halo updates, global reductions, and global broadcasts. MPAS also supports decomposing multiple domain blocks on each processor to, for example, optimize model performance by minimizing transfer of data from disk to memory. Shared memory parallelization through OpenMP is also supported, but the implementation is left up to each core.

- *Parallel input and output capabilities.* MPAS performs parallel input and output of data from and to disk through the commonly used libraries of NetCDF, Parallel NetCDF (pnetcdf), and Parallel Input/Output (PIO) (Dennis et al., 2012). The Registry definitions control which fields can be input and/or output, and a framework *streams* functionality provides easy runtime configuration of what fields are to be written to what file name and at what frequency

through an XML streams file. The MPAS framework includes additional functionality specific to providing a flexible model restart capability.

- *Advanced timekeeping.* MPAS uses a customized version of the timekeeping functionality of the Earth System Modeling Framework (ESMF), which includes a robust set of time and calendar tools used by many Earth System Models (ESMs). This allows explicit definition of model epochs in terms of years, months, days, hours, minutes, seconds, and fractional seconds and can be set to three different calendar types: Gregorian, Gregorian no leap, and 360 day. This flexibility helps enable multi-scale physics and simplifies coupling to ESMs. To manage the complex date/time types that ensue, MPAS framework provides routines for arithmetic of time intervals and the definition of alarm objects for handling events (e.g., when to write output, when the simulation should end).

- *Run-time configurable control of model options.* Model options are configured through *namelist* files that use standard Fortran namelist file format, and input/output are configured through *streams* files that use XML format. Both are completely adjustable at run time.

- *Online, run-time analysis framework.* A system for defining analysis of model states during run time, reducing the need for post-processing and model output.

Additionally, a number of shared operators exist to perform common operations on model data. These include geometric operations (e.g., length, area, and angle operations on the sphere or the plane), interpolation (linear, barycentric, Wachspress, radial basis functions, spline), vector and tensor operations (e.g., cross products, divergence), and vector reconstruction (e.g., interpolating from cell edges to cell centers). Most operators work on both spherical and planar meshes.

Chapter 3

Building MPAS

3.1 Prequisites

To build MPAS, compatible C and Fortran compilers are required. Additionally, the MPAS software relies on the PIO parallel I/O library to read and write model fields, and the PIO library requires the standard netCDF library as well as the parallel-netCDF library from Argonne National Labs. All libraries must be compiled with the same compilers that will be used to build MPAS. Section 3.2 summarizes the basic procedure of installing the required I/O libraries for MPAS.

In order for the MPAS makefiles to find the PIO, parallel-netCDF, and netCDF include files and libraries, the environment variables `PIO`, `PNETCDF`, and `NETCDF` should be set to the root installation directories of the PIO, parallel-netCDF, and netCDF installations, respectively. Newer versions of the netCDF library use a separate Fortran interface library; the top-level MPAS Makefile attempts to add `-lnetcdff` to the linker flags, but some linkers require that `-lnetcdff` appear before `-lnetcdf`, in which case `-lnetcdff` will need to be manually added just before `-lnetcdf` in the specification of `LIBS` in the top-level Makefile.

An MPI installation such as MPICH or OpenMPI is also required, and there is no option to build a serial version of the MPAS executables. There is currently no support for shared-memory parallelism with OpenMP within the MPAS framework.

3.2 Compiling I/O Libraries

NOTE: It's important to note the MPAS Developers are not responsible for any of the libraries that are used within MPAS. Support for specific libraries should be taken up with the respective developer groups.

Although most recent versions of the I/O libraries should work, the most tested versions of these libraries are: netCDF 4.1.3, parallel-netCDF 1.3.1, and PIO 1.4.1. The netCDF and parallel-netCDF libraries must be installed before building PIO library.

All commands are presented for csh, and will not work if pasted into another shell. Please translate them to the appropraite commands in your shell.

3.2.1 netCDF

Version 4.1.3 of the netCDF library may be downloaded from http://www.unidata.ucar.edu/downloads/netcdf/netcdf-4_1_3/index.jsp. Assuming the gfortran and gcc compilers will be used, the following shell commands are generally sufficient to install netCDF.

```
> setenv FC gfortran
> setenv F77 gfortran
> setenv F90 gfortran
> setenv CC gcc
> ./configure --prefix=XXXXX --disable-dap --disable-netcdf-4 --disable-cxx
--disable-shared --enable-fortran
> make all check
> make install
```

Here, `XXXXX` should be replaced with the directory that will serve as the root installation directory for netCDF. *Before proceeding to compile PIO the* `NETCDF_PATH` *environment variable should be set to the netCDF root installation directory.*

Certain compilers require addition flags in the CPPFLAGS environment variable. Please refer to the netCDF installation instructions for these flags.

3.2.2 parallel-netCDF

Version 1.3.1 of the parallel-netCDF library may be downloaded from https://trac.mcs.anl.gov/projects/parallel-netcdf/wiki/Download. Assuming the gfortran and gcc compilers will be used, the following shell commands are generally sufficient to install parallel-netCDF.

```
> setenv MPIF90 mpif90
> setenv MPIF77 mpif90
> setenv MPICC mpicc
> ./configure --prefix=XXXXX
> make
> make install
```

Here, `XXXXX` should be replaced with the directory that will serve as the root installation directory for parallel-netCDF. *Before proceeding to compile PIO the* `PNETCDF_PATH` *environment variable should be set to the parallel-netCDF root installation directory.*

3.2.3 PIO

Instructions for building PIO can be found at http://www.cesm.ucar.edu/models/pio/. Please refer to these instructions for building PIO.

After PIO is built, and installed the PIO enviroment variable needs to be defined to point at the directory PIO is installed into. Older versions of PIO cannot be installed, and the PIO environment variable needs to be set to the directory where PIO was built instead.

3.3 Compiling MPAS

Before compiling MPAS, the `NETCDF`***,*** `PNETCDF`***, and*** `PIO` ***environment variables must be set to the library installation directories as described in the previous section. A*** `CORE` ***variable also needs to either be defined or passed in during the make process. If*** `CORE` ***is not specified, the build process will fail.***

The MPAS code uses only the 'make' utility for compilation. Rather than employing a separate configuration step before building the code, all information about compilers, compiler flags, etc.,

is contained in the top-level `Makefile`; each supported combination of compilers (i.e., a configuration) is included in the `Makefile` as a separate make target, and the user selects among these configurations by running `make` with the name of a build target specified on the command-line, e.g.,

```
> make gfortran
```

to build the code using the GNU Fortran and C compilers. Some of the available targets are listed in the table below, and additional targets can be added by simply editing the `Makefile` in the top-level directory.

Target	Fortran compiler	C compiler	MPI wrappers
`xlf`	xlf90	xlc	mpxlf90 / mpcc
`pgi`	pgf90	pgcc	mpif90 / mpicc
`ifort`	ifort	gcc	mpif90 / mpicc
`gfortran`	gfortran	gcc	mpif90 / mpicc
`g95`	g95	gcc	mpif90 / mpicc

In order to get a more complete and up-to-date list of available tagets, one can use the following command within the top-level of MPAS. **NOTE:** This command is known to not work with Mac OSX.

```
> make -rpn | sed -n -e '/^$/ { n ; /^[^ ]*:/p }' | sed "s/: *.*$//g"
```

The MPAS framework supports multiple *cores* — currently a shallow water model, an ocean model, a non-hydrostatic atmosphere model, a non-hydrostatic atmosphere initialization core, and a land ice core — so the build process must be told which core to build. This is done by either setting the environment variable `CORE` to the name of the model core to build, or by specifying the core to be built explicitly on the command-line when running `make`. For the shallow water core, for example, one may run either

```
> setenv CORE sw
> make gfortran
```

or

```
> make gfortran CORE=sw
```

If the `CORE` environment variable is set and a core is specified on the command-line, the command-line value takes precedence; if no core is specified, either on the command line or via the `CORE` environment variable, the build process will stop with an error message stating such. Assuming compilation is successful, the model executable, named `${CORE}_model` (e.g., `sw_model`), should be created in the top-level MPAS directory.

In order to get a list of available cores, one can simply run the top-level `Makefile` without setting the `CORE` environment variable, or passing the core via the command-line. And example of the output from this can be seen below.

```
> make
```

```
( make error )
make[1]: Entering directory ‘mpas’

Usage: make target CORE=[core] [options]

Example targets:
ifort
gfortran
xlf
pgi

Availabe Cores:
atmosphere
init_atmosphere
landice
ocean
sw

Available Options:
DEBUG=true    - builds debug version. Default is optimized version.
USE_PAPI=true - builds version using PAPI for timers. Default is off.
TAU=true      - builds version using TAU hooks for profiling. Default is off.

Ensure that NETCDF, PNETCDF, PIO, and PAPI (if USE_PAPI=true) are environment variables
that point to the absolute paths for the libraries.

************ ERROR ************
No CORE specified. Quitting.
************ ERROR ************

make[1]: Leaving directory ‘mpas’
```

3.4 Cleaning

To remove all files that were created when the model was built, including the model executable itself, `make` may be run for the 'clean' target:

```
> make clean
```

As with compiling, the core to be cleaned is specified by the `CORE` environment variable, or by specifying a core explicitly on the command-line with `CORE=`.

3.5 Graph partitioning with METIS

Before MPAS can be run in parallel, a mesh decomposition file with an appropriate number of partitions (equal to the number of MPI tasks that will be used) is required in the run directory. A limited number of mesh decomposition files (`graph.info.part.*`) are provided with each test case. In order to create new mesh decomposition files for your desired number of partitions, begin with the provided `graph.info` file and partition with METIS software (http://glaros.dtc.umn.edu/gkhome/views/metis). The serial graph partitioning program, METIS (rather than ParMETIS or

hMETIS) should be sufficient for quickly partitioning any SCVT produced by the grid_gen mesh generator.

After installing METIS, a `graph.info` file may be partitioned into N partitions by running

```
> gpmetis graph.info N
```

The resulting file, `graph.info.part.`N, can then be copied into the MPAS run directory before running the model with N MPI tasks.

Chapter 4

Grid Description

This chapter provides a brief introduction to the common types of grids used in the MPAS framework.

The MPAS grid system requires the definition of seven elements. These seven elements are composed of two types of *cells*, two types of *lines*, and three types of *points*. These elements are depicted in Figure 4.1 and defined in Table 4.1. These elements can be defined on either the plane or the surface of the sphere. The two types of cells form two meshes, a primal mesh composed of Voronoi regions and a dual mesh composed of Delaunay triangles. Each corner of a primal mesh cell is uniquely associated with the "center" of a dual mesh cell and vice versa. So we define the two mesh as either a primal mesh (composed of cells P_i) or a dual mesh (composed of cells D_v). The center of any primal mesh cell, P_i, is denoted by $\mathbf{x}_i$ and the center of any the dual mesh cell, D_v, is denoted by $\mathbf{x}_v$. The boundary of a given primal mesh cell P_i is composed of the set of lines that connect the $\mathbf{x}_v$ locations of associated dual mesh cells D_v. Similarly, the boundary of a given dual mesh cell D_v is composed of the set of lines that connect the $\mathbf{x}_i$ locations of the associated primal mesh cells P_i.

As shown in Figure 4.1, a line segment that connects two primal mesh cell centers is uniquely associated with a line segment that connects two dual mesh cell centers. We assume that these two line segments cross and the point of intersection is labeled as $\mathbf{x}_e$. In addition, we assume that these two line segments are orthogonal as indicated in Figure 4.1. Each $\mathbf{x}_e$ is associated with two distances: d_e measures the distance between the primal mesh cells sharing $\mathbf{x}_e$ and l_e measures the distance between the dual mesh cells sharing $\mathbf{x}_e$.

Since the two line segments crossing at $\mathbf{x}_e$ are orthogonal, these line segments form a convenient local coordinate system for each edge. At each $\mathbf{x}_e$ location a unit vector $\mathbf{n}_e$ is defined to be parallel to the line connecting primal mesh cells. A second unit vector $\mathbf{t}_e$ is defined such that $\mathbf{t}_e = \mathbf{k} \times \mathbf{n}_e$.

In addition to these seven element types, we require the definition of *sets of elements*. In all, eight different types of sets are required and these are defined and explained in Table 4.2 and Figure 4.2. The notation is always of the form of, for example, $i \in CE(e)$, where the LHS indicates the type of element to be gathered (cells) based on the RHS relation to another type of element (edges).

Table 4.3 provides the names of all *elements* and all *sets of elements* as used in the MPAS framework. Elements appear twice in the table when described in the grid file in more than one way, e.g. points are described with both cartesian and latitude/longitude coordinates. An "ncdump -h" of any MPAS grid, output or restart file will contain all variable names shown in second column of Table 4.3.

Table 4.1: Definition of elements used to build the MPAS grid.

Element	*Type*	*Definition*
$\mathbf{x}_i$	point	location of center of primal-mesh cells
$\mathbf{x}_v$	point	location of center of dual-mesh cells
$\mathbf{x}_e$	point	location of edge points where velocity is defined
d_e	line segment	distance between neighboring $\mathbf{x}_i$ locations
l_e	line segment	distance between neighboring $\mathbf{x}_v$ locations
P_i	cell	a cell on the primal-mesh
D_v	cell	a cell on the dual-mesh

Table 4.2: Definition of element groups used to reference connections in the MPAS grid. Examples are provided in Figure 4.2.

Syntax	*ouptut*
$e \in EC(i)$	set of edges that define the boundary of P_i.
$e \in EV(v)$	set of edges that define the boundary of D_v.
$i \in CE(e)$	two primal-mesh cells that share edge e.
$i \in CV(v)$	set of primal-mesh cells that form the vertices of dual mesh cell D_v.
$v \in VE(e)$	the two dual-mesh cells that share edge e.
$v \in VI(i)$	the set of dual-mesh cells that form the vertices of primal-mesh cell P_i.
$e \in ECP(e)$	edges of cell pair meeting at edge e.
$e \in EVC(v,i)$	edge pair associated with vertex v and mesh cell i.

Table 4.3: Variable names used to describe a MPAS grid.

Element	*Name*	*Size*	*Comment*
$\mathbf{x}_i$	{x,y,z}Cell	nCells	cartesian location of $\mathbf{x}_i$
$\mathbf{x}_i$	{lon,lat}Cell	nCells	longitude and latitude of $\mathbf{x}_i$
$\mathbf{x}_v$	{x,y,z}Vertex	nVertices	cartesian location of $\mathbf{x}_v$
$\mathbf{x}_v$	{lon,lat}Vertex	nVertices	longitude and latitude of $\mathbf{x}_v$
$\mathbf{x}_e$	{x,y,z}Edge	nEdges	cartesian location of $\mathbf{x}_e$
$\mathbf{x}_e$	{lon,lat}Edge	nEdges	longitude and latitude of $\mathbf{x}_e$
d_e	dcEdge	nEdges	distance between $\mathbf{x}_i$ locations
l_e	dvEdge	nEdges	distance between $\mathbf{x}_v$ locations
$e \in EC(i)$	edgesOnCell	(nEdgesMax,nCells)	edges that define P_i.
$e \in EV(v)$	edgesOnVertex	(3,nCells)	edges that define D_v.
$i \in CE(e)$	cellsOnEdge	(2,nEdges)	primal-mesh cells that share edge e.
$i \in CV(v)$	cellsOnVertex	(3,nVertices)	primal-mesh cells that define D_v.
$v \in VE(e)$	verticesOnEdge	(2,nEdges)	dual-mesh cells that share edge e.
$v \in VI(i)$	verticesOnCell	(nEdgesMax,nCells)	vertices that define P_i.

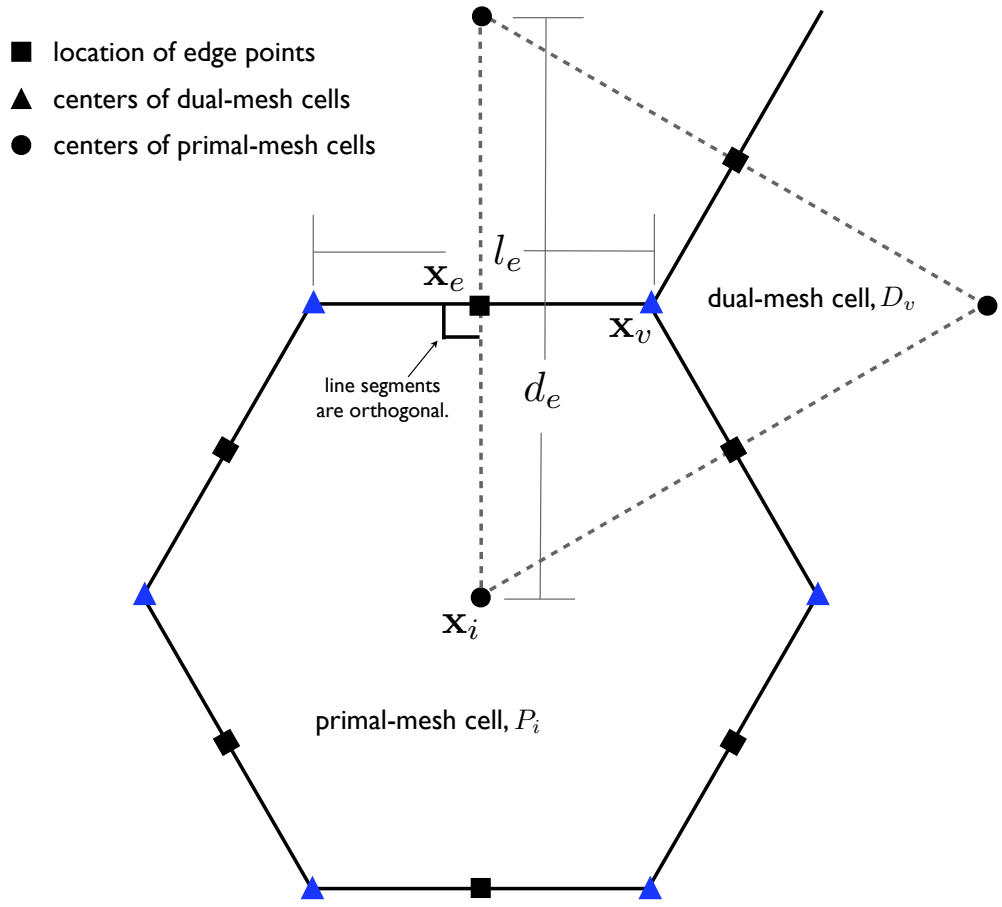


Figure 4.1: Definition of elements used to build the MPAS grid. Also see Table 4.1.

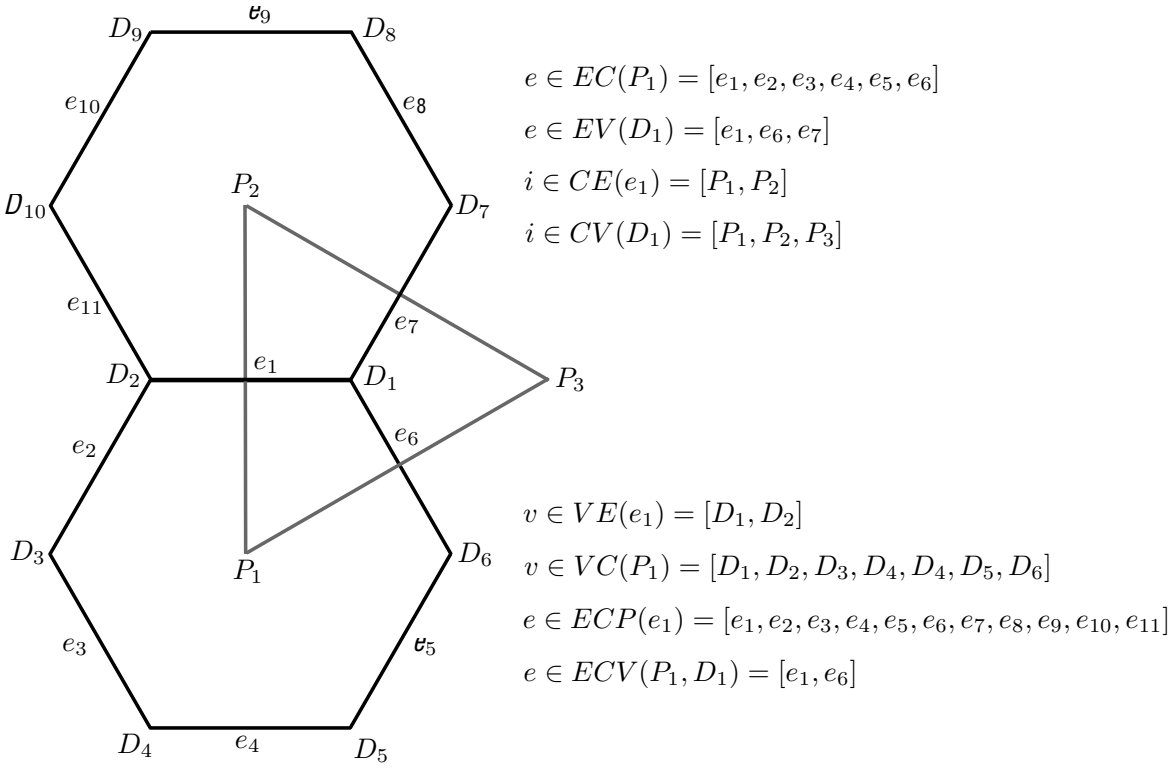


Figure 4.2: Definition of element groups used to reference connections in the MPAS grid. Also see Table 4.2.

Chapter 5

Configuring Model Input and Output

The reading and writing of model fields in MPAS is handled by user-configurable *streams*. A stream represents a fixed set of model fields, together with dimensions and attributes, that are all written or read together to or from the same file or set of files. Each MPAS model core may define its own set of default streams that it typically uses for reading initial conditions, for writing and reading restart fields, and for writing additional model history fields. Besides these default streams, users may define new streams to, e.g., write certain diagnostic fields at a higher temporal frequency than the usual model history fields.

Streams are defined in XML configuration files that are created at build time for each model core. The name of this XML file is simply 'streams.' suffixed with the name of the core. For example, the streams for the *sw* (shallow-water) core are defined in a file named 'streams.sw'. An XML stream file may further reference other text files that contain lists of the model fields that are read or written in each of the streams defined in the XML stream file.

Changes to the XML stream configuration file will take effect the next time an MPAS core is run; there is no need to re-compile after making modifications to the XML files. As described in the next section, it is therefore possible, e.g., to change the interval at which a stream is written, the template for the filenames associated with a stream, or the set of fields that are written to a stream, without the need to re-compile any code.

Two classes of streams exist in MPAS: *immutable* streams and *mutable* streams. Immutable streams are those for which the set of fields that belong to the stream may not be modified at model run-time; however, it is possible to modify the interval at which the stream is read or written, the filename template describing the files containing the stream on disk, and several other parameters of the stream. In contrast, all aspects of mutable streams, including the set of fields that belong to the stream, may be modified at run-time. The motivation for the creation of two stream classes is the idea that an MPAS core may not function correctly if certain fields are not read in upon model start-up or written to restart files, and it is therefore not reasonable for users to modify this set of required fields at run-time. An MPAS core developer may choose to implement such streams as immutable streams. Since fields may not be added to an immutable stream at run-time, new immutable streams may not be defined at run-time, and the only type of new stream that may be defined at run-time is the mutable stream type.

5.1 XML stream configuration files

The XML stream configuration file for an MPAS core always has a parent XML *element* named `streams`, within which individual streams are defined:

```
<streams>

    ... one or more stream definitions ...

</streams>
```

Immutable streams are defined with the `immutable_stream` element, and mutable streams are defined with the `stream` element:

```
<immutable_stream name="initial_conditions"
                  type="input"
                  filename_template="init.nc"
                  input_interval="initial_only"
                  />


<stream name="history"
        type="output"
        filename_template="output.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

    ... model fields belonging to this stream ...

</stream>
```

As shown in the example stream definitions, above, both classes of stream have the following required attributes:

- `name` — A unique name used to refer to the stream.
- `type` — The type of stream, either `"input"`, `"output"`, `"input;output"`, or `"none"`. A stream may be both an input and an output stream (i.e., `"input;output"`) if, for example, it is read once at model start-up to provide initial conditions and thereafter written periodically to provide model checkpoints. A stream may be defined as neither input nor output (i.e., `"none"`) for the purposes of defining a set of fields for inclusion other streams. Note that, for immutable streams, the type attribute may not be changed at run-time.
- `filename_template` — The template for files that exist or will be created by the stream. The filename template may include any of the following variables, which are expanded based on the simulated time at which files are first created.
  - `$Y` — Year
  - `$M` — Month
  - `$D` — Day of the month
  - `$d` — Day of the year
  - `$h` — Hour

  - `$m` — Minute
  - `$s` — Second

  A filename template may include either a relative or an absolute path, in which case MPAS will attempt to create any directories in the path that do not exist, subject to filesystem permissions.

- `input_interval` — For streams that have type `"input"` or `"input;output"`, the interval, beginning at the model initial time, at which the stream will be read. Possible values include a time interval specification in the format `"YYYY-MM-DD_hh:mm:ss"`; the value `"initial_only"`, which specifies that the stream is read only once at the model initial time; or the value `"none"`, which specifies that the stream is not read during a model run.

- `output_interval` — For streams that have type `"output"` or `"input;output"`, the interval, beginning at the model initial time, at which the stream will be written. Possible values include a time interval specification in the format `"YYYY-MM-DD_hh:mm:ss"`; the value `"initial_only"`, which specifies that the stream is written only once at the model initial time; or the value `"none"`, which specifies that the stream is not written during a model run.

Finally, the set of fields that belong to a mutable stream may be specified with any combination of the following elements. Note that, for immutable streams, no fields are specified at run-time in the XML configuration file.

- `var` — Associates the specified variable with the stream. The variable may be any of those defined in an MPAS core's Registry.xml file, but may not include individual constituent arrays from a var_array.

- `var_array` — Associates all constituent variables in a var_array, defined in an MPAS core's Registry.xml file, with the stream.

- `var_struct` — Associates all variables in a var_struct, defined in an MPAS core's Registry.xml file, with the stream.

- `stream` — Associates all explicitly associated fields in the specified stream with the stream; streams are not recursively included.

- `file` — Associates all variables listed in the specified text file, with one field per line, with the stream.

5.2 Optional stream attributes

Besides the required attributes described in the preceding section, several additional, optional attributes may be added to the definition of a stream.

- `filename_interval` — The interval between the timestamps used in the construction of the names of files associated with a stream. Possible values include a time interval specification in the format `"YYYY-MM-DD_hh:mm:ss"`; the value `"none"`, indicating that only one file containing all times is associated with the stream; the value `"input_interval"` that, for input type streams, indicates that each time to be read from the stream will come from a unique file; or the value `"output_interval"` that, for output type streams, indicates that each time to

be written to the stream will go to a unique file whose name is based on the timestamp of the data being written. The default value is `"input_interval"` for input type streams and `"output_interval"` for output type streams. For streams of type `"input;output"`, the default filename interval is `"input_interval"` if the input interval is an interval (i.e., not `"initial_only"`), or `"output_interval"` otherwise. Refer to Section 5.3.1 for an example of the use of the filename_interval attribute.

- `reference_time` — A time that is an integral number of filename intervals from the timestamp of any file associated with the stream. The default value is the start time of the model simulation. Refer to Section 5.3.3 for an example of the use of the reference_time attribute.

- `clobber_mode` — Specifies how a stream should handle attempts to write to a file that already exists. Possible values for the mode include:

  - `"overwrite"` — The stream is allowed to overwrite records in existing files and to append new records to existing files; records not explicitly written to are left untouched.
  - `"truncate"` or `"replace_files"` — The stream is allowed to overwrite existing files, which are first truncated to remove any existing records; this is equivalent to replacing any existing files with newly created files of the same name.
  - `"append"` — The stream is only allowed to append new records to existing files; existing records may not be overwritten.
  - `"never_modify"` — The stream is not allowed to modify existing files in any way.

  The default clobber mode for streams is `"never_modify"`. Refer to Section 5.3.2 for an example of the use of the clobber_mode attribute.

- `precision` — The precision with which real-valued fields will be written or read in a stream. Possible values include `"single"` for 4-byte real values, `"double"` for 8-byte real values, or `"native"`, which specifies that real-valued fields will be written or read in whatever precision the MPAS core was compiled. The default value is `"native"`. Refer to Section 5.3.1 for an example of the use of the precision attribute.

- `packages` — A list of packages attached to the stream. A stream will be active (i.e., read or written) only if at least one of the packages attached to it is active, or if no packages at all are attached. Package names are provided as a semi-colon-separated list. Note that packages may only be defined in an MPAS core's Registry.xml file at build time. By default, no packages are attached to a stream.

- `io_type` — The underlying library and file format that will be used to read or write a stream. Possible values include:

  - `"pnetcdf"` — Read/write the stream with classic large-file NetCDF files (CDF-2) using the ANL Parallel-NetCDF library.
  - `"pnetcdf,cdf5"` — Read/write the stream with large-variable files (CDF-5) using the ANL Parallel-NetCDF library.
  - `"netcdf"` — Read/write the stream with classic large-file NetCDF files (CDF-2) using the Unidata serial NetCDF library.
  - `"netcdf4"` — Read/write the stream with HDF-5 files using the Unidata parallel NetCDF-4 library.

Note that the PIO library must have been built with support for the selected `io_type`. By default, all input and output streams are read and written using the `"pnetcdf"` option.

5.3 Stream definition examples

This section provides several example streams that make use of the optional stream attributes described in Section 5.2. All examples are of output streams, since it is more likely that a user will need to write additional fields than to read additional fields, which a model would need to be aware of; however, the concepts that are illustrated here translate directly to input streams as well.

5.3.1 Example: a single-precision output stream with one month of data per file

In this example, the optional attribute specification `filename_interval="01-00_00:00:00"` is added to force a new output file to be created for the stream every month. Note that the general format for time interval specifications is `YYYY-MM-DD_hh:mm:ss`, where any leading terms can be omitted; in this case, the year part of the interval is omitted. To reduce the file size, the specification `precision="single"` is also added to force real-valued fields to be written as 4-byte floating-point values, rather than the default of 8 bytes.

```
<stream name="diagnostics"
        type="output"
        filename_template="diagnostics.$Y-$M.nc"
        filename_interval="01-00_00:00:00"
        precision="single"
        output_interval="6:00:00" >

     <var name="u10"/>
     <var name="v10"/>
     <var name="t2"/>
     <var name="q2"/>

</stream>
```

The only fields that will be written to this stream are the hypothetical 10-m diagnosed wind components, the 2-m temperature, and the 2-m specific humidity variables. Also, note that the filename template only includes the year and month from the model valid time; this can be problematic when the simulation starts in the middle of a month, and a solution for this problem is illustrated in the example of Section 5.3.3.

5.3.2 Example: appending records to existing output files

By default, streams will never modify existing files whose filenames match the name of a file that would otherwise be written during the course of a simulation. However, when restarting a simulation that is expected to add more records to existing output files, it can be useful to instruct the MPAS I/O system to append these records, thereby modifying existing files. This may be accomplished with the `clobber_mode` attribute.

```
<stream name="diagnostics"
        type="output"
        filename_template="diagnostics.$Y-$M.nc"
        filename_interval="01-00_00:00:00"
        precision="single"
        clobber_mode="append"
        output_interval="6:00:00" >

     <var name="u10"/>
     <var name="v10"/>
     <var name="t2"/>
     <var name="q2"/>

</stream>
```

In general, if MPAS were to attempt to write a record at a time that already existed in an output file, a clobber_mode of 'append' would not permit the write to take place, since this would modify existing data; in 'append' mode, only new records may be added. However, due to a peculiarity in the implementation of the 'append' clobber mode, it may be possible for an output file to contain duplicate times. This can happen when the first record that is appended to an existing file has a timestamp not matching any in the file, after which, any record that is written — regardless of whether its timestamp matches one already in the file — will be appended to the end of the file. This situation may arise, for example, when restarting a model simulation with a shorter `output_interval` than was used in the original model simulation with an MPAS core that does not write the first output time for restart runs.

5.3.3 Example: referencing filename intervals to a time other than the start time

The example stream of the previous sections creates a new file each month during the simulation, and the filenames contain only the year and month of the timestamp when the file was created. If a simulation begins at 00 UTC on the first day of a month, then each file in the diagnostic stream will contain only output times that fall within the month in the filename. However, if a simulation were to begin in the middle of a month — for example, the month of June, 2014 — the first diagnostics output file would have a filename of 'diagnostics.2014-06.nc', but rather than containing only output fields valid in June, it would contain all fields written between the middle of June and the middle of July, at which point one month of simulation would have elapsed, and a new output file, 'diagnostics.2014-07.nc', would be created.

In order to ensure that the file 'diagnostics.2014-06.nc' contained only data from June 2014, the `reference_time` attribute may be added such that the day, hour, minute, and second in the date and time represent the first day of the month at 00 UTC. In this example, the year and month of the reference time are not important, since the purpose of the reference time here is to describe to MPAS that the monthly filename interval begins (i.e., is referenced to) the first day of the month.

```
<stream name="diagnostics"
        type="output"
        filename_template="diagnostics.$Y-$M.nc"
```

```
        filename_interval="01-00_00:00:00"
        reference_time="2014-01-01_00:00:00"
        precision="single"
        clobber_mode="append"
        output_interval="6:00:00" >

     <var name="u10"/>
     <var name="v10"/>
     <var name="t2"/>
     <var name="q2"/>

</stream>
```

In general, the components of a timestamp, `YYYY-MM-DD_hh:mm:ss`, that are less significant than (i.e., to the right of) those contained in a filename template are important in a reference_time. For example, with a `filename_template` that contained only the year, the month component of the `reference_time` would become important to identify the month of the year on which the yearly basis for filenames would begin.

Chapter 6

Visualization

This chapter discusses visualization tools that may be used by all cores.

6.1 ParaView

ParaView may be used to visualize MPAS initialization, output, and restart files. It includes a reader that was specifically designed to read MPAS NetCDF files, including Cartesian and spherical domains. At this time, only cell-centered quantities may be plotted with ParaView. Variables located at edges and vertices must be interpolated to cell centers for visualization.

ParaView is freely available for download at `http://www.paraview.org`. Binary installations are available for Windows, Mac, and Linux, as well as source code files and tutorials. From the ParaView website:

> ParaView is an open-source, multi-platform data analysis and visualization application. ParaView users can quickly build visualizations to analyze their data using qualitative and quantitative techniques. The data exploration can be done interactively in 3D or programmatically using ParaView's batch processing capabilities. ParaView was developed to analyze extremely large datasets using distributed memory computing resources. It can be run on supercomputers to analyze datasets of terascale as well as on laptops for smaller data.

To visualize an MPAS cell-centered variable in ParaView, open the file and choose `MPAS NetCDF (Unstructured)` as the file format. In the lower left Object Inspector panel, choose your variables of interest (Figure 6.1). For large data sets, loading fewer variables will result in less wait time. Options are available for latitude-longitude projections, vertical level, etc. Click the 'Apply' button to load the data set. In the toolbars at the top, choose the variable to plot from the pull-down menu, and 'Surface' for the type of visualization. The color bar button displays a color bar, and the color scale editor button allows the user to manually change the color bar type and extents. The Filters menu provides computational tools for interactive data manipulation. Movies, in avi format or as individual frames, may be conveniently created with the `Save Animation` tool in the File menu.

Paraview may be used to view the grid from any MPAS NetCDF file by choosing `Wireframe` or `Suface With Edges` from the visualization-type pull-down menu (Figure 6.2). This produces a view of the Delaunay triangulation, which is the dual mesh to the primal Voronoi cell grid (Figure 4.1). Paraview plots all variables by interpolating colors between each corner of the Delaunay triangles. These corners are the cell-center locations of the primal grid.

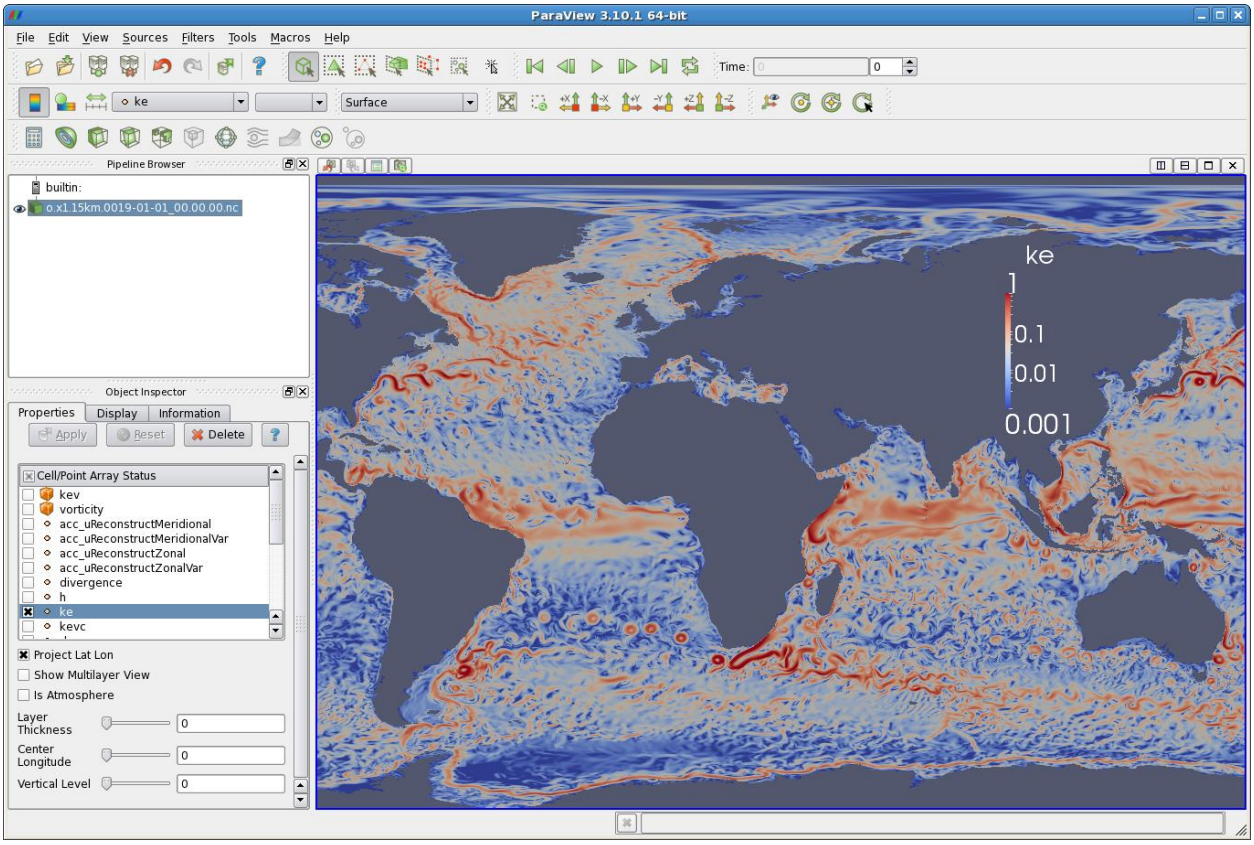


Figure 6.1: Example of ParaView to view an MPAS NetCDF file.

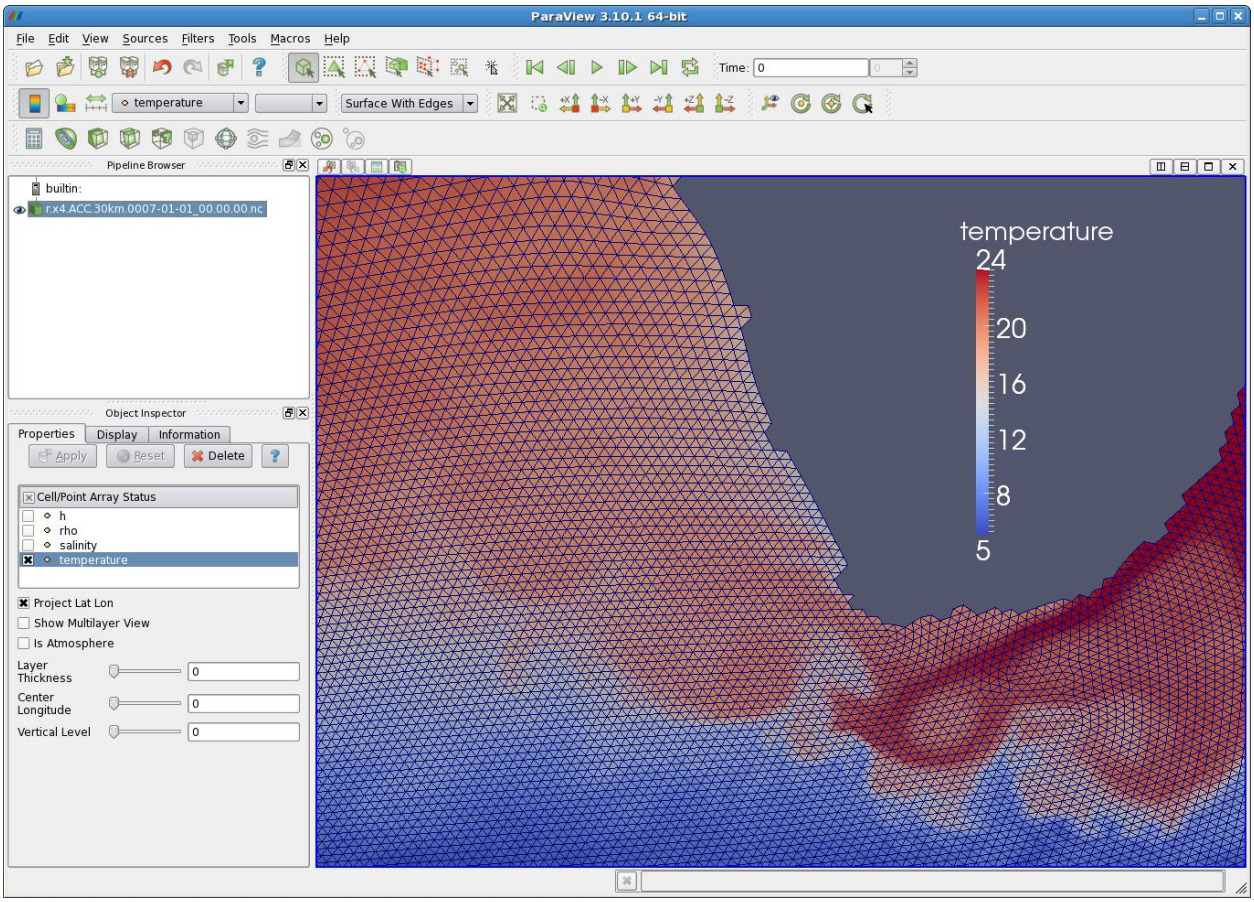


Figure 6.2: Example of visualizing the dual mesh from an MPAS NetCDF file.

Part II

MPAS-Albany Land Ice

Chapter 7

MPAS-Albany Land Ice Introduction

7.1 Background

During the past decade, numerical ice sheet models (ISMs) have undergone a renaissance relative to their predecessors. This period of intense model development was initiated following the Fourth Assessment Report of the Intergovernmental Panel on Climate Change (IPCC, 2007), which pointed to deficiencies in ISMs of the time as being the single largest shortcoming with respect to the scientific community's ability to project future sea-level rise stemming from ice sheets. Model maturation during this period, which continued through the IPCC's Fifth Assessment Report (IPCC, 2013) and to the present day, has focused on improvements to ISM "dynamical cores" (including the fidelity, discretization, and solution methods for the governing conservation equations (e.g., Bueler and Brown, 2009; Schoof and Hindmarsh, 2010; Goldberg, 2011; Perego et al., 2012; Leng et al., 2012; Larour et al., 2012; Aschwanden et al., 2012; Cornford et al., 2013; Gagliardini et al., 2013; Brinkerhoff and Johnson, 2013)), ISM model "physics" (for example, the addition of improved models of basal sliding coupled to explicit subglacial hydrology (e.g., Schoof, 2005; Werder et al., 2013; Hewitt, 2013; Hoffman and Price, 2014; Bueler and van Pelt, 2015); and ice damage, fracture, and calving (e.g., Åström et al., 2014; Bassis and Ma, 2015; Borstad et al., 2016; Jiménez et al., 2017)) and the coupling between ISMs and Earth System Models (ESMs) (e.g., Ridley et al., 2005; Vizcaíno et al., 2008, 2009; Fyke et al., 2011; Lipscomb et al., 2013). These "next generation" ISMs have been applied to community-wide experiments focused on assessing (i) the sensitivity of ISMs to idealized and realistic boundary conditions and environmental forcing and (ii) the potential future contributions of ice sheets to sea-level rise (see e.g., Pattyn et al., 2013; Nowicki et al., 2013b,a; Bindschadler et al., 2013; Shannon et al., 2013; Edwards et al., 2014b).

While these efforts represent significant steps forward, next-generation ISMs continue to confront new challenges. These come about as a result of (1) applying ISMs to larger (whole-ice sheet), higher-resolution (regionally $O(1$ km) or less), and more realistic problems, (2) adding new or improved sub-models of critical physical processes to ISMs, and (3) applying ISMs as partially or fully coupled components of ESMs. The first two challenges relate to maintaining adequate performance and robustness, as increased resolution and/or complexity have the potential to increase forward model cost and/or degrade solver reliability. The latter challenge relates to the added complexity and cost associated with optimization workflows, which are necessary for obtaining model initial conditions that are realistic and compatible with forcing from ESMs. These challenges argue for ISM development that specifically targets the following model features and capabilities:

1. parallel, scalable, and robust, linear and nonlinear solvers

2. variable and / or adaptive mesh resolution

3. computational kernels based on flexible programming models, to allow for implementation on a range of High-Performance Computing (HPC) architectures[1]

4. adjoint capabilities for use in high-dimensional parameter field optimization and uncertainty quantification

Based on these considerations, we have developed MPAS-Albany Land Ice (MALI), which is composed of three major components: 1) model framework, 2) dynamical cores for solving equations of conservation of momentum, mass, and energy, and 3) modules for additional model physics. The model leverages existing and mature frameworks and libraries, namely the Model for Prediction Across Scales (MPAS) framework and the Albany and Trilinos solver libraries. These have allowed us to take into consideration and address, from the start, many of the challenges discussed above. The model is described in detail in the following sections. MPAS-Albany Land Ice is described by a model description paper currently in review in Geoscientific Model Descriptions at: https://www.geosci-model-dev-discuss.net/gmd-2018-78/ Much of the information in this User's Guide is derived from the text of that paper. The User's Guide is further updated at the model evolves.

7.2 MALI Meshes

MALI typically uses centroidal Voronoi meshes on a plane. Spherical Voronoi meshes can also be used, but little work has been done with such meshes to date. Tools for creating and manipulating meshes are not yet publicly available. MALI employs a C-grid discretization (Arakawa and Lamb, 1977) for advection, meaning state variables (ice thickness and tracer values) are located at Voronoi cell centers, and flow variables (transport velocity, u_n) are located at cell edge midpoints (Figure 7.1). MALI uses a sigma vertical coordinate (specified number of layers, each with a spatially uniform layer thickness fraction, see (Petersen et al., 2015) for more information):

$$\sigma = \frac{s - z}{H} \tag{7.1}$$

where s is surface elevation, H is ice thickness, and z is the vertical coordinate. Table 7.1 describes the relationship between the MPAS Voronoi grid and the Delaunay Triangulation used by the Albany First Order velocity solver.

Table 7.1: Correspondence between the MPAS Voronoi tesselation and its dual Delaunay triangulation used by Albany. Key MALI model variables that are natively found at each location are listed. Note that variables are interpolated from one location to another as required for various calculations.

Voronoi tesselation	Delaunay triangulation	Variables
cell center	triangle node	H, T, u, v, Φ (MPAS)
cell edge	triangle edge	u_n (for advection)
cell vertex	triangle center	Φ (Albany)

[1]For example, traditional CPU-only architectures and MPI programming models versus CPU+GPU, hybrid architectures using MPI for nodal communication and OpenMP or CUDA for on-node parallelism.

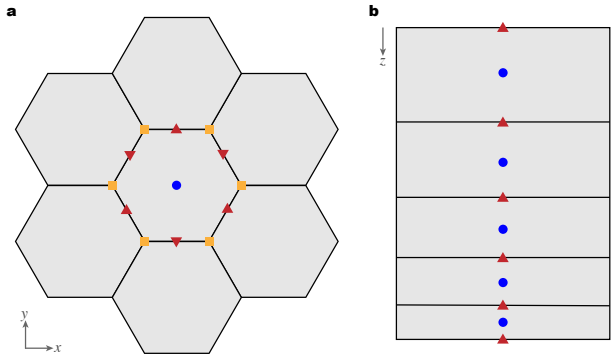


Figure 7.1: MALI grids. a) Horizontal grid with cell center (blue circles), edge midpoint (red triangles), and vertices (orange squares) identified for the center cell. Scalar fields (H, T) are located at cell centers. Advective velocities (u_n) and fluxes are located at cell edges. b) Vertical grid with layer midpoints (blue circles) and layer interfaces (red triangles) identified. Scalar fields (H, T) are located at layer midpoints. Fluxes are located at layer interfaces.

7.3 Albany velocity solver

MPAS-Albany Land Ice can optionally be compiled with support for the Albany First Order velocity solver. Albany is an open source, C++ multi-physics code base for the solution and analysis of coupled systems of partial-differential equations (PDEs) (Salinger et al., 2016). It is a finite element code that can (in three spatial dimensions) employ unstructured meshed comprised of hexahedral, tetrahedral, or prismatic elements. Albany is designed to take advantage of the computational mathematics tools available within the Trilinos suite of software libraries (Heroux et al., 2005) and it uses template-based generic programming methods to provide extensibility and flexibility (Pawlowski et al., 2012). Together, Albany and Trilinos provide parallel data structures and I/O, discretization and integration algorithms, linear solvers and preconditioners, nonlinear solvers, continuation algorithms, and tools for automatic differentiation (AD) and optimization. By formulating a system of equations in the residual form, Albany employs AD to automatically compute the Jacobian matrix, as well as forward and adjoint sensitivities. Albany can solve large-scale PDE-constrained optimization problems, using Trilinos optimization package ROL, and it provides uncertainty quantification capabilities through the Dakota framework (Adams et al., 2013). It is a massively parallel code by design and recently it has been adopting the Kokkos (Edwards et al., 2014a) programming model to provide manycore performance portability (Demeshko et al., 2018) on major HPC platforms. Albany provides several applications including LCM (Laboratory for Computational Mechanics) for solid mechanics problems, QCAD (Quantum Computer Aided Design) for quantum device modeling, and LI (Land Ice) for modeling ice sheet flow. We refer to the code that discretizes these ice sheet diagnostic momentum balance equations as Albany-LI. Albany-LI was formerly known as Albany/FELIX (Finite Elements for Land Ice eXperiments), and described by Tezaur et al. (2015a,b) and Tuminaro et al. (2016) under that name.

To compile MALI with support for Albany capabilities requires an installation of the Albany libraries. Installations exist on many Department of Energy supercomputers. To build them yourself, review the information at https://github.com/gahansen/Albany/wiki. Compiling Albany requires a build of Trilinosi (https://github.com/trilinos/Trilinos) with specific packages and third party libraries, and is not a trivial exercise. Compiling Albany for usage with MPAS requires

setting the Albany cmake configuration variable `ENABLE_MPAS_INTERFACE:BOOL=ON`. A successful Albany installation will include the file `export_albany.in`. Before compiling MPAS, the environment variable `MPAS_EXTERNAL_LIBS` should be set to the list of libraries listed in this file. Then, MALI with Albany support is enabled by compiling with `ALBANY=true`, e.g.:

```
make gfortran CORE=landice ALBANY=true
```

For MALI runs using the Albany velocity solver (`config_velocity_solver='FO'`), an Albany .xml configuration file named `albany_input.xml` is required in addition to the standard MPAS namelist and streams configuration files. Examples of that file exist in the MPAS repository within subdirectories for various tests in the `testing_and_setup/compass/landice` directory. For further details on compiling Albany and configuring run-time options for Albany, see https://github.com/gahansen/Albany.

Chapter 8

Governing Equations

The "dynamical core" of the the MALI ice sheet model solves the governing equations expressing the conservation of momentum, mass, and energy.

8.1 Conservation of Momentum

Treating glacier ice as an incompressible fluid in a low-Reynolds number flow, the conservation of momentum in a Cartesian reference frame is expressed by the Stokes-flow equations, for which the gravitational-driving stress is balanced by gradients in the viscous stress tensor, σ_{ij}:

$$\frac{\partial \sigma_{ij}}{\partial x_j} + \rho g \;=\; 0, \quad i,j = 1,2,3 \tag{8.1}$$

where x_i is the coordinate vector, ρ is the density of ice, and $g =$ is acceleration due to gravity[1].

Deformation results from the deviatoric stress, τ_{ij}, which relates to the full stress tensor as

$$\tau_{ij} \;=\; \sigma_{ij} \;-\; \frac{1}{3}\sigma_{kk}\delta_{ij}, \tag{8.2}$$

for which $-\frac{1}{3}\sigma_{kk}$ is the mean compressive stress and δ_{ij} is the Kroneker delta (or the identity tensor). Stress and strain rate are related through the constitutive relation,

$$\tau_{ij} \;=\; 2\eta_e \dot{\epsilon}_{ij}, \tag{8.3}$$

where $\dot{\epsilon_{ij}}$ is the strain-rate tensor and η_e is the "effective", non-Newtonian ice viscosity given by Nye's generalization of Glen's flow law (Glen, 1955),

$$\eta_e \;=\; \gamma A^{-\frac{1}{n}} \dot{\epsilon}_e^{\frac{1-n}{n}}. \tag{8.4}$$

In Equation 8.4, A is a temperature dependent rate factor, n is an exponent commonly taken as 3 for polycrystalline glacier ice, and γ is an ice "stiffness" factor (inverse enhancement factor) commonly used to account for other impacts on ice rheology, such as impurities or crystal anisotropy. The effective strain rate $\dot{\epsilon}_e$ is given by the second invariant of the strain-rate tensor,

$$\dot{\epsilon}_e = \left(\frac{1}{2}\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}\right)^{\frac{1}{2}}, \tag{8.5}$$

[1] In Equation 8.1 and elsewhere we use indicial notation, with summation over repeat indices.

The strain rate tensor is defined by gradients in the components of the ice velocity vector u_i:

$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \quad i,j = 1,2,3. \tag{8.6}$$

Finally, the rate-factor A follows an Arrhenius relationship

$$A\left(T^*\right) = A_o e^{-Q_a/RT^*}, \tag{8.7}$$

in which A_o is a constant, (T^*) is the absolute temperature (i.e., corrected for the dependence of melt temperature on ice pressure), Q_a is the activation energy for crystal creep, and R is the gas constant.

Boundary conditions required for the solution of Eq. 8.1 depend on the form of reduced-order approximation applied and are discussed further below.

8.2 Reduced-order Equations

Ice sheet models solve Eq. 8.1-8.7 with varying degrees of complexity in terms of the tensor components in Eq. 8.1-8.6 that are accounted for or omitted, based on geometric scaling arguments. Because ice sheets are inherently "thin" – their widths are several orders of magnitude larger than their thickness – reduced-order approximations of the full momentum balance are often appropriate (see, e.g., Dukowicz et al., 2010; Schoof and Hewitt, 2013) and, importantly, can often result in considerable computational cost savings. Here, we employ two such approximations, a first-order-accurate "Blatter-Pattyn" approximation and a zero-order, "shallow-ice approximation" as described in more detail in the following sections.

8.2.1 First-Order Velocity Solver and Coupling

Ice sheets typically have a small aspect ratio, small surface and bed slopes, and vertical pressure distributions that are very nearly hydrostatic. These characteristics imply that reduced-order approximations of the Stokes momentum balance may apply over large areas of the ice sheets, potentially allowing for significant computational savings. Formal derivations involve non-dimensionalizing the Stokes momentum balance and introducing a geometric scaling factor, $\delta = H/L$, where H and L represent characteristic vertical and horizontal length scales (often taken as the ice thickness and the ice sheet span), respectively. Upon conducting an asymptotic expansion, reduced-order models with a chosen degree of accuracy (relative to the original Stokes flow equations) can be derived by retaining terms of the appropriate order in δ. For example, the first-order accurate Stokes approximation is arrived at by retaining terms of $\mathcal{O}(\delta^1)$ and lower (The reader is referred to Schoof and Hindmarsh (2010) and Dukowicz et al. (2010) for additional discussion[2]).

Using the notation of Perego et al. (2012) and Tezaur et al. (2015a) [3], the first-order accurate Stokes approximation (also referred to as the "Blatter-Pattyn" approximation, see Blatter, 1995; Pattyn, 2003) is expressed through the following system of PDEs,

$$\begin{cases} -\nabla\cdot\left(2\eta_f\dot{\boldsymbol{\epsilon}}_1\right) + \rho g\frac{\partial s}{\partial x} = 0, \\ -\nabla\cdot\left(2\eta_f\dot{\boldsymbol{\epsilon}}_2\right) + \rho g\frac{\partial s}{\partial y} = 0, \end{cases} \tag{8.8}$$

[2] In practice, additional scaling parameters describing the ratio of deformation to sliding velocity may also be introduced.

[3] Vectors and tensors are given in bold rather than using indices. Note that, in a slight abuse of notation, we have switched from using x_1, x_2, x_3 to denote the three coordinate directions to x, y, z.

where $\nabla\cdot$ is the divergence operator, $s \equiv s(x,y)$ represents the ice sheet upper surface, and the vectors $\dot{\boldsymbol{\epsilon}}_1$ and $\dot{\boldsymbol{\epsilon}}_2$ are given by

$$\dot{\boldsymbol{\epsilon}}_1 = \left(\; 2\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy}, \quad \dot{\epsilon}_{xy}, \quad \dot{\epsilon}_{xz} \; \right)^T, \tag{8.9}$$

and

$$\dot{\boldsymbol{\epsilon}}_2 = \left(\; \dot{\epsilon}_{xy}, \quad \dot{\epsilon}_{xx} + 2\dot{\epsilon}_{yy}, \quad \dot{\epsilon}_{yz} \; \right)^T. \tag{8.10}$$

Akin to Equations 8.4 and 8.5, η_f in Equation 8.8 represents the effective viscosity but for the case of the first-order stress balance with an effective-strain rate is given by

$$\dot{\epsilon}_e \equiv \left(\dot{\epsilon}_{xx}^2 + \dot{\epsilon}_{yy}^2 + \dot{\epsilon}_{xx}\dot{\epsilon}_{yy} + \dot{\epsilon}_{xy}^2 + \dot{\epsilon}_{xz}^2 + \dot{\epsilon}_{yz}^2\right)^{\frac{1}{2}}, \tag{8.11}$$

rather than by Equation 8.5, and with individual strain rate terms given by,

$$\dot{\epsilon}_{xx} = \frac{\partial u}{\partial x}, \quad \dot{\epsilon}_{yy} = \frac{\partial v}{\partial y}, \quad \dot{\epsilon}_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \quad \dot{\epsilon}_{xz} = \frac{1}{2}\frac{\partial u}{\partial z}, \quad \dot{\epsilon}_{yz} = \frac{1}{2}\frac{\partial v}{\partial z}. \tag{8.12}$$

At the upper surface, a stress-free boundary condition is applied,

$$\dot{\boldsymbol{\epsilon}}_1 \cdot \mathbf{n} = \dot{\boldsymbol{\epsilon}}_2 \cdot \mathbf{n} = 0, \tag{8.13}$$

with $\mathbf{n}$ the outward normal vector at the ice sheet surface, $z = s(x,y)$. At the bed, $z = b(x,y)$, we apply no slip or continuity of basal tractions ("sliding"),

$$\begin{array}{ll} u = v = 0, & \text{no slip} \\ 2\mu\dot{\boldsymbol{\epsilon}}_1 \cdot \mathbf{n} + \beta u^m = 0, \;\; 2\mu\dot{\boldsymbol{\epsilon}}_2 \cdot \mathbf{n} + \beta v^m = 0, & \text{sliding}, \end{array} \tag{8.14}$$

where β is a linear-friction parameter and $m \geq 1$. In most applications we set $m = 1$ (see also Section 9.1.5).

On lateral boundaries, a stress boundary condition is applied,

$$2\mu\dot{\boldsymbol{\epsilon}}_i \cdot \mathbf{n} - \rho g(s - z)\mathbf{n} = \rho_o g \max(z, 0)\mathbf{n}, \tag{8.15}$$

where ρ_o is the density of ocean water and $\mathbf{n}$ the outward normal vector to the lateral boundary (i.e., parallel to the (x,y) plane), so that lateral boundaries above sea level are effectively stress free and lateral boundaries submerged within the ocean experience hydrostatic pressure due to the overlying column of ocean water.

We solve these equations using the Albany-LI momentum balance solver, which is built using the Albany and Trilinos software libraries discussed above. The mathematical formulation, discretization, solution methods, verification, and scaling of Albany-LI are discussed in detail in Tezaur et al. (2015a). Albany-LI implements a classic finite element discretization of the first-order approximation. At the grounding line, the basal friction coefficient β can abruptly drop to zero within an element of the mesh. This discontinuity is resolved by using an higher-order Gauss quadrature rule on elements containing the grounding line, which corresponds to the sub-element parametrization *SEP3* proposed in Seroussi et al. (2014). Additional exploration of solver scalability and demonstrations of solver robustness on large scale, high-resolution, realistic problems are discussed in Tezaur et al. (2015b). The efficiency and robustness of nonlinear solvers are achieved using a combination of the Newton method (damped with a line search strategy when needed) and of a parameter continuation algorithm for the numerical regularization of the viscosity. The scalability of linear solvers is obtained using a multilevel preconditioner (see Tuminaro et al. (2016))

specifically designed to target shallow problems characterized by meshes extruded in the vertical dimension, like those found in ice sheet modeling. The preconditioner has been demonstrated to be particularly effective and robust even in the presence of ice shelves that typically lead to highly ill-conditioned linear systems. Because the momentum balance solver is $\geq 95\%$ of the cost of a typical forward model time step, the model performance reported on in Tezaur et al. (2015a,b) and Tuminaro et al. (2016) is generally representative of overall MALI performance.

The Albany-LI first-order velocity solver written in C++ is coupled to MPAS written in Fortran using an interface layer. Albany uses a three-dimensional mesh extruded from a basal triangulation and composed of prisms or tetrahedra (see Tezaur et al. (2015a)). When coupled to MPAS, the basal triangulation is the part of the Delaunay triangulation, dual to an MPAS Voronoi mesh, that contains active ice and it is generated by the interface. Bed topography, ice lower surface, ice thickness, basal friction coefficient (β), and three-dimensional ice temperature, all at cell centers (Table 7.1), are passed from MPAS to Albany. Optionally, Dirichlet velocity boundary conditions can also be passed. After the velocity solve is complete, Albany returns the x and y components of velocity at each cell center and blue layer interfaces, the normal component of velocity at each cell edge and blue layer interfaces, and viscous dissipation at blue each cell vertex and layer midpoints.

The interface code defines the lateral boundary conditions on the finite element mesh that Albany will use. Lateral boundaries in Albany are applied at cell centers (triangle nodes) that do not contain dynamic ice on the MPAS mesh and that are adjacent to the last cell of the MPAS mesh that does contain dynamic ice. This one element extension is required to support calculation of normal velocity on edges (u_n) required for advection of ice out of the final cell containing dynamic ice (Figure 8.1). The interface identifies three types of lateral boundaries for the first-order velocity solve: terrestrial, floating marine, and grounded marine. Terrestrial margins are defined by bed topography above sea level. At these boundary nodes, ice thickness is set to a small ice minimum thickness value ($\epsilon = 1$ m). Floating marine margin triangle nodes are defined as neighboring one or more triangle edges that satisfy the hydrostatic floatation criterion. At these boundary nodes, we need to ensure the existence of a realistic calving front geometry, so we set ice thickness to the minimum of thickness at neighboring cells with ice. Grounded marine margins are defined as locations where the bed topography is below sea level, but no adjacent triangle edges satisfy the floatation criterion. At these boundary nodes, we apply a small floating extension with thickness ϵ. For all three boundary types, ice temperature is averaged from the neighboring locations containing ice.

8.2.2 Shallow-Ice Approximation Velocity Solver

A similar procedure to that described above for the first-order accurate Stokes approximation can be used to derive the so-called "shallow-ice approximation" (SIA) (Hutter, 1983; Fowler and Larson, 1978; Morland and Johnson, 1980; Payne et al., 2000), in this case by retaining only terms of $\mathcal{O}(\delta^0)$. In the case of the SIA, the local gravitational driving stress is everywhere balanced by the local basal traction and the horizontal velocity as a function of depth is simply the superposition of the local basal sliding velocity and the integral of the vertical shear from the ice base to that depth:

$$\boldsymbol{u} = -2(\rho g)^n \left(\int_b^z A(s-z)^n dz \right) |\nabla s|^{n-1} \nabla s + \boldsymbol{u_b} \tag{8.16}$$

where b is the bed elevation and $\vec{u_b}$ is the sliding velocity.

SIA ice sheet models typically combine the momentum and mass balance equations to evolve the ice geometry directly in the form of a depth-integrated, two-dimensional diffusion problem (Hindmarsh and Payne, 1996; Payne et al., 2000). However we implement the SIA as an explicit

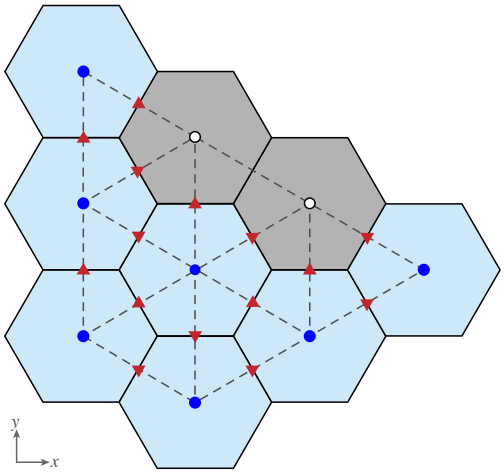


Figure 8.1: Correspondence between MPAS and Albany meshes and application of boundary conditions for the first-order velocity solver. Solid black lines are cells on the Voronoi mesh and dashed gray lines are triangles on the Delaunay Triangulation. Light blue Voronoi cells contain dynamic ice and gray cells do not. Dark blue circles are Albany triangle nodes that use variable values directly from the co-located MPAS cell centers. White circles are extended node locations that receive variable values as described in the text based on whether they are terrestrial, floating marine, or grounded marine locations. Red triangles indicate Voronoi cell edges on which velocities (u_n) are required for advection.

velocity solver that can be enabled in place of the more accurate first order solver, while keeping the rest of the model identical. The purpose of the SIA velocity solver is primarily for rapid testing, so the less efficient explicit implementation of Eq. 8.16 is not a concern.

We implement Eq. 8.16 in sigma coordinates on cell edges, where we only require the normal component of velocity, u_n:

$$u_n = -2(\rho g)^n H^{n+1} |\nabla s|^{n-1} \frac{ds}{dx_n} \int_1^\sigma A \sigma^n d\sigma + u_{b_n} \tag{8.17}$$

where x_n is the normal direction to a given edge and u_{b_n} is sliding velocity in the normal direction to the edge. We average A and H from cell centers to cell edges. $\frac{ds}{dx_n}$ is calculated as the difference in surface elevation between the two cells that neighbor a given edge divided by the distance between the cell centers; on a Voronoi grid, cells edges are midway between cell centers by definition. The surface slope component tangent to an edge (required to complete the calculation of ∇s) is calculated by first interpolating surface elevation from cell centers to vertices.

8.3 Conservation of Mass

Conservation of mass is used to conduct ice sheet mass transport and evolution. Assuming constant density to write conservation of mass in volume form, the equation relates ice thickness change to the divergence of mass and sources and sinks:

$$\frac{\partial H}{\partial t} + \nabla \cdot H\bar{\mathbf{u}} = \dot{a} + \dot{b}, \tag{8.18}$$

where H is ice thickness, t is time, $\bar{\mathbf{u}}$ is depth-averaged velocity, $\dot{a}$ is surface mass balance, and $\dot{b}$ is basal mass balance. Both $\dot{a}$ and $\dot{b}$ are positive for ablation and negative for accumulation.

Eq. 8.18 is used to update thickness in each grid cell on each time step using a forward Euler, fully explicit time evolution scheme. Eq. 8.18 is implemented using a finite volume method, such that fluxes are calculated for each edge of each cell to calculate $\nabla \cdot H\bar{\mathbf{u}}$. Specifically, we use a first-order upwind method that applies the normal velocity on each edge (u_n) and an upwind value of cell centered ice thickness. Note that with the First Order velocity solver, normal velocity is interpolated from cell centers to edges using the finite element basis functions in Albany. In the shallow ice approximation velocity solver, normal velocity is calculated natively at edges. MPAS Framework includes a higher-order flux-corrected transport scheme (Ringler et al., 2013) for which we have performed some initial testing, but is not routinely used in the Land Ice core at this time.

Tracers are advected layer by layer with a similar equation:

$$\frac{\partial(Q_t l)}{\partial t} + \nabla \cdot (Q_t l \bar{\mathbf{u}}) = \dot{S} \tag{8.19}$$

where Q_t is a tracer quantity (e.g., temperature – see below), l is layer thickness, and $\dot{S}$ represents any tracer sources or sinks. While any number of tracers can be included in the model, the only one to be considered here is temperature, due to its important effect on ice rheology through Eq. 8.7 and will be discussed further in the following section.

Because we employ a sigma vertical coordinate system with fixed layer fractions, after Eqs. 8.18 and 8.19 are applied, a vertical remapping operation is required. Overlaps between the newly calculated layers and the target sigma layers are calculated for each grid cell. Assuming uniform values within each layer, mass, energy, and other tracers are transferred between layers based on these overlaps to restore the prescribed sigma layers while conserving mass and energy.

8.4 Conservation of Energy

Conservation of energy is expressed through the three-dimensional, advective-diffusive heat equation:

$$\frac{\partial T}{\partial t} = \frac{1}{\rho c}\frac{\partial}{\partial x_i}\left(k\frac{\partial T}{\partial x_i}\right) - u_i\frac{\partial T}{\partial x_i} + \frac{\Phi}{\rho c}, \tag{8.20}$$

with thermal conductivity k and heat capacity c. In Equation 8.20, the rate of temperature change (left-hand side) is balanced by diffusive, advective, and internal (viscous dissipation – see Equation 8.27 for Φ) source terms (first, second, and third terms on the right-hand side, respectively). In MALI we solve an approximation of Equation 8.20,

$$\frac{\partial T}{\partial t} = \frac{k}{\rho c}\frac{\partial^2 T}{\partial z^2} - u_i\frac{\partial T}{\partial x_i} + \frac{\Phi}{\rho c}, \tag{8.21}$$

in which horizontal diffusion is assumed negligible (van der Veen, 2013, p. 280) and k is assumed constant and uniform. The viscous dissipation term Φ is discussed further below (Section 8.4.2).

Temperatures are staggered in the vertical relative to velocities and are located at the centers of $nz-1$ vertical layers, which are bounded by nz vertical levels (grid point locations). This convention allows for conservative temperature advection, since the total internal energy in a column (the sum of $\rho c T \Delta z$ over $nz-1$ layers) is conserved under transport. The upper surface temperature T_s and the lower surface temperature T_b, coincident with the surface and bed grid points, give a total of $nz+1$ temperature values within each column.

Equation 8.21 is solved using an operator splitting technique. At each time step, we first perform an implicit vertical solve accounting for the diffusion and dissipation terms, and then we explicitly advect horizontally the resulting temperature.

8.4.1 Vertical Diffusion

Using a "sigma" vertical coordinate, the vertical diffusion portion of Equation 8.21 can be discretized as:

$$\frac{\partial^2 T}{\partial z^2} = \frac{1}{H^2}\frac{\partial^2 T}{\partial \sigma^2}. \tag{8.22}$$

In σ–coordinates, the central difference formulas for first partial derivatives at the upper and lower interfaces of layer k are

$$\begin{aligned} \left.\frac{\partial T}{\partial \sigma}\right|_{\sigma_k} &= \frac{T_k - T_{k-1}}{\tilde{\sigma}_k - \tilde{\sigma}_{k-1}}, \\ \left.\frac{\partial T}{\partial \sigma}\right|_{\sigma_{k+1}} &= \frac{T_{k+1} - T_k}{\tilde{\sigma}_{k+1} - \tilde{\sigma}_k}, \end{aligned} \tag{8.23}$$

where $\tilde{\sigma}_k$ is the value of σ at the midpoint of layer k, halfway between σ_k and σ_{k+1}. The second partial derivative, defined at the midpoint of layer k, is then given by

$$\left.\frac{\partial^2 T}{\partial \sigma^2}\right|_{\tilde{\sigma}_k} = \frac{\left.\frac{\partial T}{\partial \sigma}\right|_{\sigma_{k+1}} - \left.\frac{\partial T}{\partial \sigma}\right|_{\sigma_k}}{\sigma_{k+1} - \sigma_k} \tag{8.24}$$

By inserting Equation (8.23) into Equation (8.24), we obtain the discrete form of the vertical diffusion term in Equation 8.21:

$$\begin{aligned} \left.\frac{\partial^2 T}{\partial \sigma^2}\right|_{\tilde{\sigma}_k} = \frac{T_{k-1}}{(\tilde{\sigma}_k - \tilde{\sigma}_{k-1})(\sigma_{k+1} - \sigma_k)} - T_k\left(\frac{1}{(\tilde{\sigma}_k - \tilde{\sigma}_{k-1})(\sigma_{k+1} - \sigma_k)} + \frac{1}{(\tilde{\sigma}_{k+1} - \tilde{\sigma}_k)(\sigma_{k+1} - \sigma_k)}\right) \\ + \frac{T_{k+1}}{(\tilde{\sigma}_{k+1} - \tilde{\sigma}_k)(\sigma_{k+1} - \sigma_k)}. \end{aligned} \tag{8.25}$$

To simplify some expressions below, we define the following coefficients associated with the vertical temperature diffusion,

$$a_k = \frac{1}{(\tilde{\sigma}_k - \tilde{\sigma}_{k-1})(\sigma_{k+1} - \sigma_k)}, b_k = \frac{1}{(\tilde{\sigma}_{k+1} - \tilde{\sigma}_k)(\sigma_{k+1} - \sigma_k)}. \tag{8.26}$$

8.4.2 Viscous Dissipation

The source term from viscous dissipation in Equation 8.21 is given by the product of the stress and strain rate tensors:

$$\Phi = \sigma_{ij}\dot{\epsilon}_{ij} = \tau_{ij}\dot{\epsilon}_{ij}. \tag{8.27}$$

The change to deviatoric stress on the right-hand side of Equation 8.27 follows from terms related to the mean compressive stress (or pressure) dropping out due to incompressibility. Analogous to the effective strain rate given in Equation 8.5, the effective-deviatoric stress is given by

$$\tau_e = \left(\frac{1}{2}\tau_{ij}\tau_{ij}\right)^{\frac{1}{2}}, \tag{8.28}$$

which can be combined with Equations 8.27 and 8.5 to derive an expression for the viscous dissipation in terms of effective deviatoric stress and strain,

$$\Phi = 2\tau_e \dot{\epsilon}_e \tag{8.29}$$

Finally, an analog to Equation 8.3 gives

$$\tau_e \;=\; 2\eta_e \dot{\epsilon}_e, \tag{8.30}$$

which can be used to eliminate $\dot{\epsilon}_e$ in Equation 8.29 and arrive at an alternate expression for the dissipation based on only two scalar quantities

$$\Phi = 4\eta_e \dot{\epsilon}_e^2. \tag{8.31}$$

The viscous dissipation source term is computed within Albany-LI at MPAS cell vertices and then reconstructed at cell centers in MPAS.

For the SIA model, dissipation can be calculated in sigma coordinates as

$$\Phi(\sigma) = \frac{\sigma g}{c} \frac{\partial \vec{u}}{\partial \sigma} \cdot \nabla s \tag{8.32}$$

which can be combined with Eq. 8.16 to make:

$$\Phi(\sigma) = -\frac{2\sigma g}{c\rho} (g\sigma\rho)^{n+1} (H|\nabla s|)^{n+1} A \tag{8.33}$$

We calculate Φ on cell edges following the procedure described for Eq. 8.17, and then interpolate Φ back to cell centers to solve Eq. 8.21.

8.4.3 Vertical Temperature Solution

The vertical diffusion portion of Equation 8.21 is discretized according to

$$\frac{T_k^{n+1} - T_k^n}{\Delta t} = \frac{k}{\rho c H^2} \left(a_k T_{k-1}^{n+1} - (a_k + b_k) T_k^{n+1} + b_k T_{k+1}^{n+1} \right) + \frac{\Phi_k}{\rho c}, \tag{8.34}$$

where a_k and b_k are defined in (8.26), n is the current time level, and $n+1$ is the new time level. Because the vertical diffusion terms are evaluated at the new time level, the discretization is backward-Euler (fully implicit) in time.

The temperature T_0 at the upper boundary is set to $\min(T_{\text{air}}, 0)$, where the mean-annual surface air temperature T_{air} is a two-dimensional field specified from observations or climate model output.

At the lower boundary, for grounded ice there are three potential heat sources and sinks: (1) the diffusive flux from the bottom surface to the ice interior (positive up),

$$F_d^{\text{bot}} = \frac{k}{H} \frac{T_{nz} - T_{nz-1}}{1 - \tilde{\sigma}_{nz-1}}; \tag{8.35}$$

(2) the geothermal flux F_g, prescribed from a spatially variable input file (based on observations), and (3) the frictional heat flux associated with basal sliding,

$$F_f = \tau_{\mathbf{b}} \cdot \mathbf{u}_{\mathbf{b}}, \tag{8.36}$$

where $\tau_\mathbf{b}$ and $\mathbf{u_b}$ are 2D vectors of basal shear stress and basal velocity, respectively, and the friction law from Equation 8.14 becomes

$$F_f = \beta\sqrt{u_b^2 + v_b^2}. \tag{8.37}$$

If the basal temperature $T_{nz} < T_{\text{pmp}}$ (where T_{pmp} is the pressure melting point temperature), then the fluxes at the lower boundary must balance,

$$F_g + F_f = F_d^{\text{bot}}, \tag{8.38}$$

so that the energy supplied by geothermal heating and sliding friction is equal to the energy removed by vertical diffusion. If, on the other hand, $T_{nz} = T_{\text{pmp}}$, then the net flux is nonzero and is used to melt or freeze ice at the boundary:

$$M_b = \frac{F_g + F_f - F_d^{\text{bot}}}{\rho L}, \tag{8.39}$$

where M_b is the melt rate and L is the latent heat of melting. Melting generates basal water, which may either be stored at the bed locally, serve as a source for the basal hydrology model (See Section 9.1), or may simply be ignored. If basal water is present locally, T_{nz} is held at T_{pmp}.

For floating ice the basal boundary condition is simpler: T_{nz} is simply set to the freezing temperature T_f of seawater. Optionally, a melt rate can be prescribed at the lower surface.

Rarely, the solution for T may exceed T_{pmp} for a given internal layer. In this case, T is set to T_{pmp}, excess energy goes towards melting of ice internally, and the resulting melt is assumed to drain to the bed immediately.

If (8.39) applies, we compute M_b and adjust the basal water depth. When the basal water goes to zero, T_{nz} is set to the temperature of the lowest layer (less than T_{pmp} at the bed) and flux boundary conditions apply during the next time step.

8.4.4 Horizontal Advection

Temperature advection in any individual layer k is treated using tracer advection, as in Equation 8.19 above, where the ice temperature T_k is substituted for the generic tracer Q. After horizontal transport, the surface and basal mass balance is applied to the top and bottom ice surfaces, respectively. Because layer transport and the application of mass balance terms results in an altered vertical-layer spacing with respect to σ coordinates, a vertical remapping scheme is applied. This conservatively transfers ice volume and internal energy between adjacent layers while restoring σ layers to their initial distribution. Internal energy divided by mass gives the new layer temperatures.

Chapter 9

Model Physics

Physical processes currently implemented in MALI are a mass-conserving subglacial hydrology model and a small number of basic schemes for iceberg calving. These are described in more detail below.

9.1 Subglacial Hydrology

Sliding of glaciers and ice sheets over their bed can increase ice velocity by orders of magnitude and is the primary control on ice flux to the oceans. The state of the subglacial hydrologic system is the primary control on sliding (Clarke, 2005; Cuffey and Paterson, 2010; Flowers, 2015), and ice sheet modelers have therefore emphasized subglacial hydrology and its effects on basal sliding as a critical missing piece of current ice sheet models (Little et al., 2007; Price et al., 2011).

MALI includes a mass-conserving model of subglacial hydrology that includes representations of any or all of water storage in till, distributed drainage, and channelized drainage and is coupled to ice dynamics. The model is based on the model of Bueler and van Pelt (2015) with an additional component for channelized drainage and modified for MALI's unstructured horizontal grid. While the implementation follows closely that of Bueler and van Pelt (2015), the model and equations are summarized here along with a description of the features unique to the application in MALI.

9.1.1 Till

The simple till component represents local storage of water in subglacial till without horizontal transport within the till. Evolution of the effective water depth in till, W_{till} is therefore a balance of delivery of meltwater, m_b, to the till, drainage of water out of the till at rate C_d (mass leaving the subglacial hydrologic system, for example, to deep groundwater storage), and overflow to the distributed drainage system, γ:

$$\frac{\partial W_{till}}{\partial t} = \frac{m_b}{\rho_w} - C_d - \gamma_t. \tag{9.1}$$

In the model, meltwater (from either the bed or drained from the surface), is first delivered to the till component. Water in excess of the the maximum storage capacity of the till, W_{till}^{max}, is instantaneously transferred as a source term to the distributed drainage system through the γ_t term.

Distributed drainage

The distributed drainage component is implemented as a "macroporous sheet" that represents bulk flow through linked cavities that form in the lee of bedrock bumps as the glacier slides over the bed (Flowers and Clarke, 2002; Hewitt, 2011; Flowers, 2015). Water flow in the system is driven by the gradient of the hydropotential, ϕ, defined as

$$\phi = \rho_w g z_b + P_w \tag{9.2}$$

where P_w is the water pressure in the distributed drainage system. A related variable, the ice effective pressure, N, is the difference between ice overburden pressure and water pressure in the distributed drainage system, P_w:

$$N = \rho g H - P_w. \tag{9.3}$$

The evolution of the area-averaged cavity space is a balance of opening of cavity space by the glacier sliding over bedrock bumps and closing through creep of the ice above. The model uses the commonly used assumption (e.g. Schoof, 2010; Hewitt, 2011; Werder et al., 2013; Hoffman and Price, 2014) that cavities always remain water filled (c.f. Schoof et al., 2012), so cavity space can be represented by the effective water depth in the macorporous sheet, W:

$$\frac{\partial W}{\partial t} = c_s |\vec{u_b}| (W_r - W) - c_{cd} A_b N^3 W \tag{9.4}$$

where c_s is bed roughness parameter, W_r is the maximum bed bump height, c_{cd} is creep scaling parameter representing geometric and possibly other effects, and A_b is the ice flow parameter of the basal ice.

Water flow in the distributed drainage system, $\vec{q}$, is driven the hydropotential gradient and is described by a general power-law:

$$\vec{q} = -k_q W^{\alpha_1} |\nabla \phi|^{\alpha_2 - 2} \nabla \phi \tag{9.5}$$

where k_q is a conductivity coefficient. The α_1 and α_2 exponents can be adjusted so that Eq. 9.5 reduces to commonly used water flow relations, such as Darcy flow, the Darcy-Weisbach relation, and the Manning equation.

9.1.2 Channelized drainage

The inclusion of channelized drainage in MALI is an extension to the model of Bueler and van Pelt (2015). The distributed drainage model ignores dissipative heating within the water, which in the real world leads to melting of the ice roof, and the formation of discrete, efficient channels melted into the ice above when the distributed discharge reaches a critical threshold (Schoof, 2010; Hewitt, 2011; Werder et al., 2013; Flowers, 2015). These channels can rapidly evacuate water from the distributed drainage system and lower water pressure, even under sustained meltwater input (Schoof, 2010; Hewitt, 2011; Werder et al., 2013; Hoffman and Price, 2014; Flowers, 2015).

The implementation of channels follows the channel network models of Werder et al. (2013) and Hewitt (2013). The evolution of channel area, S, is a balance of opening and closing processes as in the distributed system, but in channels the opening mechanism is melting caused by dissipative heating of the ice above:

$$\frac{dS}{dt} = \frac{1}{\rho L} (\Xi - \Pi) - c_{cc} A_b N^3 S \tag{9.6}$$

where c_{cc} is the creep scaling parameter for channels.

The channel opening rate, the first term in Eq. 9.6, is itself a balance of dissipation of potential energy, Ξ, and sensible heat change of water, Π, due to changes in the pressure-dependent melt temperature. Dissipation of potential energy includes energy produced by flow in both the channel itself and a small region of the distributed system along the channel:

$$\Xi = \left| \frac{d\phi}{ds} \vec{Q} \right| + \left| \frac{d\phi}{ds} \vec{q_c} l_c \right| \tag{9.7}$$

where s is the spatial coordinate along a channel segment, $\vec{Q}$ is the flow rate in the channel, and $\vec{q_c}$ is the flow in the distributed drainage system parallel to the channel within a distance l_c of the channel. The term adding the contribution of dissipative melting within the distributed drainage system near the channel is included to represent some of the energy that has been ignored from that process in the description of the distributed drainage system and allows channels to form even when channel area is initially zero if discharge in the distributed drainage system is sufficient (Werder et al., 2013). The term representing sensible heat change of the water, Π, is necessitated by the assumption that the water always remains at the pressure-dependent melt temperature of the water. Changes in water pressure must therefore result in melting or freezing:

$$\Pi = -c_t c_w \rho_w \left(\vec{Q} + l_c \vec{q_c} \right) \frac{dP_w}{ds} \tag{9.8}$$

where c_t is the Clapeyron slope and c_w is the specific heat capacity of water. The pressure-dependent melt term can be disabled in the model.

Water flow in channels, $\vec{Q}$, mirrors Eq. 9.5:

$$\vec{Q} = -k_Q S^{\alpha_1} |\nabla \phi|^{\alpha_2 - 2} \nabla \phi \tag{9.9}$$

where k_Q is a conductivity coefficient for channels.

9.1.3 Drainage component coupling

Eqs. 9.1-9.9 are coupled together by describing the drainage system with two equations, mass conservation and pressure evolution. Mass conservation of the subglacial drainage system is described by

$$\frac{\partial W}{\partial t} + \frac{\partial W_{till}}{\partial t} = -\nabla \cdot (\vec{V_d} W) + \nabla \cdot (D_d \nabla W) - \left[\frac{\partial S}{\partial t} + \frac{\partial Q}{\partial s} \right] \delta(x_c) + \frac{m_b}{\rho_w} \tag{9.10}$$

where V_d is water velocity in the distributed flow, D_d is the diffusivity of the distributed flow, and $\delta(x_c)$ is the Dirac delta function applied along the locations of the linear channels.

Combining Eq. 9.10 and Eq. 9.4 and making the simplification that cavities remain full at all times yields an equation for water pressure within the distributed drainage system, P_w:

$$\frac{\phi_0}{\rho_w g} \frac{\partial P_w}{\partial t} = -\nabla \cdot \vec{q} + c_s |\vec{u_b}| (W_r - W) - c_{cd} A_b N^3 W - \left[\frac{\partial S}{\partial t} + \frac{\partial Q}{\partial s} \right] \delta(x_c) + \frac{m_b}{\rho_w} - \frac{\partial W_{till}}{\partial t} \tag{9.11}$$

where ϕ_0 is an englacial porosity used to regularize the pressure equation. Following Bueler and van Pelt (2015), the porosity is only included in the pressure equation and is excluded from the mass conservation equation.

Any of the three drainage components (till, distributed drainage, channelized drainage) can be deactivated at runtime. The most common configuration currently used is to run with distributed drainage only.

9.1.4 Numerical implementation

The drainage system model is implemented using Finite Volume Methods on the unstructured grid used by MALI. State variables (W, W_{till}, S, P_w) are located at cell centers and velocities and fluxes $(\vec{q}, \vec{V}_d, \vec{Q})$ are calculated at edge midpoints. Channel segments exist along the lines joining neighboring cell centers. Eq. 9.10 is evaluated by summing tendencies from discrete fluxes into or out of each cell. First-order upwinding is used for advection. At land-terminating ice sheet boundaries, $P_w = 0$ is applied as the boundary condition. At marine-terminating ice sheet boundaries, the boundary condition is $P_w = -\rho_w g z_b$, where ρ_w is ocean water density. The drainage model uses explicit forward Euler time-stepping using Eqs. 9.1, 9.10, 9.6, and 9.11. This requires obeying advective and diffusive Courant-Friedrichs-Lewy (CFL) conditions for distributed drainage as described by Bueler and van Pelt (2015), as well as an additional advective CFL condition for channelized drainage, if it is active.

We acknowledge that the non-continuum implementation of channels can make the solution grid-dependent, and grid convergence may therefore not exist for many problems (Bueler and van Pelt, 2015). However, for realistic problems with irregular bed topography, we have found dominant channel location is controlled by topography, mitigating this issue.

9.1.5 Coupling to ice sheet model

The subglacial drainage model is coupled to the ice dynamics model through a basal friction law. Currently, the only option is a modified Weertman-style power law (Bindschadler, 1983; Hewitt, 2013) that adds a term for effective pressure to Eq. 8.14:

$$\vec{\tau_b} = C_0 N \vec{u_b}^m \tag{9.12}$$

where C_0 is a friction parameter. Implementation of a Coulomb friction law (Schoof, 2005; Gagliardini et al., 2007) and a plastic till law (Tulaczyk et al., 2000; Bueler and van Pelt, 2015) are in development. When the drainage and ice dynamics components are run together, coupling of the systems allows the negative feedback described by (Hoffman and Price, 2014) where elevated water pressure increases ice sliding and increased sliding opens additional cavity space, lowering water pressure. The meltwater source term, m, is calculated by the thermal solver in MALI. Either or both of the ice dynamics and thermal solvers can be disabled, in which case the relevant coupling fields can be prescribed to the drainage model.

9.1.6 Verification and Real-world Application

To verify the implementation of the distributed drainage model, we use the nearly exact solution described by Bueler and van Pelt (2015). The problem configuration uses distributed drainage only on a two-dimensional, radially-symmetric ice sheet of radius 22.5 km with parabolic ice sheet thickness and a nontrivial sliding profile. Bueler and van Pelt (2015) showed that this configuration allows for nearly-exact reference values of W and P_w to be solved at steady state from an ordinary differential equation initial value problem with very high accuracy. We follow the test protocol of Bueler and van Pelt (2015) and initialize the model with the near-exact solution and then run the model forward for one month, after which we evaluate model error due to drift away from the expected solution. Performing this test with the MALI drainage model, we find error comparable to that found by Bueler and van Pelt (2015) and approximately first order convergence (Figure 9.1).

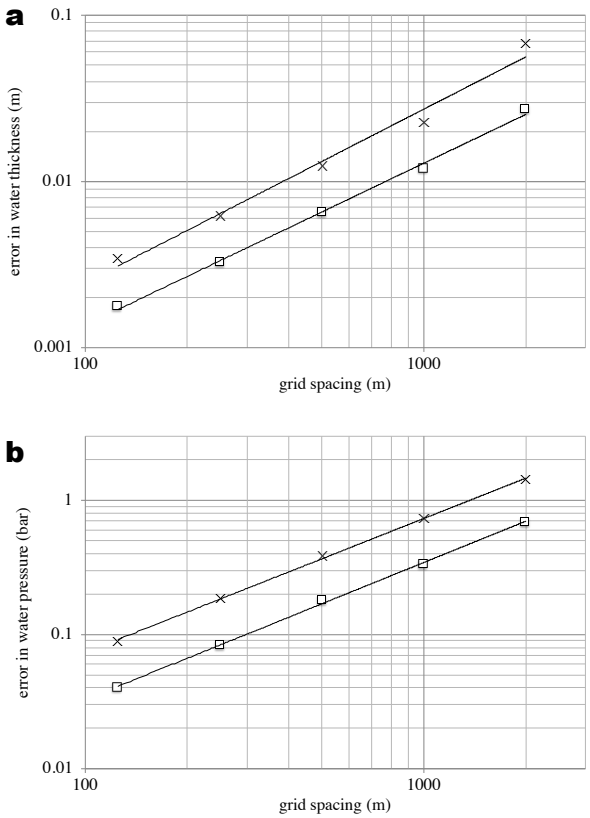


Figure 9.1: Error in subglacial hydrology model for radial test case with near-exact solution described by Bueler and van Pelt (2015) for different grid resolutions. a) Error in water thickness. $\times$ symbols indicate maximum error, and squares indicate mean error. Average error in water thickness decays as $O(\Delta x^{0.97})$. b) Error in water pressure, with same symbols. Average error in water pressure decays as $O(\Delta x^{1.02})$.

To check the model implementation of channels, we use comparisons to other, more mature drainage models through the Subglacial Hydrology Model Intercomparison Project (SHMIP)[1]. Steady state solutions of the drainage system effective pressure, water fluxes, and channel development for an idealized ice sheet with varying magnitudes of meltwater input (SHMIP experiment suites A and B) compared between MALI and other models of similar complexity (GlaDS, Elmer) are very similar.

To demonstrate a real-world application of the subglacial hydrology model, we perform a standalone subglacial hydrology simulation of the entire Antarctica Ice Sheet on a uniform 20 km resolution mesh (Figure 9.2). We force this simulation with basal sliding and basal melt rate after optimizing the first-order velocity solver optimized to surface velocity observations (Figure 9.2a). We then run the subglacial hydrology model to steady state with only distributed drainage active and using standard parameter values. Results show a subglacial hydrologic state that is reasonable. For example, the subglacial water flux shows similar spatial patterns to the basal sliding speed (Figure 9.2), suggesting a basal friction law based on the subglacial hydrologic state could be configured to yield realistic ice velocity. Calibrating parameters for the subglacial hydrology

[1] https://shmip.bitbucket.io/

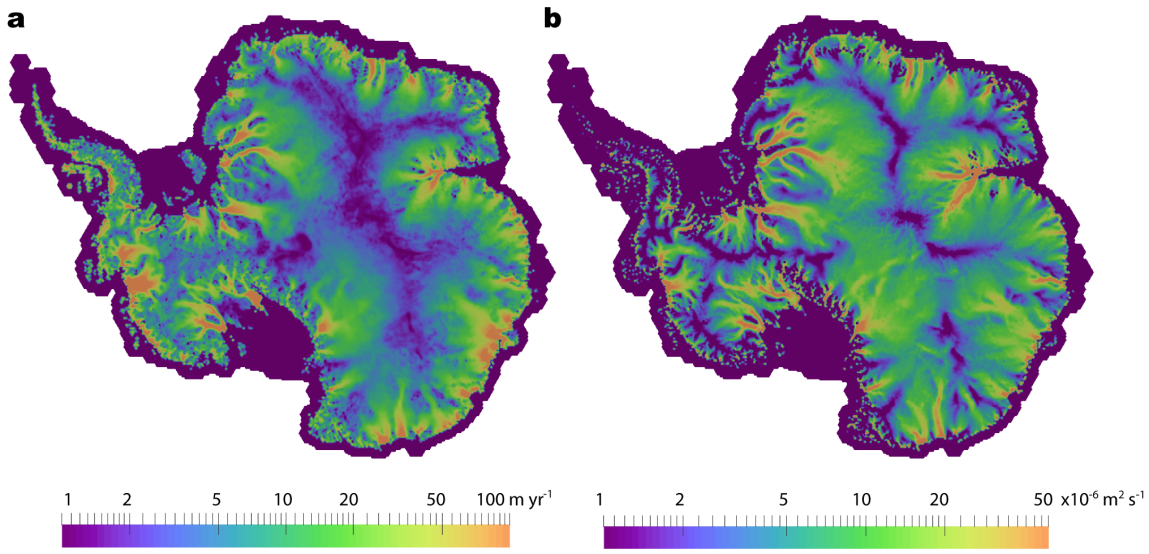


Figure 9.2: Subglacial hydrology model results for 20 km resolution Antarctic Ice Sheet. a) Basal ice speed calculated by the first-order velocity solver optimized to surface velocity observations. This field and the calculated basal melt are the forcings applied to the standalone subglacial hydrology model. b) Water flux in the distributed system calculated by the subglacial hydrology model at steady state.

model and a basal friction law and performing coupled subglacial-hydrology/ice-dynamics simulations are beyond the scope of this paper; we merely mean to demonstrate plausible behavior from the subglacial hydrology model for a realistic ice-sheet-scale problem.

9.2 Iceberg Calving

MALI includes a few simple methods for removing ice from calving fronts during each model time step:

1. All floating ice is removed.
2. All floating ice in cells with an ocean bathymetry deeper than a specified threshold is removed.
3. All floating ice thinner than a specified threshold is removed.
4. The calving front is maintained at its initial location by adding or removing ice after thickness evolution is complete. This option does *not* conserve mass or energy but provides a simple way to maintain a realistic ice shelf extent (e.g., for model spinup).
5. "Eigencalving" scheme (Levermann et al., 2012). Calving front retreat rate, C_v, is proportional to the product of the principal strain rates $(\dot{\epsilon_1}, \dot{\epsilon_2})$ if they both are extensional:

$$C_v = K_2 \dot{\epsilon_1} \dot{\epsilon_2} \text{ for } \dot{\epsilon_1} > 0 \text{ and } \dot{\epsilon_2} > 0. \tag{9.13}$$

The eigencalving scheme can optionally also remove floating ice at the calving front with thickness below a specified thickness threshold (Feldmann and Levermann, 2015). In practice

we find this is necessary to prevent formation of tortuous ice tongues and continuous, gradual extension of some ice shelves along the coast.

Ice that is eligible for calving can be removed immediately or fractionally each time step based on a calving timescale. To allow ice shelves to advance as well as retreat, we implement a simple parameterization for sub-grid motion of the calving front by forcing floating cells adjacent to open ocean to remain dynamically inactive until ice thickness there reaches 95% of the minimum thickness of all floating neighbors. This is an *ad hoc* alternative to methods tracking the calving front position at sub-grid scales (Albrecht et al., 2011; Bondzio et al., 2016).

Chapter 10

Model Analysis

As with other climate model components built using the MPAS framework, MALI supports the development and application of "analysis members", which allow for a wide range of run-time-generated simulation diagnostics and statistics output at user specified time intervals. Support tools included with the code release allow for the definition of any number or combination of pre-defined "geographic features" – points, lines ("transects"), or areas ("regions") – of interest within an MPAS mesh. Features are defined using the standard GeoJSON format (Butler et al., 2016) and a large existing database of globally defined features is currently supported[1]. Python-based scripts are available for editing GeoJSON feature files, combining or splitting them, and using them to define their coverage within MPAS mesh files. Currently, MALI includes support for standard ice sheet model diagnostics (see Table 10.1) defined over the global domain (by default) and / or over specific ice sheet drainage basins and ice shelves (or their combination). Support for generating model output at points and along transects will be added in the future (e.g., vertical samples at ice core locations or along ground-penetrating radar profile lines). Defining regions requires tools that are not yet publicly available.

[1] https://github.com/MPAS-Dev/geometric_features

Table 10.1: Standard model diagnostics available for an arbitrary number of predefined geographic regions.

diagnostic	units
net ice area and volume	m^2, m^3
net grounded ice area and volume	m^2, m^3
net floating ice area and volume	m^2, m^3
net volume above floatation	m^3
minimum, maximum, and mean ice thickness	m
net surface mass balance	kg yr^{-1}
net basal mass balance	kg yr^{-1}
net basal mass balance for floating ice	kg yr^{-1}
net basal mass balance for grounded ice	kg yr^{-1}
average surface mass balance	m yr^{-1}
average basal mass balance for grounded ice	m yr^{-1}
average basal mass balance for floating ice	m yr^{-1}
net flux due to iceberg calving	kg yr^{-1}
net flux across grounding lines	kg yr^{-1}
maximum surface and basal velocity	m yr^{-1}

Chapter 11

Verification and Validation

MALI has been verified by a series of configurations that test different components of the code. In some cases analytic solutions are used, but other tests rely on intercomparison with community benchmarks that have been run previously by many different ice sheet models.

MALI currently includes 86 automated system regression tests that run the model for various problems with analytic solutions or community benchmarks. In addition to checking the accuracy of model answers, some of the tests check that model restarts give bit-for-bit exact answers with longer runs without restart. Some others check that the model gives bit-for-bit exact answers on different numbers of processors. All but 20 of the longer running tests are run every time new features are added to the code, and these tests each also include a check for answer changes. This chapter describes the configuration and analysis of some of the more significant tests. Detailed description of tests that can easily be run by new users are described in Chapter 12.

11.1 Halfar analytic solution

In (Halfar, 1981, 1983), Halfar described an analytic solution for the time-evolving geometry of a radially-symmetric, isothermal dome of ice on a flat bed with no accumulation flowing under the shallow ice approximation. This test provides an obvious test of the implementation of the shallow-ice velocity calculation and thickness evolution schemes in numerical ice sheet models and a way to assess model order of convergence (Bueler et al., 2005; Egholm and Nielsen, 2010). Bueler et al. (2005) showed the Halfar test is the zero accumulation member of a family of analytic solutions, but we apply the original Halfar test here.

In our application we use a dome following the analytic profile prescribed by Halfar (1983) with an initial radius of 21213.2 m and an initial height of 707.1 m. We run MALI with the shallow ice velocity solver and isothermal ice for 200 years and then compare the modeled ice thickness to the analytic solution at 200 years. We find the root mean square error in model thickness decreases as model grid spacing is decreased (Figure 11.1a). The order of convergence of 0.78, consistent with the first-order approximation used for advection.

We also use this test to assess the accuracy of simulations with variable resolution. We perform an addition run of the Halfar test using a variable resolution mesh that has 1000 m cell spacing beyond a radius of 20 km that transitions to 5000 m cell spacing at a radius of 3 km (Figure 11.1b), generated with the JIGSAW(GEO) mesh generation tool (Engwirda, 2017a,b). Root mean square error in thickness for this simulation is similar to that for the uniform 1000 m resolution case (Figure 11.1a), providing confidence in the advection scheme applied to variable resolution meshes. The variable resolution mesh has about half the cells of the 1000 m uniform resolution mesh.

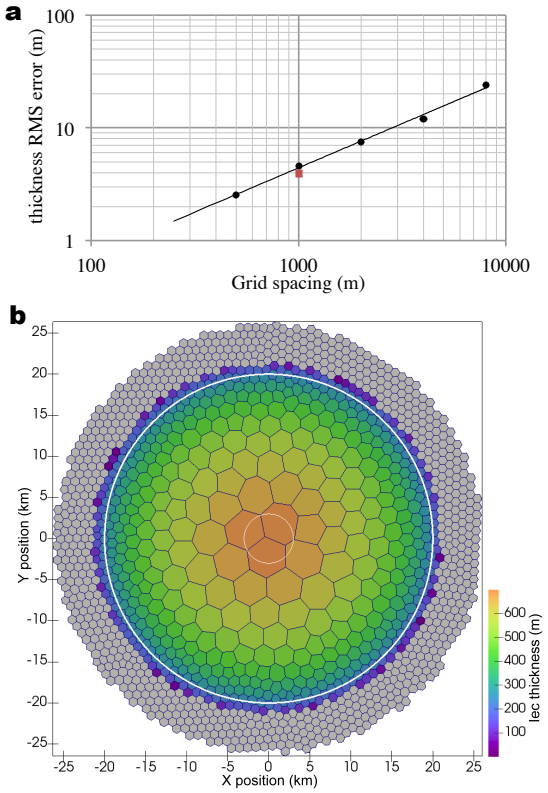


Figure 11.1: a) Root mean square error in ice thickness as a function of grid cell spacing for the Halfar dome after 200 years shown with black dots. The order of convergence is 0.78. The red square show the RMS thickness error for the variable resolution mesh shown in b) with 1000 m spacing around the margin. b) Mesh with resolution that varies linearly from 1000 m grid spacing beyond a radius of 20 km (thick white line) to 5000 m at a radius of 3 km (thin white line). The ice thickness initial condition for the Halfar problem is shown. This mesh requires 1265 cells for the 200 yr duration Halfar test case, while a uniform 1000 m resolution mesh requires 2323 cells.

11.2 EISMINT

European Ice Sheet Modeling Initiative (EISMINT) model intercomparison consisted of two phases designed to provide community benchmarks for shallow-ice models. Both phases included experiments that grow a radially symmetric ice sheet on a flat bed to steady state with a prescribed surface mass balance. The EISMINT intercomparisons test ice geometry evolution and ice temperature evolution with a variety of forcings. Bueler et al. (2007) describe an alternative tool for testing thermomechanical shallow-ice models with artificially constructed exact solutions. While their approach has the notable advantage of providing exact solutions, we have not implemented the non-physical three-dimensional compensatory heat source necessary for its implementation. While we hope to use the verification of Bueler et al. (2007) in the future, for now we use the EISMINT intercomparison suites to test our implementation of thermal evolution and thermomechanical coupling.

The first phase (Huybrechts et al., 1996) (sometimes called EISMINT1) prescribes evolving ice

geometry and temperature, but the flow rate parameter A is set to a prescribed value so there is no thermomechanical coupling. We have conducted the Moving Margin experiment with steady surface mass balance and surface temperature forcing. Following the specifications described by Huybrechts et al. (1996), we run the ice sheet to steady state over 200 ka. We use the grid spacing prescribed by Huybrechts et al. (1996) (50 km), but due to the uniform Voronoi grid of hexagons we employ, we have a slightly larger number of grid cells in our mesh (1080 vs 961). At the end of the simulation, the modeled ice thickness at the center of the dome by MALI is 2976.7 m, compared with a mean of 2978.0 ± 19.3 m for the ten three-dimensional models reported by Huybrechts et al. (1996). MALI achieves similar good agreement for basal homologous temperature at the center of the dome with a value of $-13.09°$ C, compared with $-13.34 \pm 0.56°$ C for the six models that reported temperature in Huybrechts et al. (1996).

The second phase of EISMINT (Payne et al., 2000) (sometimes called EISMINT2), uses the basic configuration of the EISMINT1 Moving Margin experiment but activates thermomechanical coupling through Eq. 8.7. Two experiments (A and F) grow an ice sheet to steady state over 200 ka from an initial condition of no ice, but with different air temperature boundary conditions. Additional experiments use the steady state solution from experiment A (the warmer air temperature case) as the initial condition to perturbations in the surface air temperature or surface mass balance forcings (experiments B, C, and D). Because these experiments are thermomechanically coupled, they test model ice dynamics and thickness and temperature evolution, as well as their coupling. There is no analytic solution to these experiments, but ten different models contributed results, yielding a range of behavior against which to compare additional models. Here we present MALI results for the five such experiments that prescribe no basal sliding (experiments A, B, C, D, F). Our tests use the same grid spacing as prescribed by Payne et al. (2000) (25 km), again with a larger number of grid cells in our mesh (4464 vs 3721).

Payne et al. (2000) report results for five basic glaciological quantities calculated by ten different models, which we have summarized here with the corresponding values calculated by MALI (Table 11.1). All MALI results fall within the range of previously reported values, except for volume change and divide thickness change in experiment C and melt fraction change in experiment D. However, these discrepancies are close to the range of results reported by Payne et al. (2000), and we consider temperature evolution and thermomechanical coupling within MALI to be consistent with community models, particularly given the difference in model grid and thickness evolution scheme.

A long-studied feature of the EISMINT2 intercomparison is the cold "spokes" that appear in the basal temperature field of all models in Experiment F and, for some models, experiment A (Payne et al., 2000; Saito et al., 2006; Bueler et al., 2007; Brinkerhoff and Johnson, 2013). MALI with shallow-ice velocity exhibits cold spokes for experiment F but not experiment A (Figure 11.2). Bueler et al. (2007) argue these spokes are a numerical instability that develops when the derivative of the strain heating term is large. Brinkerhoff and Johnson (2013) demonstrate that the model VarGlaS avoids the formation of these cold spokes. However, that model differs from previously analyzed models in several ways: it solves a three-dimensional, advective-diffusive description of an enthalpy formulation for energy conservation; it uses the Finite Element Method on unstructured meshes; conservation of momentum and energy are iterated on until they are consistent (rather than lagging energy and momentum solutions as in most other models). At present, it is unclear which combination of those features is responsible for preventing the formation of the cold spokes.

Table 11.1: EISMINT2 results for MALI shallow-ice model. For each experiment, model name "EISMINT2" refers to mean and range of models reported in Payne et al. (2000), where we assume the range reported in by Payne et al. (2000) is symmetric about the mean. For experiments B, C, and D reported values are the change from experiment A results. MALI results that lie outside the range of values in Payne et al. (2000) are italicized.

Exp.	Model	Volume 10^6 km^3	Area 10^6 km^2	Melt fraction	Divide thickness m	Divide basal temperature K
A	EISMINT2	2.128±0.0725	1.034±0.043	0.718±0.145	3688.3±48.3	255.6±1.4
	MALI	2.097	1.030	0.637	3671.8	255.2
Exp.	Model	% change	% change	% change	% change	change (K)
B	EISMINT2	-2.589±0.4735	0.0±0.0	11.836±9.3345	-4.9±0.658	4.6±0.259
	MALI	-2.258	0.0	16.832	-5.013	4.6
C	EISMINT2	-28.505±0.602	-19.515±1.777	-27.806±15.6855	-12.9±0.7505	3.7±0.3075
	MALI	*-27.529*	-20.179	-35.521	*-12.049*	3.8
D	EISMINT2	-12.085±0.618	-9.489±1.63	-1.613±2.8725	-2.2±0.266	-0.2±0.03
	MALI	-12.265	-9.459	*-5.216*	-2.092	-0.2

11.3 ISMIP-HOM

The Ice Sheet Model Intercomparison Project-Higher Order Models (ISMIP-HOM) is a set of community benchmark experiments for testing higher-order approximations of ice dynamics (Pattyn et al., 2008). Tezaur et al. (2015a) describe results from the Albany-LI velocity solver for ISMIP-HOM experiments A (flow over a bumpy bed) and C (ice stream flow). For all configurations of both tests, Albany-LI results were within one standard deviation of the mean of first-order models presented in Pattyn et al. (2008), and showed excellent agreement with the similar first-order model formulation of Perego et al. (2012). These tests only require a single diagnostic solve of velocity, and thus results through MALI match those of the standalone Albany-LI code it is using.

11.4 MISMIP3d

The Marine Ice Sheet Model Intercomparison Project-3d (MISMIP3d) is a community benchmark experiment testing grounding line migration of marine ice sheets and includes nontrivial effects in all three dimensions (Pattyn et al., 2013). The experiments use a rectangular domain that is 800 km long in the longitudinal direction and 50 km wide in the transverse direction, with the transverse direction making up half of a symmetric glacier. The bedrock forms a sloping plane below sea level. The first phase of the experiment (Stnd) is to build a steady state ice sheet from a spatially uniform positive surface mass balance, with a prescribed flow rate factor A (no temperature calculation or coupling) and prescribed basal friction for a nonlinear basal friction law. A marine ice sheet forms with an unbuttressed floating ice shelf that terminates at a fixed ice front at the edge of the domain. From this steady state, the P75R perturbation experiment reduces basal friction by a maximum of 75% across a Gaussian ellipse centered where the Stnd grounding line position crosses the symmetry axis. The perturbation is applied for 100 years, resulting in a curved grounding line that is advanced along the symmetry axis. After the completion of P75S, a reversibility experiment named P75R removes the basal friction perturbation and allows the ice sheet to relax back towards the Stnd state.

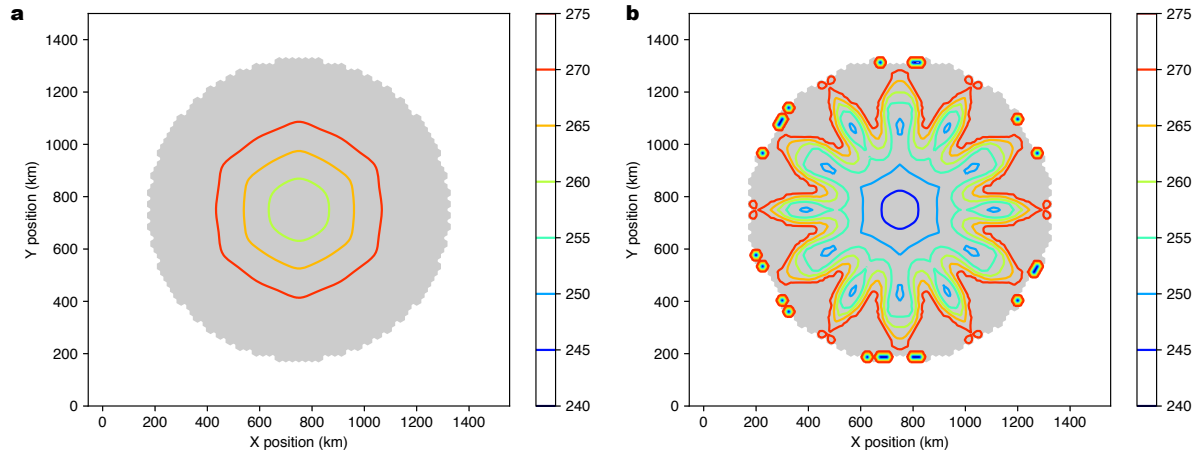


Figure 11.2: a) Basal homologous temperature (K) for EISMINT2 Experiment A. b) Same for Experiment F. Figures are plotted following Payne et al. (2000).

Pattyn et al. (2013) report results from 33 models of varying complexity and applied at resolution ranging from 0.1 to 20 km. Participating models used depth-integrated shallow-shelf or L1L1/L2L2 approximations, hybrid shallow-ice/shallow-shelf approximation, or the complete Stokes equations; there were no three-dimensional first-order approximation models included. This relatively simple experiment revealed a number of key features necessary to accurately model even a simple marine ice sheet. Insufficient grid resolution prevented reversibility of the steady state grounding line position after experiments P75S and P75R. Reversibility required grid resolution well below 1 km without a subgrid parameterization of grounding line position, and grids a couple times coarser with a grounding line parameterization (Pattyn et al., 2013; Gladstone et al., 2010). The steady state grounding line position in the Stnd experiment was dependent on the stress approximation employed, with Stokes model calculating grounding lines the farthest upstream and models that simplify or eliminate vertical shearing (e.g., shallow shelf) having grounding lines farther downstream, by up to 100 km. With these features resolved, numerical error due to grounding line motion is smaller than errors due to parameter uncertainty (Pattyn et al., 2013).

We find MALI using the Albany-LI first-order velocity solver is able to resolve the MISMIP3d experiments satisfactorily compared to the Pattyn et al. (2013) benchmark results when using a grid resolution of 500 m with grounding line parameterization. Results at 1 km resolution with grounding line parameterization are close to fully resolved. We first assess grid convergence by comparing the position of the steady state grounding line in the Stnd experiment for a range of resolutions against our highest resolution configuration of 250 m (Figure 11.3). Without a grounding line parameterization, the grounding line position appears unconverged, but with the grounding line parameterization, grid convergence occurs at 500 m resolution. The converged grounding line position for the Stnd experiment with MALI is 534 km. This represents the first results published for a three-dimensional, first-order stress approximation that we are aware of, and this grounding line position falls between that of the L1L2 and Stokes models reported by Pattyn et al. (2013), consistent with the intermediate level of approximation of our model. The dissertation work by Leguy (2015) reported similar results for the Blatter-Pattyn velocity solver with grounding line parameterization in the Community Ice Sheet Model.

Reversibility of the P75S and P75R experiments shows the same grid resolution requirement of 500 m, while the 1 km simulation with grounding line parameterization is close to reversible

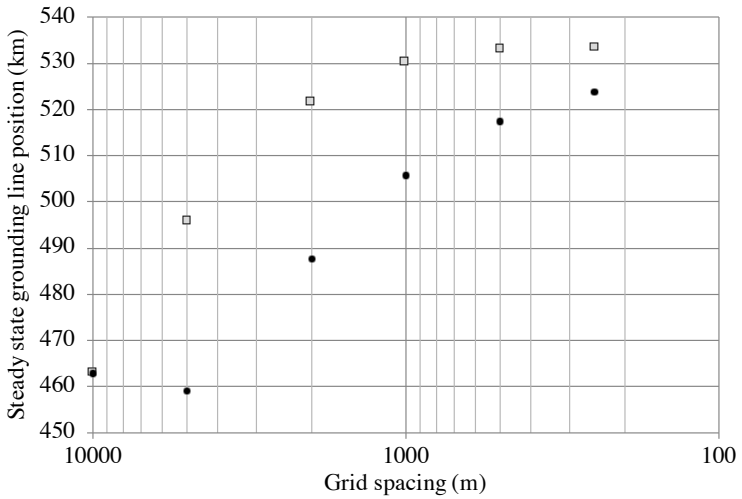


Figure 11.3: Grid resolution convergence for MISMIP3d Stnd experiment with (gray squares) and without (black circles) grounding line parameterization.

(Figure 11.4). Our highest resolution 250 m simulation without grounding line parameterization does show reversibility at the end of P75R (not shown), but the results differ somewhat from the runs with grounding line parameterization due to the differing starting position determined from the Stnd experiment. Thus for marine ice sheets with similar configuration to the MISMIP3d test, we recommend using MALI with the grounding line parameterization and a resolution of 1 km or less.

The transient results using the MALI three-dimensional first-order stress balance look most similar to those of the "SCO6" L1L2 model presented by Pattyn et al. (2013) in that it takes about 50 years for the grounding line to reach its most advanced position during P75S. In contrast, the Stokes models took notably longer and the models with reduced or missing representation of membrane stresses reached their furthest advance within the first couple decades (Pattyn et al., 2013).

In addition to MISMIP3d, we have used MALI to perform the MISMIP+ experiments (Asay-Davis et al., 2016). These results are included in the MISMIP+ results paper in preparation and not shown here.

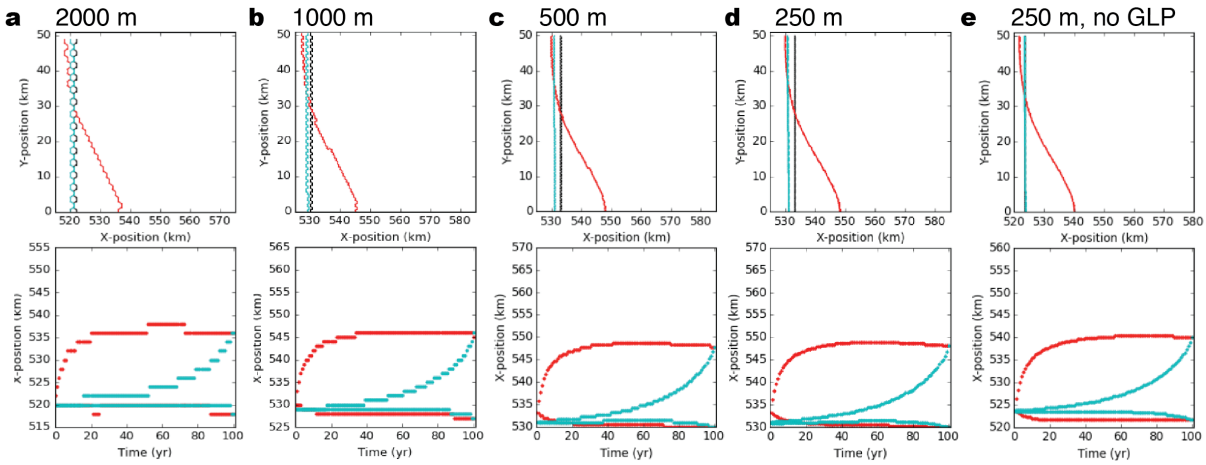


Figure 11.4: Results of the MISMIP3d P75R and P75S experiments from MALI at increasing grid resolution: a) 2000 m, b) 1000 m, c) 500 m, d) 250 m. Results for 250 m without grounding line parameterization (e) are also shown for reference. Plots follow conventions of Figures 5 and 6 in Pattyn et al. (2013). Upper plot in each subplot shows steady state grounding line positions for steady-state spin-up (black), P75S (red), and P75R (blue) experiments. Lower plot in each subplot shows grounding line position with time for P75R (red) and P75S (blue) at y=0 km (top curves) and y=50 km (bottom curves). 500 m and 250 m results are nearly identical. 250 m results without grounding line parameterization are intermediate of those at 1000 m and 2000 m resolution with grounding line parameterization.

Chapter 12

Test Cases

This chapter describes detail of a couple of simple test cases that can easily be run by new users. It does not provide an exhaustive description of all the tests that have been added to MALI. Some additional tests are summarized in Chapter 11. MPAS includes a test case system called "Configuration Of Model for Prediction Across Scales Setups" (COMPASS) located within the model source code at `testing_and_setup/compass`. We hope to include instructions for setting up COMPASS tests in future releases.

The test cases discussed below are available for download at http://mpas-dev.github.io/land_ice/download.html.

12.1 Halfar Dome

This test case describes the time evolution of a dome of ice as described by Halfar (1983). This test provide an analytic solution for a flat-bedded SIA problem.

$$\frac{\partial H}{\partial t} = \nabla \cdot (\Gamma H^{n+2} |\nabla H|^{n-1} \nabla H) \tag{12.1}$$

where n is the exponent in the Glen flow law, commonly taken as 3, and Γ is a positive constant:

$$\Gamma = \frac{2}{n+2} A (\rho g)^n \tag{12.2}$$

For $n = 3$, this reduces to:

$$H(t,r) = H_0 \left(\frac{t_0}{t}\right)^{\frac{1}{9}} \left[1 - \left(\left(\frac{t_0}{t}\right)^{\frac{1}{18}} \frac{r}{R_0}\right)^{\frac{4}{3}}\right]^{\frac{3}{7}} \tag{12.3}$$

where

$$t_0 = \frac{1}{18\Gamma} \left(\frac{7}{4}\right)^3 \frac{R_0^4}{H_0^7} \tag{12.4}$$

and H_0, R_0 are the central height of the dome and its radius at time $t = t_0$.

For more details see http://www.projects.science.uu.nl/iceclimate/karthaus/2009/more/lecturenotes/EdBueler.pdf, Bueler et al. (2005), Halfar (1983).

12.1.1 Provided Files

Our implementation of the Halfar dome has an initial radius of $R_0 = 21.2$ km and an initial thickness of $H = 707.1$ m. These values can be changed by editing `setup_dome_initial_conditions.py`.

- README:
  Information about the test case.
- namelist.landice:
  This file is used for actually running the dome test case in the MPAS land ice core. It may not include all options available to the model. See the namelist.landice.defaults file in the MPAS root directory for a list of all options available. They are also documented in Section 15.
- streams.landice:
  This file is used for specifying file input/output settings for the model.
- check_halfar_solution.py:
  This is the script to compare model results to the analytic solution.
- visualize_dome.py:
  This python script provides some general visualization of the model output. It can be used in addition to `halfar.py` for additional visualization.
- graph.info.part.* files:
  These provide grid partitioning information for running the test case on more than one processor.
- setup_dome_initial_conditions.py:
  This python script generates the dome initial condition after an empty `landice_grid.nc` file exists. If you downloaded a tar archive, you do not need to do this. However, if you want to modify the IC for some reason, you can edit and run this script.

12.1.2 Results

As the dome of ice evolves, its margin advances and its thickness decreases (there is no surface mass balance to add new mass). The script `halfar.py` will plot the modeled and analytic thickness at a specified time (Figure 12.1), as well as report model error statistics. Invoke `halfar.py --help` for details of its usage.

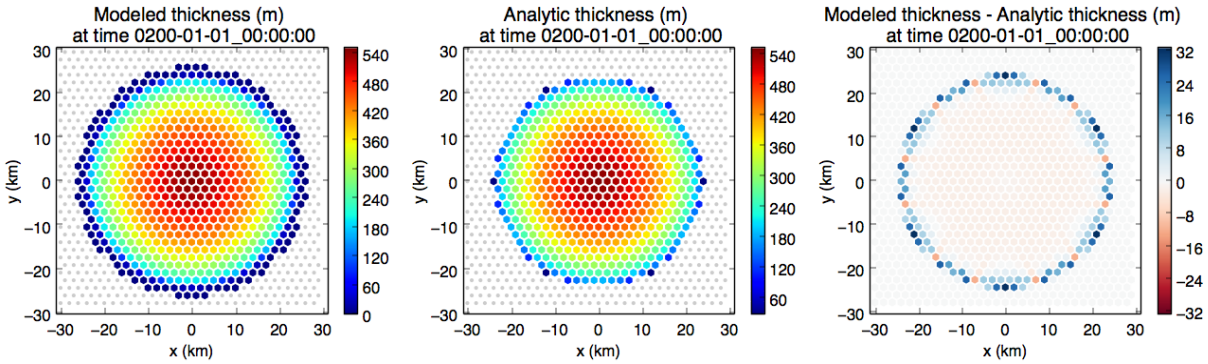


Figure 12.1: Halfar test case results after 200 years of dome evolution. This figure is generated by `halfar.py`.

12.2 EISMINT-1 Test Cases

This test case is from the European Ice Sheet Modelling INiTiative intercomparison experiments. These experiments are described at http://homepages.vub.ac.be/~phuybrec/eismint.html and in Huybrechts et al. (1996).

Currently only the Moving Margin 1 Test Case from EISMINT-1 is included.

12.2.1 Provided Files

- namelist.landice:
  This file is used for actually running the dome test case in the MPAS land ice core. It may not include all options available to the model. See the namelist.landice.defaults file in the MPAS root directory for a list of all options available. They are also documented in Section 15.

- streams.landice:
  This file is used for specifying file input/output settings for the model.

- check_output_eismint-mm1.py
  This script can be used to compare model output to results from the EISMINT intercomparison.

- graph.info.part.* files:
  These provide grid partitioning information for running the test case on more than one processor.

- setup_initial_conditions_EISMINT1-MovingMargin-1.py
  This file can be used to setup the initial conditions for the test case. If you downloaded a tar archive, you do not need to do this. However, if you want to modify the initial condition for some reason, you can edit and run this script.

12.2.2 Results

As the initial ice sheet evolves, its shape eventually reaches a steady-state with the imposed surface mass balance. The script `check_output_eismint-mm1.py` will plot the modeled thickness at a specified time, as well as compare the model results to the results from the original EISMINT

intercomparison. Invoke `check_output_eismint-mm1.py --help` for details of its usage. The script will compare the maximum ice thickness at the final time of the model output to the values reported from the models participating in the EISMINT-1 intercomparison. You should see something similar to this:

```
====================================
Max modeled thickness (m) = 2976.68217741
EISMINT models ice thickness at divide (m):
  3d models (10 of them): 2978.0 +/- 19.3
  2d models (3 of them):  2982.2 +/- 26.4
====================================

====================================
Basal homologous temperature at divide (deg C) = -13.0860572743
EISMINT models basal temperature at divide (m):
  3d models (6 of them): -13.34 +/- 0.56
====================================
```

Chapter 13

MALI within the Energy Exascale Earth System Model

MALI is the current land ice model component of the U.S. Department of Energy's *Energy Exascale Earth System Model* (E3SM, https://github.com/E3SM-Project/E3SM). E3SM is an Earth System Model with atmosphere, land, ocean, and sea ice components, linked through a coupler that passes the necessary fields (e.g., model state, mass, momentum, and energy fluxes) between the components. E3SM, which branched from the Community Earth System Model (CESM, version 1.2 beta10) in 2014, targets high resolution global simulations, and all components have a variable resolution mesh capability. The ocean (Ringler et al., 2013; Petersen et al., 2015, 2018) and sea ice (Turner et al., 2018) components are also built on the MPAS Framework. Because the coupling between E3SM and MALI is currently still fairly rudimentary, we include only a few additional details below and leave a more detailed description to future work. Having all three of these E3SM components in the MPAS framework has simplified adding and maintaining them within E3SM, because developments in the component driver code and build and configuration scripts made by one MPAS component can easily be leveraged by the others.

Physics at the ice sheet atmosphere interface are handled by the snow model within the E3SM Land Model (ELM; Zhu et al. (2018); Ricciuto et al. (2018)). ELM's snow model calculates ice sheet surface mass balance using a surface energy balance model and, at each coupling interval, MALI passes the current ice sheet extent and surface elevation through the coupler to ELM. The coupler then returns the surface mass balance and surface temperature calculated by ELM to MALI. These fields are used within MALI as boundary conditions to the mass and thermal evolution equations (Sections 8.3 and 8.4). Currently, runoff from surface melting is calculated within ELM and routed directly through E3SM's runoff model, rather than being passed to and used by MALI. The subglacial discharge model discussed above in Section 9.1 is not currently coupled to the rest of E3SM.

Ongoing and future work on MALI and E3SM coupling includes: passing subglacial discharge at terrestrial ice margins to the land runoff model in E3SM; passing surface runoff calculated in E3SM to the land ice model (for use as a source term in the subglacial hydrology model); two-way coupling between the ocean and a dynamic MALI model[1]; discharge of icebergs (solid ice flux from MALI) to the coupler and from there to the ocean and sea ice models.

[1]Coupling to a static Antarctic ice sheet with ocean circulation in sub-ice shelf cavities is supported in E3SM version 1.0.0

Chapter 14

Model Configuration

This chapter describes the configuration of the Land Ice core. The chapter covers the dimensions used in the model, the Namelist options which are used to provide run-time configurability of model options, the variables used in the model, and the usage of run-time I/O streams.

Chapter 15

Namelist options

Embedded links point to more detailed namelist information in the appendix.

15.1 velocity_solver

The velocity_solver namelist record controls which velocity solver is used and options associated with velocity solvers.

Name	Description
config_velocity_solver	Selection of the method for solving ice velocity. 'L1L2', 'FO', and 'Stokes' require compiling with external dycores. 'none' skips the calculation of velocity so the velocity field will be 0 or set to a field read from an input file. 'simple' gives a simple prescribed velocity field computed at initialization.
config_sia_tangent_slope_calculation	Selection of the method for calculating the tangent component of surface slope at edges needed by the SIA velocity solver. 'from_vertex_barycentric' interpolates upperSurface values from cell centers to vertices using the barycentric interpolation routine in operators (mpas_cells_to_points_using_baryweights) and then calculates the slope between vertices. It works for obtuse triangles, but will not work correctly across the edges of periodic meshes. 'from_vertex_barycentric_kiteareas' interpolates upperSurface values from cell centers to vertices using barycentric interpolation based on kiterea values and then calculates the slope between vertices. It will work across the edges of periodic meshes, but will not work correctly for obtuse triangles. 'from_normal_slope' uses the vector operator mpas_tangential_vector_1d to calculate the tangent slopes from the normal slopes on the edges of the adjacent cells. It will work for any mesh configuration, but is the least accurate method.
config_flowParamA_calculation	Selection of the method for calculating the flow law parameter A. If 'constant' is selected, the value is set to config_default_flowParamA. The other options are calculated from the temperature field. This calculation only applies if config_velocity_solver is set to 'sia'. For the 'FO' velocity solver, this is set in the albany_input.xml file.

Name	Description (Continued)
config_do_velocity_reconstruction_for_external_dycore	By default, external, higher-order dycores return the uReconstructX and uReconstructY fields (which are the native locations of their FEM solution). If this option is set to .true., uReconstructX and uReconstructY will be calculated by MPAS using framework's vector reconstruction routines based on the values of normalVelocity supplied by the external dycore. This provides a way to test the calculation of normalVelocity in the interface.
config_simple_velocity_type	Selection of the type of simple velocity field computed at initialization when config_velocity_solver = 'simple'. See mode_forward/mpas_li_velocity_simple.F for details of what the options do.
config_use_glp	If true, then apply Albany's grounding line parameterization
config_beta_use_effective_pressure	If true, then multiply beta by effective pressure before passing to Albany. This allows, e.g., a Weertman basal friction law with an effective pressure term. Note that basal friction still needs to be selected in Albany xml file.

15.2 advection

The advection namelist record controls options assocated with advection of thickness and tracers. Tracer advection is not currently supported.

Name	Description
config_thickness_advection	Selection of the method for advecting thickness ('fo' = first-order upwinding).
config_tracer_advection	Selection of the method for advecting tracers.

15.3 calving

The calving namelist record controls options assocated with calving of floating ice.

Name	Description
config_calving	Selection of the method for calving ice (as defined further below).
config_calving_topography	Defines the topographic height below which ice calves (for topographic_threshold option).
config_calving_thickness	Defines the ice thickness below which ice calves (for thickness_threshold option).
config_calving_eigencalving_parameter_source	Source of the eigencalving parameter value

Name	Description (Continued)
config_calving_eigencalving_parameter_scalar_value	Value of eigencalving parameter if taken as a scalar by option config_calving_eigencalving_parameter_source. (Default value is 1.0e9 m a converted to units used here.)
config_data_calving	Select whether or not to configure calving in a 'data' model mode (calc. calving flux but do not update ice geometry)
config_calving_timescale	Defines the timescale for calving. The fraction of eligible ice that calves is min(dt/calving_timescale, 1.0). A value of 0 means that all eligible ice calves.
config_restore_calving_front	If true, then restore the calving front to its initial position. If ice grows beyond the initial extent, it is removed. If ice shrinks to an extent behind the initial extent, those locations are filled with thin ice (defined as 1/10th the value of config_dynamic_thickness). Note that this violates conservation of mass and energy.

15.4 thermal_solver

The thermal_solver namelist record controls options assocated with temperature evolution.

Name	Description
config_thermal_solver	Selection of the method for the vertical thermal solver (possible values are described further below).
config_thermal_calculate_bmb	Determines if basal and internal melting calculated by the thermal solver should contribute to basal mass balance or be ignored.
config_temperature_init	Selection of the method for initializing the ice temperature (as described further below).
config_thermal_thickness	Defines the minimum ice thickness for conducting thermal calculations. Ice thinner than this value is ignored by the thermal solver.
config_surface_air_temperature_source	Selection of the method for setting the surface air temperature. 'constant' uses the value set by config_surface_air_temperature_value. 'file' reads the field from an input or forcing file or ESM coupler. 'lapse' uses the value of config_surface_air_temperature_value at elevation 0 with a lapse rate applied from config_surface_air_temperature_lapse_rate.
config_surface_air_temperature_value	Constant value of the surface air temperature.
config_surface_air_temperature_lapse_rate	Lapse rate to apply to surface air temperature when config_surface_air_temperature_source='lapse'. Positive values lead to colder temperatures at higher elevations.
config_basal_heat_flux_source	Selection of the method for setting the basal heat flux.
config_basal_heat_flux_value	Constant value of the basal heat flux (positive upward).
config_basal_mass_bal_float	Selection of the method for computing the basal mass balance of floating ice. 'none' sets the basalMassBal field to 0 everywhere. 'file' uses without modification whatever value was read in through an input or forcing file or the value set by an ESM coupler. 'constant', 'mismip', 'seroussi' use hardcoded fields defined in the code.

Name	Description (Continued)
config_basal_mass_bal_-seroussi_amplitude	amplitude on the depth adjustment applied to the Seroussi subglacial melt parameterization
config_basal_mass_bal_-seroussi_period	period of the periodic depth adjustment applied to the Seroussi subglacial melt parameterization
config_basal_mass_bal_-seroussi_phase	phase of the periodic depth adjustment applied to the Seroussi subglacial melt parameterization. Units are cycles, i.e., 0-1
config_bmlt_float_flux	Value of the constant heat flux applied to the base of floating ice (positive upward).
config_bmlt_float_xlimit	x value defining region where bmlt_float_flux is applied; melt only where abs(x) is greater than xlimit.

15.5 physical_parameters

The physical_parameters namelist record sets scalar physical parameters and constants within the land ice model.

Name	Description
config_ice_density	ice density to use (assumed constant and uniform)
config_ocean_density	ocean density to use for calculating floatation (assumed constant and uniform)
config_sea_level	sea level to use for calculating floatation (assumed constant and uniform)
config_default_flowParamA	Defines the default value of the flow law parameter A to be used if it is not being calculated from ice temperature. This value will be used by either the sia or FO velocity solver if they are respectively configured to use a scalar A value. Defaults to the SI representation of 1.0e-16 yr^{-1} Pa^{-3}.
config_enhancementFactor	multiplier on the flow parameter A
config_flowLawExponent	Defines the value of the Glen flow law exponent, n. This value will be used by either the sia or FO velocity solver. A value other than 3.0 is untested.
config_dynamic_thickness	Defines the ice thickness below which dynamics are not calculated (and hence ice velocity is set to 0).

15.6 time_integration

The time integration namelist record controls parameters that pertain to all time-stepping methods. At present, Forward Euler is the only time integration method implemented.

Name	Description
config_dt	Length of model time step defined as a time interval.
config_time_integration	Time integration method (currently, only forward Euler is supported).
config_adaptive_timestep	Determines if the time step should be adjusted based on the CFL condition or should be steady in time. If true, the config_dt_* options are ignored.
config_min_adaptive_timestep	The minimum allowable time step in seconds. If the CFL condition dictates the time step should be shorter than this, then the model aborts.
config_max_adaptive_timestep	The maximum allowable time step in seconds. If the allowable time step determined by the adaptive CFL calculation is longer than this, then the model will specify config_max_adaptive_timestep as the time step instead. Defaults to 100 years (in seconds).
config_adaptive_timestep_CFL_fraction	A multiplier on the minimum allowable time step calculated from the CFL condition. (Setting to 1.0 may be unstable, so smaller values are recommended.)
config_adaptive_timestep_include_DCFL	Option of whether to include the diffusive CFL condition in the determination of the maximum allowable timestep. The diffusive CFL condition at any location is estimated based on the local ice flux and surface slope.
config_adaptive_timestep_force_interval	If adaptive timestep is enabled, the model will ensure a timestep ends at multiples of this interval. This is useful for ensuring that model output is written at a specific desired interval (rather than the closest time after) or when running coupled to an earth system model that expects a certain interval.

15.7 time_management

General time management is handled by the time_management namelist record. Included options handle time-related parts of MPAS, such as the calendar type and if the simulation is a restart or not.

Users should use this record to specify the beginning time of the simulation, and either the duration or the end of the simulation. Only the end or the duration need to be specified as the other is derived within MPAS from the beginning time and other specified one.

If both the run duration and stop time are specified, run duration is used in place of stop time.

Name	Description

Name	Description (Continued)
config_do_restart	Determines if the initial conditions should be read from a restart file, or an input file. To perform a restart, set this to true in the namelist.input file. The restart time will be read from config_start_time (which can be set to 'file' to have the restart time read automatically from the file defined by config_restart_timestamp_name). A restart will read everything from the restart file - no information is read from the 'input' stream. It will perform a run normally, except velocity will not be solved on a restart.
config_restart_timestamp_name	Path to the filename for restart timestamps to be read and written from.
config_start_time	Timestamp describing the initial time of the simulation. If it is set to 'file', the initial time is read from the filename specified by config_restart_timestamp_name (defaults to 'restart_timestamp').
config_stop_time	Timestamp describing the final time of the simulation. If it is set to 'none' the final time is determined from config_start_time and config_run_duration. If config_run_duration is also specified, it takes precedence over config_stop_time. Set config_stop_time to be equal to config_start_time (and config_run_duration to 'none') to perform a diagnostic solve only.
config_run_duration	Timestamp describing the length of the simulation. If it is set to 'none' the duration is determined from config_start_time and config_stop_time. config_run_duration overrides inconsistent values of config_stop_time. If a time value is specified for config_run_duration, it must be greater than 0.
config_calendar_type	Selection of the type of calendar that should be used in the simulation.

15.8 io

The io namelist record provides options for modifications to the I/O system of MPAS. These include frequency, file name, and parallelization options.

Name	Description
config_stats_interval	Integer specifying interval (number of timesteps) for writing global/local statistics. If set to 0, then statistics are not written (except perhaps at startup, as determined by 'config_write_stats_on_startup'). Applies to statistics written to log file and not analysis member output written to netCDF files.
config_write_stats_on_startup	Logical flag determining if statistics should be written prior to the first time step. Applies to statistics written to log file and not analysis member output written to netCDF files.
config_stats_cell_ID	global ID for the cell selected for local statistics/diagnostics. Applies to statistics written to log file and not analysis member output written to netCDF files.

Name	Description (Continued)
config_write_output_on_startup	Logical flag determining if an output file should be written prior to the first time step.
config_pio_num_iotasks	Integer specifying how many IO tasks should be used within the PIO library. A value of 0 causes all MPI tasks to also be IO tasks. IO tasks are required to write contiguous blocks of data to a file. Optimal performance is typically found by having 1-2 tasks per node performing I/O. To do so, config_pio_num_iotasks must be manually set in conjunction with config_pio_stride as appropriate for the processor layout used. For example, running on 240 processors on a machine with 24 processors per node, setting config_pio_num_iotasks=20 and config_pio_stride=12 would configure two I/O tasks per node.
config_pio_stride	Integer specifying the stride of each IO task. See config_pio_num_iotasks for details.
config_year_digits	Integer specifying the number of digits used to represent the year in time strings.
config_output_external_velocity_solver_data	If .true., external velocity solvers (if enabled) will write their own output data in addition to any MPAS output that is configured.
config_write_albany_ascii_mesh	Logical flag determining if ascii mesh files will be created. These files are written in a format that can be used by the standalone Albany velocity solver for optimization. If .true., the model initializes, writes the mesh files, and then terminates.

15.9 decomposition

MPAS handles decomposing all variables into computational blocks. The decomposition used needs to be specified at run time and is computed by an external tool (e.g. metis). Additionally, MPAS supports multiple computational blocks per MPI process, and the user may specify an additional decomposition file which can specify the assignment of blocks to MPI processes. Run-time parameters that control the run-time decomposition used are specified within the decomposition namelist record.

Name	Description
config_num_halos	Determines the number of halo cells extending from a blocks owned cells (Called the 0-Halo). The default first-order upwinding advection requires a minimum of 2. Note that a minimum of 3 is required for incremental remapping advection on a quad mesh or for FCT advection (neither of which is currently supported for land ice).
config_block_decomp_file_prefix	Defines the prefix for the block decomposition file. Can include a path. The number of blocks is appended to the end of the prefix at run-time.
config_number_of_blocks	Determines the number of blocks a simulation should be run with. If it is set to 0, the number of blocks is the same as the number of MPI tasks at run-time.

Name	Description (Continued)
config_explicit_proc_decomp	Determines if an explicit processor decomposition should be used. This is only useful if multiple blocks per processor are used.
config_proc_decomp_file_prefix	Defines the prefix for the processor decomposition file. This file is only read if config_explicit_proc_decomp is .true. The number of processors is appended to the end of the prefix at run-time.

15.10 debug

At run-time a user can enable debugging features within MPAS-Land Ice. Currently the only debug option is to print more detailed information about thickness advection. Potential future debug options would be to include disabling of any tendencies to help determine why an issue might be happening; various checks on certain fields; and the ability to prescribe both a thickness and velocity field at run-time which are constant throughout a simulation. All options that control these debugging features are specified within the debug namelist record.

Name	Description
config_print_thickness_advection_info	Prints additional information about thickness advection.
config_print_calving_info	Prints additional information about calving physics (if enabled).
config_print_thermal_info	Prints additional information about thermal calculations (if enabled).
config_always_compute_fem_grid	Always compute finite-element grid information for external dycores rather than only doing so when the ice extent changes.
config_print_velocity_cleanup_details	After velocity is calculated there are a few checks for appropriate values in certain geometric configurations. Setting this option to .true. will cause detailed information about those adjustments to be printed.

15.11 subglacial_hydro

The subglacial_hydro namelist record controls options assocated with the subglacial hydrology model.

Name	Description
config_SGH	activate subglacial hydrology model
config_SGH_adaptive_timestep_fraction	fraction of adaptive CFL timestep to use

Name	Description (Continued)
config_SGH_max_adaptive_timestep	The maximum allowable time step in seconds. If the allowable time step determined by the adaptive CFL calculation is longer than this, then the model will specify config_SGH_max_adaptive_timestep as the time step instead. Defaults to 100 years (in seconds).
config_SGH_tangent_slope_calculation	Selection of the method for calculating the tangent component of slope at edges. 'from_vertex_barycentric' interpolates scalar values from cell centers to vertices using the barycentric interpolation routine in operators (mpas_cells_to_points_using_baryweights) and then calculates the slope between vertices. It works for obtuse triangles, but will not work correctly across the edges of periodic meshes. 'from_vertex_barycentric_kiteareas' interpolates scalar values from cell centers to vertices using barycentric interpolation based on kiterea values and then calculates the slope between vertices. It will work across the edges of periodic meshes, but will not work correctly for obtuse triangles. 'from_normal_slope' uses the vector operator mpas_tangential_vector_1d to calculate the tangent slopes from the normal slopes on the edges of the adjacent cells. It will work for any mesh configuration, but is the least accurate method.
config_SGH_pressure_calc	Selection of the method for calculating water pressure. 'cavity' closes the hydrology equations by assuming cavities are always completely full. 'overburden' assumes water pressure is always equal to ice overburden pressure.
config_SGH_alpha	power of alpha parameter in subglacial water flux formula
config_SGH_beta	power of beta parameter in subglacial water flux formula
config_SGH_conduc_coeff	conductivity coefficient for subglacial water flux
config_SGH_till_drainage	background subglacial till drainage rate
config_SGH_advection	Advection method for SGH. 'fo'=first-order upwind; 'fct'=flux-corrected transport. FCT currently not enabled.
config_SGH_bed_roughness	cavitation coefficient
config_SGH_bed_roughness_max	bed roughness scale
config_SGH_creep_coefficient	creep closure coefficient
config_SGH_englacial_porosity	notional englacial porosity
config_SGH_till_max	maximum water thickness in subglacial till
config_SGH_chnl_active	activate channels in subglacial hydrology model
config_SGH_chnl_alpha	power of alpha parameter in subglacial water flux formula (in channels)
config_SGH_chnl_beta	power of beta parameter in subglacial water flux formula (in channels)
config_SGH_chnl_conduc_coeff	conductivity coefficient (in channels)
config_SGH_chnl_creep_coefficient	creep closure coefficient (in channels)
config_SGH_incipient_channel_width	width of sheet beneath/around channel that contributes to melt within the channel
config_SGH_include_pressure_melt	whether to include the pressure melt term in the rate of channel opening
config_SGH_shmip_forcing	calculate time-varying forcing specified by SHMIP experiments C or D

Name	Description (Continued)
config_SGH_basal_melt	source for the basalMeltInput term. 'file' takes whatever field was input and performs no calculation. 'thermal' uses the groundedBasalMassBal field calculated by the thermal model. 'basal_heat' calculates a melt rate assuming the entirety of the basal heat flux (basalFrictionFlux+basalHeatFlux) goes to melting ice at the bed. This is calculated in the SGH module and is independent of any calculations in the thermal model.

15.12 AM_globalStats

The AM_globalStats namelist record controls options assocated with the global statistics analysis members.

Name	Description
config_AM_globalStats_enable	If true, landice analysis member globalStats is called.
config_AM_globalStats_compute_interval	Timestamp determining how often analysis member computations should be performed.
config_AM_globalStats_stream_name	Name of the stream that the globalStats analysis member should be tied to.
config_AM_globalStats_compute_on_startup	Logical flag determining if analysis member computations occur on start-up.
config_AM_globalStats_write_on_startup	Logical flag determining if an analysis member write occurs on start-up.

15.13 AM_regionalStats

The AM_regionalStats namelist record controls options assocated with the regional statistics analysis members.

Name	Description
config_AM_regionalStats_enable	If true, landice analysis member regionalStats is called.
config_AM_regionalStats_compute_interval	Timestamp determining how often analysis member computations should be performed.
config_AM_regionalStats_stream_name	Name of the stream that the regionalStats analysis member should be tied to.
config_AM_regionalStats_compute_on_startup	Logical flag determining if analysis member computations occur on start-up.
config_AM_regionalStats_write_on_startup	Logical flag determining if an analysis member write occurs on start-up.

Chapter 16

Dimensions

Name	Units	Description
nCells	unitless	The number of polygons in the primary grid.
nEdges	unitless	The number of edge midpoints in either the primary or dual grid.
maxEdges	unitless	The largest number of edges any polygon within the grid has.
maxEdges2	unitless	Two times the largest number of edges any polygon within the grid has.
nVertices	unitless	The total number of cells in the dual grid. Also the number of corners in the primary grid.
ONE	unitless	The number one as a dimension.
TWO	unitless	The number two as a dimension.
R3	unitless	The number three as a dimension.
vertexDegree	unitless	The number of cells or edges touching each vertex.
nVertLevels	unitless	The number of levels in the vertical direction. All vertical levels share the same horizontal locations.
nVertInterfaces	unitless	The number of interfaces in the vertical direction.
maxTracersAdvect	unitless	The maximum number of tracers to be advected.
nRegions	unitless	The number of regions used for AM_regionalStats
nRegionGroups	unitless	The number of region groups used for AM_regionalStats
maxRegionsInGroup	unitless	The maximum number of regions in a region group used for AM_regionalStats

Chapter 17

Variable definitions

Embedded links point to more detailed variable information in the appendix.

17.1 mesh

The mesh data type contains a single time level. The fields inside the mesh structure are not assumed to be time dependent. This data structure contains fields that describe the mesh, and the connectivity of the mesh. Most of the fields contained in this structure are shared throughout all MPAS cores. Additionally, a few Land Ice specific variables (that are time-independent) are stored here, but may be moved in the future.

Name	Description
latCell	Latitude location of cell centers in radians.
lonCell	Longitude location of cell centers in radians.
xCell	X Coordinate in cartesian space of cell centers.
yCell	Y Coordinate in cartesian space of cell centers.
zCell	Z Coordinate in cartesian space of cell centers.
indexToCellID	List of global cell IDs.
latEdge	Latitude location of edge midpoints in radians.
lonEdge	Longitude location of edge midpoints in radians.
xEdge	X Coordinate in cartesian space of edge midpoints.
yEdge	Y Coordinate in cartesian space of edge midpoints.
zEdge	Z Coordinate in cartesian space of edge midpoints.
indexToEdgeID	List of global edge IDs.
latVertex	Latitude location of vertices in radians.
lonVertex	Longitude location of vertices in radians.
xVertex	X Coordinate in cartesian space of vertices.
yVertex	Y Coordinate in cartesian space of vertices.
zVertex	Z Coordinate in cartesian space of vertices.
indexToVertexID	List of global vertex IDs.
nEdgesOnCell	Number of edges that border each cell.
nEdgesOnEdge	Number of edges that surround each of the cells that straddle each edge. These edges are used to reconstruct the tangential velocities.
cellsOnEdge	List of cells that straddle each edge.
edgesOnCell	List of edges that border each cell.

Name	Description (Continued)
edgesOnEdge	List of edges that border each of the cells that straddle each edge.
cellsOnCell	List of cells that neighbor each cell.
verticesOnCell	List of vertices that border each cell.
verticesOnEdge	List of vertices that straddle each edge.
edgesOnVertex	List of edges that share a vertex as an endpoint.
cellsOnVertex	List of cells that share a vertex.
weightsOnEdge	Reconstruction weights associated with each of the edgesOnEdge.
dvEdge	Length of each edge, computed as the distance between verticesOnEdge.
dcEdge	Length of each edge, computed as the distance between cellsOnEdge.
angleEdge	Angle the edge normal makes with local eastward direction.
areaCell	Area of each cell in the primary grid.
areaTriangle	Area of each cell (triangle) in the dual grid.
kiteAreasOnVertex	Area of the portions of each dual cell that are part of each cellsOnVertex.
meshDensity	The value of the generating density function at each cell center.
localVerticalUnitVectors	Unit surface normal vectors defined at cell centers.
edgeNormalVectors	Normal vector defined at an edge.
cellTangentPlane	The two vectors that define a tangent plane at a cell center.
coeffs_reconstruct	Coefficients to reconstruct velocity vectors at cell centers.
layerThicknessFractions	Fractional thickness of each sigma layer
layerCenterSigma	Sigma (fractional) level at center of each layer
layerInterfaceSigma	Sigma (fractional) level at interface between each layer (including top and bottom)
edgeSignOnCell	Sign of edge contributions to a cell for each edge on cell. Used for bit-reproducible loops. Represents directionality of vector connecting cells.
edgeSignOnVertex	Sign of edge contributions to a vertex for each edge on vertex. Used for bit-reproducible loops. Represents directionality of vector connecting vertices.
cellProcID	processor number for each cell
baryCellsOnVertex	Cell center indices to use for interpolating from cell centers to vertex locations. Note these are local indices!
baryWeightsOnVertex	Weights to interpolate from cell centers to vertex locations. Each weight is used with the corresponding cell center index indentified by baryCellsOnVertex.
wachspressWeightVertex	Wachspress weights used to interpolate from vertices to cell centers.
xtime	model time, with format 'YYYY-MM-DD_HH:MM:SS'
deltat	time step length, in seconds. Value on a given time is the value used between the previous time level and the current time level.
allowableDtACFL	The maximum allowable time step based on the advective CFL condition. Value on a given time is the value appropriate for between the previous time level and the current time level.
allowableDtDCFL	The maximum allowable time step based on the diffusive CFL condition. Value on a given time is the value appropriate for between the previous time level and the current time level.
simulationStartTime	start time of first simulation, with format 'YYYY-MM-DD_HH:MM:SS'
daysSinceStart	Time since simulationStartTime in days, for plotting
timestepNumber	time step number. initial time is 0.

17.2 geometry

The geometry data structure contains fields related to ice sheet geometry.

Name	Description
bedTopography	Elevation of ice sheet bed. Once isostasy is added to the model, this should become a state variable.
thickness	ice thickness
layerThickness	layer thickness
lowerSurface	elevation at bottom of ice
upperSurface	elevation at top of ice
layerThicknessEdge	layer thickness on cell edges
dHdt	diagnostic field of rate of thickness change with time (dH/dt). This includes all processes (flux divergence, SMB, BMB, calving, etc.) because it is calculated as the new thickness minus the old thickness divided by the time step.
thicknessOld	ice thickness from previous time level (only used to calculate thicknessTendency)
dynamicThickening	diagnostic field of dynamic thickening rate (calculated as negative of flux divergence)
cellMask	bitmask indicating various properties about the ice sheet on cells. cellMask only needs to be a restart field if config_allow_additional_advance = false (to keep the mask of initial ice extent)
edgeMask	bitmask indicating various properties about the ice sheet on edges.
vertexMask	bitmask indicating various properties about the ice sheet on vertices.
sfcMassBal	applied surface mass balance
basalMassBal	applied basal mass balance
groundedBasalMassBal	Basal mass balance on grounded regions
floatingBasalMassBal	Basal mass balance on floating regions
calvingThickness	thickness of ice that calves on a given timestep (less than or equal to ice thickness)
eigencalvingParameter	proportionality constant K2+- used in eigencalving formulation
calvingVelocity	rate of calving front retreat due to calving, represented as a velocity normal to the calving front (in the x-y plane). This retreat rate is converted from a flux to a rate in the code requiredCalvingVolumeRate.
requiredCalvingVolumeRate	total volume of ice that needs to be removed based on eigencalving rate at this margin cell
uncalvedVolume	volume of ice that was left uncalved from required calving flux due to only applying flux over immediate neighbors (diagnostic field to assess if this limitation is a problem)
basalWaterThickness	thickness of basal water
restoreThickness	thickness of ice added when the config_restore_calving_front option is set to .true. (in order to maintain the calving front at its initial position)

Name	Description (Continued)
normalSlopeEdge	normal surface slope on edges
apparentDiffusivity	apparent diffusivity at cell centers (estimated based on the local ice flux and surface slope)
upperSurfaceVertex	elevation at top of ice on vertices
tangentSlopeEdge	tangent surface slope on edges
slopeEdge	surface slope magnitude on edges

17.3 velocity

The velocity data structure includes fields related to ice velocity and dynamics.

Name	Description
flowParamA	flow law parameter, A, used by shallow-ice velocity solver
normalVelocity	horizonal velocity, normal component to an edge, layer interface
layerNormalVelocity	horizonal velocity, normal component to an edge, layer midpoint
normalVelocityInitial	horizonal velocity, normal component to an edge, computed at initialization
uReconstructX	x-component of velocity reconstructed on cell centers. Also, for higher-order dycores, on input: value of the x-component of velocity that should be applied where dirichletVelocityMask==1.
uReconstructY	y-component of velocity reconstructed on cell centers. Also, for higher-order dycores, on input: value of the y-component of velocity that should be applied where dirichletVelocityMask==1.
uReconstructZ	z-component of velocity reconstructed on cell centers
uReconstructZonal	zonal velocity reconstructed on cell centers
uReconstructMeridional	meridional velocity reconstructed on cell centers
surfaceSpeed	ice surface speed reconstructed at cell centers
basalSpeed	ice basal speed reconstructed at cell centers
beta	input value of basal traction parameter for sliding law used with first-order momentum balance solver (NOTE non-SI units)
betaSolve	value of basal traction parameter for sliding law used with first-order momentum balance solver (NOTE non-SI units); differs from beta due to any necessary adjustments made for internal consistency (e.g., zeroed out where the ice is found to be floating)
exx	x-component of surface strain rate
eyy	y-component of surface strain rate
exy	shear component of surface strain rate
eTheta	orientation of principal surface strain rate
eyx	shear component of surface strain rate
eMax	magnitude of first principal surface strain rate
eMin	magnitude of second principal surface strain rate
anyDynamicVertexMaskChanged	flag needed by external velocity solvers that indicates if the region to solve on the block's domain has changed (treated as a logical)
dirichletVelocityMask	mask of where Dirichlet boundary conditions should be applied to the velocity solution. 1 means apply a Dirichlet boundary condition, 0 means do not. (higher-order dycores only)

Name	Description (Continued)
dirichletMaskChanged	flag needed by external velocity solvers that indicates if the Dirichlet boundary condition mask has changed (treated as a logical)
floatingEdges	edges which are floating have a value of 1. non floating edges have a value of 0.

17.4 observations

The observations data structure includes fields related to observations of ice sheet state. None of these are currently used internally by the model, but can be written out in Exodus format to be used as input to Albany's optimization capability.

Name	Description
observedSurfaceVelocityX	X-component of observed surface velocity
observedSurfaceVelocityY	Y-component of observed surface velocity
observedSurfaceVelocity-Uncertainty	uncertainty in observed surface velocity magnitude
observedThicknessTendency	observed tendency in thickness (dH/dt)
observedThicknessTendency-Uncertainty	uncertainty in observed tendency in thickness (dH/dt)
sfcMassBalUncertainty	uncertainty in observed surface mass balance
thicknessUncertainty	uncertainty in observed thickness
floatingBasalMassBalUncertainty	uncertainty in observed floating basal mass balance

17.5 thermal

The thermal data structure includes fields related to ice temperature and thermodynamics.

Name	Description
temperature	interior ice temperature
waterfrac	interior ice water fraction
enthalpy	interior ice enthalpy
surfaceAirTemperature	air temperature at the ice sheet surface
surfaceTemperature	temperature at upper ice service
basalTemperature	temperature at lower ice surface
pmpTemperature	pressure melt temperature
basalPmpTemperature	pressure melt temperature at lower ice surface
surfaceConductiveFlux	conductive heat flux at the upper ice surface (positive downward)
basalConductiveFlux	conductive heat flux at the lower ice surface (positive downward)
basalHeatFlux	basal heat flux into the ice (positive upward)

Name	Description (Continued)
basalFrictionFlux	basal frictional heat flux into the ice (positive upward)
heatDissipation	interior heat dissipation rate, divided by rhoi*cp_ice

17.6 scratch

The scratch data structure includes reusable fields that are used as temporary arrays within the code.

Name	Description
iceCellMask	mask set to 1 in cells where some criterion is satisfied and 0 otherwise
iceCellMask2	mask set to 1 in cells where some criterion is satisfied and 0 otherwise
iceCellMask3	mask set to 1 in cells where some criterion is satisfied and 0 otherwise
iceEdgeMask	mask set to 1 for edges adjacent to ice-covered cells and 0 otherwise
workLevelCell	generic work array with dimensions of (nVertLevels nCells)
workLevelEdge	generic work array with dimensions of (nVertLevels nEdges)
workLevelVertex	generic work array with dimensions of (nVertLevels nVertices)
workCell	generic work array with dimensions of (nCells)
workCell2	generic work array with dimensions of (nCells)
workCell3	generic work array with dimensions of (nCells)
workTracerCell	generic work array with dimensions of (maxTracersAdvect nCells)
workTracerCell2	generic work array with dimensions of (maxTracersAdvect nCells)
workTracerLevelCell	generic work array with dimensions of (maxTracersAdvect nVertLevels nCells)
workTracerLevelCell2	generic work array with dimensions of (maxTracersAdvect nVertLevels nCells)
slopeCellX	x-component of slope on cell centers
slopeCellY	y-component of slope on cell centers
vertexIndices	local indices of each vertex

17.7 regions

The regions data structure includes fields related to regions defined for use with regional statistics analysis members.

Name	Description
regionCellMasks	masks set to 1 in cells that fall within a given region and 0 otherwise

17.8 hydro

The hydro data structure includes fields related to the subglacial hydrology model.

Name	Description
waterThickness	water layer thickness in subglacial hydrology system
waterThicknessOld	water layer thickness in subglacial hydrology system from previous time step
waterThicknessTendency	rate of change in water layer thickness in subglacial hydrology system
tillWaterThickness	water layer thickness in subglacial till
tillWaterThicknessOld	water layer thickness in subglacial till from previous time step
waterPressure	pressure in subglacial hydrology system
waterPressureOld	pressure in subglacial hydrology system from previous time step
waterPressureTendency	tendency in pressure in subglacial hydrology system
basalMeltInput	basal meltwater input to subglacial hydrology system
externalWaterInput	external water input to subglacial hydrology system
frictionAngle	subglacial till friction angle
effectivePressure	effective ice pressure in subglacial hydrology system
hydropotential	hydropotential in subglacial hydrology system
waterFlux	total water flux in subglacial hydrology system
waterFluxMask	mask indicating how to handle fluxes on each edge: 0=calculate based on hydropotential gradient; 1=allow outflow based on hydropotential gradient, but no inflow (NOT YET IMPLEMENTED); 2=zero flux
waterFluxAdvec	advective water flux in subglacial hydrology system
waterFluxDiffu	diffusive water flux in subglacial hydrology system
waterVelocity	water velocity in subglacial hydrology system
waterVelocityCellX	subglacial water velocity reconstructed on cell centers, x-component
waterVelocityCellY	subglacial water velocity reconstructed on cell centers, y-component
effectiveConducEdge	effective Darcy hydraulic conductivity on edges in subglacial hydrology system
waterThicknessEdge	water layer thickness on edges in subglacial hydrology system
waterThicknessEdgeUpwind	water layer thickness of cell upwind of edge in subglacial hydrology system
diffusivity	diffusivity of water sheet in subglacial hydrology system
hydropotentialBase	hydropotential in subglacial hydrology system without water thickness contribution
hydropotentialBaseVertex	hydropotential without water thickness contribution on vertices. Only used for some choices of config_SGH_tangent_slope_calculation.
hydropotentialBaseSlopeNormal	normal component of gradient of hydropotentialBase
hydropotentialBaseSlopeTangent	tangent component of gradient of hydropotentialBase
gradMagPhiEdge	magnitude of the gradient of hydropotentialBase, on Edges

Name	Description (Continued)
waterPressureSlopeNormal	normal component of gradient of waterPressure in subglacial hydrology system
divergence	flux divergence of water in subglacial hydrology system
openingRate	rate of cavity opening in subglacial hydrology system
closingRate	rate of ice creep closure in subglacial hydrology system
zeroOrderSum	sum of zero order terms in subglacial hydrology system
deltatSGHadvec	advective CFL limited time step length in subglacial hydrology system
deltatSGHdiffu	diffusive CFL limited time step length in subglacial hydrology system
deltatSGHpressure	time step length limited by pressure equation scheme in subglacial hydrology system
deltatSGH	time step used for evolving subglacial hydrology system
channelArea	area of channel in subglacial hydrology system
channelDischarge	discharge through channel in subglacial hydrology system
channelVelocity	water velocity in channel in subglacial hydrology system
channelMelt	melt rate in channel in subglacial hydrology system
channelPressureFreeze	freezing rate in subglacial channel due to water pressure gradient (positive=freezing, negative=melting)
flowParamAChannel	flow parameter A on edges used for channel in subglacial hydrology system
channelEffectivePressure	effective pressure in the channel in subglacial hydrology system
channelClosingRate	closing rate from creep of the channel in subglacial hydrology system
channelOpeningRate	opening rate from melt of the channel in subglacial hydrology system
channelChangeRate	rate of change of channel area in subglacial hydrology system
deltatSGHadvecChannel	time step length limited by channel advection
deltatSGHdiffuChannel	time step length limited by channel diffusion
divergenceChannel	divergence due to channel flow in subglacial hydrology system
channelAreaChangeCell	change in channel area within each cell, averaged over cell area
channelDiffusivity	diffusivity in channel in subglacial hydrology system

17.9 globalStatsAM

The globalStatsAM data structure includes fields related to the global statistics analysis members.

Name	Description
totalIceVolume	total ice sheet volume
volumeAboveFloatation	total ice sheet volume above floatation
totalIceArea	total ice sheet area
floatingIceVolume	total floating ice sheet volume
floatingIceArea	total floating ice sheet area
groundedIceVolume	total grounded ice sheet volume
groundedIceArea	total grounded ice sheet area

Name	Description (Continued)
iceThicknessMean	spatially averaged ice thickness
iceThicknessMax	maximum ice thickness in domain
iceThicknessMin	minimum ice thickness in domain
totalSfcMassBal	total, area integrated surface mass balance. Positive values represent ice gain.
avgNetAccumulation	average sfcMassBal, as a thickness rate. Positive values represent ice gain.
totalBasalMassBal	total, area integrated basal mass balance. Positive values represent ice gain.
totalGroundedBasalMassBal	total, area integrated grounded basal mass balance. Positive values represent ice gain.
avgGroundedBasalMelt	average groundedBasalMassBal value, as a thickness rate. Positive values represent ice loss.
totalFloatingBasalMassBal	total, area integrated floating basal mass balance. Positive values represent ice gain.
avgSubshelfMelt	average floatingBasalMassBal value, as a thickness rate. Positive values represent ice loss.
totalCalvingFlux	total, area integrated mass loss due to calving. Positive values represent ice loss.
groundingLineFlux	total mass flux across all grounding lines. Note that flux from floating to grounded ice makes a negative contribution to this metric.
surfaceSpeedMax	maximum surface speed in the domain
basalSpeedMax	maximum basal speed in the domain

17.10 regionalStatsAM

The regionalStatsAM data structure includes fields related to the regional statistics analysis members.

Name	Description
regionalIceArea	total ice sheet area within region
regionalIceVolume	total ice sheet volume within region
regionalVolumeAboveFloatation	total ice sheet volume above floatation
regionalGroundedIceArea	total grounded ice sheet area within region
regionalGroundedIceVolume	total grounded ice sheet volume within region
regionalFloatingIceArea	total floating ice sheet area within region
regionalFloatingIceVolume	total floating ice sheet volume within region
regionalIceThicknessMin	min ice thickness within region
regionalIceThicknessMax	max ice thickness within region
regionalIceThicknessMean	mean ice thickness within region
regionalSumSfcMassBal	area-integrated surface mass balance within region
regionalAvgNetAccumulation	average sfcMassBal, as a thickness rate. Positive values represent ice gain.
regionalSumBasalMassBal	area-integrated basal mass balance within region
regionalSumGroundedBasalMassBal	total, area integrated grounded basal mass balance. Positive values represent ice gain.

Name	Description (Continued)
regionalAvgGroundedBasalMelt	average groundedBasalMassBal value, as a thickness rate. Positive values represent ice loss.
regionalSumFloatingBasalMass-Bal	total, area integrated floating basal mass balance. Positive values represent ice gain.
regionalAvgSubshelfMelt	average floatingBasalMassBal value, as a thickness rate. Positive values represent ice loss.
regionalSumCalvingFlux	area-integrated calving flux within region
regionalSumGroundingLineFlux	total mass flux across all grounding lines (note that flux from floating to grounded ice makes a negative contribution to this metric)
regionalSurfaceSpeedMax	maximum surface speed in the domain
regionalBasalSpeedMax	maximum basal speed in the domain

17.11 Run-time input/output streams

Chapter 5 provides a detailed overview of the implementation of run-time input/output streams in MPAS. Within the Land Ice core, the following streams are defined at build time:

17.11.1 A note about time strings

MPAS commonly uses time strings of the format 'YYYY-MM-DD_HH:MM:SS'. Note that items in the format string may be dropped from the left if not needed. E.g., 01-00_00:00:00 can be used instead of 0000-01-00_00:00:00 to indicate one month. Similarly, components on either side of the underscore may be replaced with a single integer representing the adjacent unit. e.g. 0000-00-01_0 and 0000-00-01_00:00:00 are identical for representing an interval of one day.

17.11.2 input

This is an immutable stream defining the fields required for input. It is only read at the initial time. Input files may have other fields in them, but only the fields specified in this stream definition are actually read. Default name is `landice_grid.nc`.

The input stream consists of the following members. All fields are optional, and some are only utilized by the model if certain model components are enabled (e.g., subglacial hydrology model).

- stream name="basicmesh"
- var name="thickness"
- var name="bedTopography"
- var name="temperature"
- var name="normalVelocity"
- var name="sfcMassBal"
- var name="floatingBasalMassBal"
- var name="surfaceAirTemperature"
- var name="basalHeatFlux"

- var name="eigencalvingParameter"
- var name="beta"
- var name="dirichletVelocityMask"
- var name="uReconstructX"
- var name="uReconstructY"
- var name="basalMeltInput"
- var name="externalWaterInput"
- var name="waterThickness"
- var name="waterPressure"
- var name="channelArea"
- var name="waterFluxMask"

17.11.3 output

This is a mutable stream defining the fields that will be output. Because it is mutable, the list of fields for output may be modified at run time by editing the streams.landice file. Default name is `output.nc`. Default clobber mode is `replace_files`, **which will overwrite existing output files**.

The output stream consists of the following members by default:

- stream name="basicmesh"
- var name="layerCenterSigma"
- var name="layerInterfaceSigma"
- var name="xtime"
- var name="simulationStartTime"
- var name="daysSinceStart"
- var name="deltat"
- var name="allowableDtACFL"
- var name="allowableDtDCFL"
- var name="thickness"
- var name="lowerSurface"
- var name="upperSurface"
- var name="cellMask"
- var name="edgeMask"

- var name="normalVelocity"
- var name="uReconstructX"
- var name="uReconstructY"

17.11.4 restart

This is an immutable stream defining the fields required for restart. It is both an input and output stream. The model writes restart files with a single time level in them periodically. If a restart from one of these checkpoints is desired, set `config_do_restart` to `.true.` and set `config_start_time` to `file` in `namelist.landice`. The model will take the start time from the value in the text file specified by `config_restart_timestamp_name` (default name is "restart_timestamp"), and use the associated .nc file to restart the model from that checkpoint.

Default name is `restart.$Y-$M-$D_$h.$m.$s.nc` with a new file for each checkpoint. Default clobber mode is `replace_files`, **which will overwrite existing output**.

All fields required to restart the model are included in restart files automatically, so the user does not need to keep track of what fields are required for restart. On a restart, only the restart file is read, and the original input file is not used. For reference the restart fields are below. Some of these are only included conditionally based on the model configuration.

- stream name="basicmesh"
- var name="xtime"
- var name="simulationStartTime"
- var name="thickness"
- var name="temperature"
- var name="surfaceAirTemperature"
- var name="basalHeatFlux"
- var name="cellMask"
- var name="bedTopography"
- var name="sfcMassBal"
- var name="floatingBasalMassBal"
- var name="eigencalvingParameter"
- var name="normalVelocity"
- var name="uReconstructX"
- var name="uReconstructY"
- var name="beta"
- var name="dirichletVelocityMask"

- var name="basalMeltInput"
- var name="externalWaterInput"
- var name="waterThickness"
- var name="waterPressure"
- var name="channelArea"
- var name="waterFluxMask"

17.11.5 basicmesh

This is an immutable stream that specifies the list of fields that make up the MPAS mesh specification. It is provided as a convenience for including mesh fields in other streams without having to list them all explicitly.

17.11.6 Other streams

As described in Chapter 5, additional streams may be added by the user at run-time. One common example would be a "forcing" stream that gets read at each time level while the model runs. This can be accomplished by creating an input stream with `input_interval` set to a time interval. If the model does not find the current time in the forcing file, it will read the latest value before the current time instead (piecewise constant forcing).

Chapter 18

Land Ice Visualization

This chapter discusses visualization tools that are specific to the Land Ice core. For instructions on visualization tools that may be used by all cores, such as Paraview, see Chapter 6.

18.1 Python

Python visualization scripts are used in some test cases. In order to use these scripts, the following python modules are required:

- matplotlib, see http://matplotlib.org
- numpy, see http://www.numpy.org
- pylab, see www.scipy.org
- netCDF4, see http://code.google.com/p/netcdf4-python

Common ways to install these packages are through Anaconda (https://anaconda.org/), Enthought Python Distribution (https://www.enthought.com/products/epd), or package managers for your operating system.

Chapter 19

Known Issues

- The barycentric interpolation used to calculate surface elevation at vertices gives garbage values for vertices associated with obtuse triangles on the dual mesh. Therefore, the model will only work properly for meshes with no obtuse triangles. Currently there is no error message when this occurs, so users must be aware of this constraint. Future work will improve the barycentric interpolation method to work for obtuse triangles.
- Paraview plots periodic fields in a messy way with lines connecting the periodic cells across the domain.
- Paraview gives the following fatal error with some Land Ice output files: "NetCDF: Start+count exceeds dimension bound".
- Paraview will not recognize fields without a vertical dimension (e.g. thickness will not be recognized) in versions earlier than 4.1.

Part III

Bibliography

Bibliography

Adams, B., L. Bauman, W. Bohnhoff, K. Dalby, M. Ebeida, J. Eddy, M. Eldred, P. Hough, K. Hu, J. Jakeman, L. Swiler, and D. Vigil, 2013: DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 5.4 User's Manual. *Sandia Technical Report SAND2010-2183*.

Albrecht, T., M. Martin, M. Haseloff, R. Winkelmann, and a. Levermann, 2011: Parameterization for subgrid-scale motion of ice-shelf calving fronts. *The Cryosphere*, **5**, 35–44, doi:10.5194/tc-5-35-2011.
URL http://www.the-cryosphere.net/5/35/2011/

Arakawa, A. and V. R. Lamb, 1977: *Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model*. Academic Press, Inc., New York, 93 pp.

Asay-Davis, X. S., S. L. Cornford, G. Durand, B. K. Galton-Fenzi, R. M. Gladstone, G. Hilmar Gudmundsson, T. Hattermann, D. M. Holland, D. Holland, P. R. Holland, D. F. Martin, P. Mathiot, F. Pattyn, and H. Seroussi, 2016: Experimental design for three interrelated marine ice sheet and ocean model intercomparison projects: MISMIP v. 3 (MISMIP +), ISOMIP v. 2 (ISOMIP +) and MISOMIP v. 1 (MISOMIP1). *Geoscientific Model Development*, **9**, 2471–2497, doi:10.5194/gmd-9-2471-2016.

Aschwanden, A., E. Bueler, C. Khroulev, and H. Blatter, 2012: An enthalpy formulation for glaciers and ice sheets. *Journal Of Glaciology*, **58**, 441–457.

Åström, J. A., D. Vallot, M. Schäfer, E. Z. Welty, S. O'neel, T. C. Bartholomaus, Y. Liu, T. I. Riikilä, T. zwinger, J. Timonen, and J. C. Moore, 2014: Termini of calving glaciers as self-organized critical systems. *Nature Geoscience*.

Bassis, J. N. and Y. Ma, 2015: Earth and Planetary Science Letters. *Earth And Planetary Science Letters*, **409**, 203–211.

Bindschadler, R., 1983: The importance of pressurized subglacial water in separation and sliding at the glacier bed. *Journal of Glaciology*, **29**, 3–19.

Bindschadler, R. A., S. Nowicki, A. Abe-Ouchi, A. Aschwanden, H. Choi, J. Fastook, G. Granzow, R. Greve, G. Gutowski, U. Herzfeld, C. Jackson, J. Johnson, C. Khroulev, A. Levermann, W. H. Lipscomb, M. A. Martin, M. Morlighem, B. R. Parizek, D. Pollard, S. F. Price, D. Ren, F. Saito, T. Sato, H. Seddik, H. Seroussi, K. Takahashi, R. Walker, and W. L. Wang, 2013: Ice-sheet model sensitivities to environmental forcing and their use in projecting future sea level (the SeaRISE project). *Journal Of Glaciology*, **59**, 195–224.

Blatter, H., 1995: Velocity and Stress-Fields in Grounded Glaciers - a Simple Algorithm for Including Deviatoric Stress Gradients. *Journal Of Glaciology*, **41**, 333–344.

Bondzio, J. H., H. Seroussi, M. Morlighem, T. Kleiner, M. Rückamp, A. Humbert, and E. Y. Larour, 2016: Modelling calving front dynamics using a level-set method : application to Jakobshavn Isbræ , West Greenland. *The Cryosphere*, **10**, 497–510, doi:10.5194/tc-10-497-2016.

Borstad, C., A. Khazendar, B. Scheuchl, M. Morlighem, E. Larour, and E. Rignot, 2016: A constitutive framework for predicting weakening and reduced buttressing of ice shelves based on observations of the progressive deterioration of the remnant Larsen B Ice Shelf. *Geophysical Research Letters*, **43**, 2027–2035, doi:10.1002/2015GL067365.
URL http://doi.wiley.com/10.1002/2015GL067365

Brinkerhoff, D. J. and J. V. Johnson, 2013: Data assimilation and prognostic whole ice sheet modelling with the variationally derived, higher order, open source, and fully parallel ice sheet model VarGlaS. *The Cryosphere*, **7**, 1161–1184, doi:10.5194/tc-7-1161-2013.
URL http://www.the-cryosphere.net/7/1161/2013/

Bueler, E. and J. Brown, 2009: Shallow shelf approximation as a "sliding law" in a thermomechanically coupled ice sheet model. *Journal of Geophysical Research*, **114**, 1–21.

Bueler, E., J. Brown, and C. Lingle, 2007: Exact solutions to the thermomechanically coupled shallow-ice approximation: effective tools for verification. *Journal of Glaciology*, **53**, 499–516, doi:10.3189/002214307783258396.
URL http://openurl.ingenta.com/content/xref?genre=article{&}issn=0022-1430{&}volume=53{&}issue=182{&}spage=499

Bueler, E., C. S. Lingle, J. a. Kallen-Brown, D. N. Covey, and L. N. Bowman, 2005: Exact solutions and verification of numerical models for isothermal ice sheets. *Journal of Glaciology*, **51**, 291–306, doi:10.3189/172756505781829449.
URL http://openurl.ingenta.com/content/xref?genre=article{&}issn=0022-1430{&}volume=51{&}issue=173{&}spage=291

Bueler, E. and W. van Pelt, 2015: Mass-conserving subglacial hydrology in the Parallel Ice Sheet Model version 0.6. *Geoscientific Model Development*, **8**, 1613–1635.

Butler, H., M. Daly, A. Doyle, S. Gillies, S. Hagen, and T. Schaub, 2016: The GeoJSON Format. Technical report, Hobu Inc.
URL https://www.rfc-editor.org/rfc/rfc7946.txt

Clarke, G. K., 2005: Subglacial Processes. *Annual Review of Earth and Planetary Sciences*, **33**, 247–276, doi:10.1146/annurev.earth.33.092203.122621.
URL http://www.annualreviews.org/doi/abs/10.1146/annurev.earth.33.092203.122621

Cornford, S. L., D. F. Martin, D. T. Graves, D. F. Ranken, A. M. Le Brocq, R. M. Gladstone, A. J. Payne, E. G. Ng, and W. H. Lipscomb, 2013: Adaptive mesh, finite volume modeling of marine ice sheets. *Journal of Computational Physics*, **232**, 529–549.

Cuffey, K. and Paterson, 2010: *The Physics of Glaciers*. Butterworth-Heinneman, Amsterdam, 4th edition, 704 pp.

Demeshko, I., J. Watkins, I. K. Tezaur, O. Guba, W. F. Spotz, A. G. Salinger, R. P. Pawlowski, and M. A. Heroux, 2018: Toward performance portability of the albany finite element analysis code using the kokkos library. *The International Journal of High Performance Computing Applications*, **0**, 1094342017749957, doi:10.1177/1094342017749957.

Dennis, J. M., J. Edwards, R. Loy, R. Jacob, A. A. Mirin, A. P. Craig, and M. Vertenstein, 2012: An application-level parallel I/O library for Earth system models. *The International Journal of High Performance Computing Applications*, **26**, 43–53, doi:10.1177/1094342011428143.

Dukowicz, J. K., S. F. Price, and W. H. Lipscomb, 2010: Consistent approximations and boundary conditions for ice-sheet dynamics from a principle of least action. *Journal of Glaciology*, **56**, 480–496, doi:10.3189/002214310792447851.
URL http://openurl.ingenta.com/content/xref?genre=article{&}issn=0022-1430{&}volume=56{&}issue=197{&}spage=480

Edwards, H. C., C. R. Trott, and D. Sunderland, 2014a: Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, **74**, 3202 – 3216, doi:https://doi.org/10.1016/j.jpdc.2014.07.003, domain-Specific Languages and High-Level Frameworks for High-Performance Computing.
URL http://www.sciencedirect.com/science/article/pii/S0743731514001257

Edwards, T. L., X. fettweis, O. Gagliardini, F. Gillet-Chaulet, H. Goelzer, J. M. Gregory, M. Hoffman, P. Huybrechts, A. J. Payne, M. Perego, S. Price, A. Quiquet, and C. Ritz, 2014b: Effect of uncertainty in surface mass balance–elevation feedback on projections of the future sea level contribution of the Greenland ice sheet. *The Cryosphere*, **8**, 195–208.

Egholm, D. L. and S. B. Nielsen, 2010: An adaptive finite volume solver for ice sheets and glaciers. *Journal of Geophysical Research*, **115**, F01006, doi:10.1029/2009JF001394.
URL http://doi.wiley.com/10.1029/2009JF001394

Engwirda, D., 2017a: JIGSAW-GEO (1.0): Locally orthogonal staggered unstructured grid generation for general circulation modelling on the sphere. *Geoscientific Model Development*, **10**, 2117–2140, doi:10.5194/gmd-10-2117-2017.

— 2017b: JIGSAW(GEO): Unstructured grid generation for geophysical modelling.
URL https://github.com/dengwirda/jigsaw-geo-matlab

Feldmann, J. and A. Levermann, 2015: Collapse of the West Antarctic Ice Sheet after local destabilization of the Amundsen Basin. *Proceedings of the National Academy of Sciences*, **112**, 14191–14196, doi:10.1073/pnas.1512482112.

Flowers, G. E., 2015: Modelling water flow under glaciers and ice sheets. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **471**, 20140907.

Flowers, G. E. and G. K. Clarke, 2002: A multicomponent coupled model of glacier hydrology 1. Theory and synthetic examples. *Journal of Geophysical Research*, **107**, 2287, doi:10.1029/2001JB001122.
URL http://doi.wiley.com/10.1029/2001JB001122

Fowler, A. C. and D. A. Larson, 1978: On the flow of polythermal glaciers I. Model and preliminary analysis. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **363**, 217–242, doi:10.1098/rspa.1983.0054.

Fyke, J. G., A. J. Weaver, D. Pollard, M. Eby, L. Carter, and A. Mackintosh, 2011: A new coupled ice sheet/climate model: description and sensitivity to model physics under Eemian, Last Glacial Maximum, late Holocene and modern climate conditions. *Geoscientific Model Development*, **4**, 117–136.

Gagliardini, O., D. Cohen, P. Råback, and T. Zwinger, 2007: Finite-element modeling of subglacial cavities and related friction law. *Journal of Geophysical Research*, **112**, F02027, doi:10.1029/2006JF000576.
URL http://www.agu.org/pubs/crossref/2007/2006JF000576.shtml

Gagliardini, O., T. Zwinger, F. Gillet-Chaulet, G. Durand, L. Favier, B. De Fleurian, R. Greve, M. Malinen, C. Martin, P. Råback, J. Ruokolainen, M. Sacchettini, M. Schäfer, H. Seddik, and J. Thies, 2013: Capabilities and performance of Elmer/Ice, a new-generation ice sheet model. *Geoscientific Model Development*, **6**, 1299–1318.

Gladstone, R. M., V. Lee, a. Vieli, and a. J. Payne, 2010: Grounding line migration in an adaptive mesh ice sheet model. *Journal of Geophysical Research*, **115**, F04014, doi:10.1029/2009JF001615.
URL http://doi.wiley.com/10.1029/2009JF001615

Glen, J. W., 1955: The Creep of Polycrystalline Ice. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **228**, 519–538, doi:10.1098/rspa.1955.0066.
URL http://rspa.royalsocietypublishing.org/cgi/doi/10.1098/rspa.1955.0066

Goldberg, D. N., 2011: A variationally derived, depth-integrated approximation to a higher-order glaciological flow model. *Journal Of Glaciology*, **57**, 157–170.

Halfar, P., 1981: On the dynamics of the ice sheets. *Journal of Geophysical Research*, **86**, 11065–11072, doi:10.1029/JC088iC10p06043.
URL http://doi.wiley.com/10.1029/JC088iC10p06043

— 1983: On the Dynamics of the Ice Sheets 2. *Journal of Geophysical Research*, **88**, 6043–6051.

Heroux, M., R. Bartlett, V. Howle, E. Vicki, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, A. Williams, and K. Stanley, 2005: An overview of the Trilinos project. *ACM Trans. Math. Softw.*, **31**, 397–423.

Hewitt, I. J., 2011: Modelling distributed and channelized subglacial drainage: the spacing of channels. *Journal of Glaciology*, **57**, 302–314, doi:10.3189/002214311796405951.
URL http://openurl.ingenta.com/content/xref?genre=article{&}issn=0022-1430{&}volume=57{&}issue=202{&}spage=302

— 2013: Seasonal changes in ice sheet motion due to melt water lubrication. *Earth And Planetary Science Letters*, **371-372**, 16–25.

Hindmarsh, R. C. and A. J. Payne, 1996: Time-step limits for stable solutions of the ice-sheet equation. *Annals of Glaciology*, **23**, 74–85.

Hoffman, M. and S. Price, 2014: Feedbacks between coupled subglacial hydrology and glacier dynamics. *Journal Of Geophysical Research-Earth Surface*, **119**, 414–436.

Hoffman, M. J., M. Perego, S. F. Price, W. H. Lipscomb, D. Jacobsen, I. Tezaur, A. G. Salinger, R. Tuminaro, and T. Zhang, 2018: MPAS-Albany Land Ice (MALI): A variable resolution ice sheet model for Earth system modeling using Voronoi grids. *Geoscientific Model Development Discussions*, **in review**, 1–47, doi:10.5194/gmd-2018-78.
URL https://www.geosci-model-dev-discuss.net/gmd-2018-78/

Hutter, K., 1983: *Theoretical glaciology; material science of ice and the mechanics of glaciers and ice sheets*. Reidel Publishing Co., Terra Scientific Publishing Co., Tokyo.

Huybrechts, P., T. Payne, and T. E. I. Group, 1996: The EISMINT benchmarks for testing ice-sheet models. *Annals of Glaciology*, **23**, 1–12.

IPCC, 2007: *Climate Change 2007 - The Physical Science Basis: Working Group I Contribution to the Fourth Assessment Report of the IPCC*. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

— 2013: *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 1535 pp.
URL www.climatechange2013.org

Jiménez, S., R. Duddu, and J. Bassis, 2017: An updated-Lagrangian damage mechanics formulation for modeling the creeping flow and fracture of ice sheets. *Comput. Methods Appl. Mech. Engrg.*, **313**, 406–432.

Larour, E., H. Seroussi, M. Morlighem, and E. Rignot, 2012: Continental scale, high order, high spatial resolution, ice sheet modeling using the Ice Sheet System Model (ISSM). *Journal of Geophysical Research*, **117**, F01022, doi:10.1029/2011JF002140.
URL http://www.agu.org/pubs/crossref/2012/2011JF002140.shtml

Leguy, G. R., 2015: *The Effect of a Basal-friction Parameterization on Grounding-line Dynamics in Ice-sheet Models*. Ph.D. thesis, New Mexico Institute of Mining and Technology, 169 pp.

Leng, W., L. Ju, M. Gunzburger, S. Price, and T. Ringler, 2012: A parallel high-order accurate finite element nonlinear Stokes ice sheet model and benchmark experiments. *Journal of Geophysical Research*, **117**.

Levermann, A., T. Albrecht, R. Winkelmann, M. A. Martin, M. Haseloff, and I. Joughin, 2012: Kinematic first-order calving law implies potential for abrupt ice-shelf retreat. *The Cryosphere*, **6**, 273–286, doi:10.5194/tc-6-273-2012.
URL http://www.the-cryosphere.net/6/273/2012/

Lipscomb, W. H., J. G. Fyke, M. Vizcaíno, W. J. Sacks, J. Wolfe, M. Vertenstein, A. Craig, E. Kluzek, and D. M. Lawrence, 2013: Implementation and Initial Evaluation of the Glimmer Community Ice Sheet Model in the Community Earth System Model. *Journal of Climate*, **26**, 7352–7371.

Little, C. M., M. Oppenheimer, R. B. Alley, V. Balaji, G. K. C. Clarke, T. L. Delworth, R. Hallberg, D. M. Holland, C. L. Hulbe, S. Jacobs, J. V. Johnson, H. Levy, W. H. Lipscomb, S. J. Marshall, B. R. Parizek, A. J. Payne, G. A. Schmidt, R. J. Stouffer, D. G. Vaughan, and M. Winton, 2007: Toward a new generation of ice sheet models. *Eos, Transactions American Geophysical Union*, **88**, 578–579, doi:10.1029/2007EO520002.
URL http://dx.doi.org/10.1029/2007EO520002

Morland, L. W. and I. R. Johnson, 1980: Steady motion of ice sheets. *Journal of Glaciology*, **25**, 229–246.

Nowicki, S., R. A. Bindschadler, et al., 2013a: Insights into spatial sensitivities of ice mass response to environmental change from the SeaRISE ice sheet modeling project I: Antarctica. *Journal of Geophysical Research*, **118**, 1002–1024.

— 2013b: Insights into spatial sensitivities of ice mass response to environmental change from the SeaRISE ice sheet modeling project II: Greenland. *Journal of Geophysical Research*, **118**, 1025–1044.

Pattyn, F., 2003: A new three-dimensional higher-order thermomechanical ice sheet model: Basic sensitivity, ice stream development, and ice flow across subglacial lakes. *Journal of Geophysical Research*, **108**, 1–15, doi:10.1029/2002JB002329.
URL http://www.agu.org/pubs/crossref/2003/2002JB002329.shtml

Pattyn, F., L. Perichon, A. Aschwanden, B. Breuer, B. de Smedt, O. Gagliardini, G. H. Gudmundsson, R. C. a. Hindmarsh, A. Hubbard, J. V. Johnson, T. Kleiner, Y. Konovalov, C. Martin, a. J. Payne, D. Pollard, S. Price, M. Rückamp, F. Saito, O. Souček, S. Sugiyama, and T. Zwinger, 2008: Benchmark experiments for higher-order and full-Stokes ice sheet models (ISMIPHOM). *The Cryosphere*, **2**, 95–108, doi:10.5194/tc-2-95-2008.
URL http://www.the-cryosphere.net/2/95/2008/

Pattyn, F. et al., 2013: Grounding-line migration in plan-view marine ice-sheet models: results of the ice2sea MISMIP3d intercomparison. *Journal Of Glaciology*, **59**, 410–422.

Pawlowski, R. P., E. T. Phipps, A. G. Salinger, S. J. Owen, C. M. Siefert, and M. L. Staten, 2012: Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part II: Application to partial differential equations. *Scientific Programming*, **20**, 327–345, doi:10.3233/SPR-2012-0351.

Payne, A. J., P. Huybrechts, R. Calov, J. L. Fastook, R. Greve, S. J. Marshall, I. Marsiat, C. Ritz, L. Tarasov, and M. P. A. Thomassen, 2000: Results from the EISMINT model intercomparison: The effects of thermomechanical coupling. *Journal of Glaciology*, **46**, 227–238.

Perego, M., M. Gunzburger, and J. Burkardt, 2012: Parallel finite-element implementation for higher-order ice-sheet models. *Journal of Glaciology*, **58**, 76–88, doi:10.3189/2012JoG11J063.

Petersen, M. R., X. Asay-Davis, D. Jacobsen, P. Jones, M. Maltrud, T. D. Ringler, A. K. Turner, L. Van Roekel, M. Veneziani, J. Wolfe, and P. J. Wolfram, 2018: An evaluation of the ocean and sea ice climate of E3SM using MPAS and interannual CORE-II forcing. *Journal of Advances in Modeling Earth Systems*, **in prep.**

Petersen, M. R., D. W. Jacobsen, T. D. Ringler, M. W. Hecht, and M. E. Maltrud, 2015: Evaluation of the arbitrary Lagrangian-Eulerian vertical coordinate method in the MPAS-Ocean model. *Ocean Modelling*, **86**, 93–113, doi:10.1016/j.ocemod.2014.12.004.
URL http://dx.doi.org/10.1016/j.ocemod.2014.12.004

Price, S., G. Flowers, and C. Schoof, 2011: Improving Hydrology in Land Ice Models. *Eos*, **92**, 164.

Ricciuto, D., K. Sargsyan, and P. Thornton, 2018: The Impact of Parametric Uncertainties on Biogeochemistry in the E3SM Land Model. *JAMES*, doi:10.1002/2017MS000962.

Ridley, J. K., P. Huybrechts, and J. M. Gregory, 2005: Elimination of the Greenland ice sheet in a high CO2 climate. *Journal of Climate*, **18**, 3409–3427.

Ringler, T., M. Petersen, R. L. Higdon, D. Jacobsen, P. W. Jones, and M. Maltrud, 2013: A multi-resolution approach to global ocean modeling. *Ocean Modelling*, **69**, 211–232, doi:10.1016/j.ocemod.2013.04.010.
URL http://dx.doi.org/10.1016/j.ocemod.2013.04.010

Ringler, T. D., D. Jacobsen, M. Gunzburger, L. Ju, M. Duda, and W. Skamarock, 2011: Exploring a Multiresolution Modeling Approach within the Shallow-Water Equations. *Monthly Weather Review*, **139**, 3348–3368, doi:10.1175/MWR-D-10-05049.1.

Saito, F., A. Abe-ouchi, and H. Blatter, 2006: European Ice Sheet Modelling Initiative (EISMINT) model intercomparison experiments with first-order mechanics. *Journal of Geophysical Research*, **111**, 1–9, doi:10.1029/2004JF000273.

Salinger, A. G., R. A. Bartlett, A. M. Bradley, Q. Chen, I. P. Demeshko, X. Gao, G. A. Hansen, A. Mota, R. P. Muller, E. Nielsen, J. T. Ostien, R. P. Pawlowski, M. Perego, E. T. Phipps, W. Sun, and I. K. Tezaur, 2016: Albany: Using component-based design to develop a flexible, generic multiphysics analysis core. *International Journal for Multiscale Computational Engineering*, **14**, 415–438.

Schoof, C., 2005: The effect of cavitation on glacier sliding. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, **461**, 609.

— 2010: Ice-sheet acceleration driven by melt supply variability. *Nature*, **468**, 803–806, doi:10.1038/nature09618.
URL http://www.nature.com/nature/journal/v468/n7325/full/nature09618.html

Schoof, C. and I. Hewitt, 2013: Ice-sheet dynamics. *Annual Review of Fluid Mechanics*, **45**, 217–239.

Schoof, C., I. J. Hewitt, and M. A. Werder, 2012: Flotation and free surface flow in a model for subglacial drainage. Part 1. Distributed drainage. *Journal of Fluid Mechanics*, **702**, 126–156, doi:10.1017/jfm.2012.165.
URL http://www.journals.cambridge.org/abstract{_}S0022112012001656

Schoof, C. and R. C. A. Hindmarsh, 2010: Thin-Film Flows with Wall Slip: An Asymptotic Analysis of Higher Order Glacier Flow Models. *The Quarterly Journal of Mechanics and Applied Mathematics*, **63**, 73–114.

Seroussi, H., M. Morlighem, E. Larour, E. Rignot, and a. Khazendar, 2014: Hydrostatic grounding line parameterization in ice sheet models. *The Cryosphere*, **8**, 2075–2087, doi:10.5194/tc-8-2075-2014.
URL http://www.the-cryosphere.net/8/2075/2014/

Shannon, S. R., A. J. Payne, I. D. Bartholomew, M. R. Van Den Broeke, T. L. Edwards, X. Fettweis, O. Gagliardini, F. Gillet-Chaulet, H. Goelzer, M. J. Hoffman, P. Huybrechts, D. W. F. Mair, P. W. Nienow, M. Perego, S. F. Price, C. J. P. P. Smeets, A. J. Sole, R. S. W. van de Wal, and T. Zwinger, 2013: Enhanced basal lubrication and the contribution of the Greenland ice sheet to future sea-level rise. *Proceedings Of The National Academy Of Sciences Of The United States Of America*, **110**, 14156–14161.

Skamarock, W. C., J. B. Klemp, M. G. Duda, L. D. Fowler, S.-H. Park, and T. D. Ringler, 2012: A Multiscale Nonhydrostatic Atmospheric Model Using Centroidal Voronoi Tesselations and C-Grid Staggering. *Monthly Weather Review*, **140**, 3090–3105, doi:10.1175/MWR-D-11-00215.1.
URL http://journals.ametsoc.org/doi/abs/10.1175/MWR-D-11-00215.1

Tezaur, I. K., M. Perego, A. G. Salinger, R. S. Tuminaro, and S. Price, 2015a: Albany/FELIX: a parallel, scalable and robust, finite element, first-order Stokes approximation ice sheet solver built for advanced analysis. *Geoscientific Model Development*, **8**, 1–24, doi:10.5194/gmd-8-1-2015.

Tezaur, I. K., R. S. Tuminaro, M. Perego, A. G. Salinger, and S. F. Price, 2015b: On the Scalability of the Albany/FELIX first-order Stokes Approximation ice Sheet Solver for Large-Scale Simulations of the Greenland and Antarctic ice Sheets. *Procedia Computer Science*, **51**, 2026–2035, doi:10.1016/j.procs.2015.05.467.
URL http://linkinghub.elsevier.com/retrieve/pii/S1877050915012752

Tulaczyk, S., W. Barclay, and F. Engelhardt, 2000: Basal mechanics of Ice Stream B, West Antarctica: 1. Till mechanics. *Journal of Geophysical Research*, **105**, 463–481.

Tuminaro, R., M. perego, I. Tezaur, A. Salinger, and S. F. Price, 2016: A matrix dependent/algebraic multigrid approach for extruded meshes with applications to ice sheet modeling. *SIAM Journal on Scientific Computing*, **38**, c504–c532.

Turner, A. K., W. H. Lipscomb, E. C. Hunke, D. W. Jacobsen, N. Jeffery, T. D. Ringler, and J. D. Wolfe, 2018: MPAS-Seaice: a new variable resolution sea-ice model. *Journal of Advances in Modeling Earth Systems*, submitted.

van der Veen, C. J., 2013: *Fundamentals of Glacier Dynamics*. CRC Press, Boca Raton, FL, 2nd edition, 389 pp.

Vizcaíno, M., U. Mikolajewicz, M. Gröger, E. Maier-Reimer, G. Schurgers, and A. M. E. Winguth, 2008: Long-term ice sheet–climate interactions under anthropogenic greenhouse forcing simulated with a complex Earth System Model. *Climate Dynamics*, **31**, 665–690.

Vizcaíno, M., U. Mikolajewicz, J. Jungclaus, and G. Schurgers, 2009: Climate modification by future ice sheet changes and consequences for ice sheet mass balance. *Climate Dynamics*, **34**, 301–324.

Werder, M. A., I. J. Hewitt, C. G. Schoof, and G. E. Flowers, 2013: Modeling channelized and distributed subglacial drainage in two dimensions. *Journa of Geophysical Research - Earth Surface*, **118**.

Zhu, Q., W. Riley, J. Tang, N. Collier, F. Hoffman, J. Randerson, X. Yang, and G. Bisht, 2018: Representing carbon, nitrogen, and phosphorus interaction in the E3SM Land Model v1: Model development and global benchmarking. *JAMES*, in review.

Part IV

Appendices

Appendix A

Namelist options

Embedded links point to information in chapter 15

A.1 velocity_solver

A.1.1 config_velocity_solver

Type:	character
Units:	unitless
Default Value:	sia
Possible Values:	'sia', 'L1L2', 'FO', 'Stokes', 'simple', 'none'

Table A.1: config_velocity_solver: Selection of the method for solving ice velocity. 'L1L2', 'FO', and 'Stokes' require compiling with external dycores. 'none' skips the calculation of velocity so the velocity field will be 0 or set to a field read from an input file. 'simple' gives a simple prescribed velocity field computed at initialization.

A.1.2 config_sia_tangent_slope_calculation

Type:	character
Units:	unitless
Default Value:	from_vertex_barycentric
Possible Values:	'from_vertex_barycentric', 'from_vertex_barycentric_kiteareas', 'from_normal_slope'

Table A.2: config_sia_tangent_slope_calculation: Selection of the method for calculating the tangent component of surface slope at edges needed by the SIA velocity solver. 'from_vertex_barycentric' interpolates upperSurface values from cell centers to vertices using the barycentric interpolation routine in operators (mpas_cells_to_points_using_baryweights) and then calculates the slope between vertices. It works for obtuse triangles, but will not work correctly across the edges of periodic meshes. 'from_vertex_barycentric_kiteareas' interpolates upperSurface values from cell centers to vertices using barycentric interpolation based on kiterea values and then calculates the slope between vertices. It will work across the edges of periodic meshes, but will not work correctly for obtuse triangles. 'from_normal_slope' uses the vector operator mpas_tangential_vector_1d to calculate the tangent slopes from the normal slopes on the edges of the adjacent cells. It will work for any mesh configuration, but is the least accurate method.

A.1.3 config_flowParamA_calculation

Type:	character
Units:	unitless
Default Value:	constant
Possible Values:	'constant', 'PB1982', 'CP2010'

Table A.3: config_flowParamA_calculation: Selection of the method for calculating the flow law parameter A. If 'constant' is selected, the value is set to config_default_flowParamA. The other options are calculated from the temperature field. This calculation only applies if config_velocity_solver is set to 'sia'. For the 'FO' velocity solver, this is set in the albany_input.xml file.

A.1.4 config_do_velocity_reconstruction_for_external_dycore

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.4: config_do_velocity_reconstruction_for_external_dycore: By default, external, higher-order dycores return the uReconstructX and uReconstructY fields (which are the native locations of their FEM solution). If this option is set to .true., uReconstructX and uReconstructY will be calculated by MPAS using framework's vector reconstruction routines based on the values of normalVelocity supplied by the external dycore. This provides a way to test the calculation of normalVelocity in the interface.

A.1.5 config_simple_velocity_type

Type:	character
Units:	unitless
Default Value:	uniform
Possible Values:	'uniform', 'radial'

Table A.5: config_simple_velocity_type: Selection of the type of simple velocity field computed at initialization when config_velocity_solver = 'simple'. See mode_forward/mpas_li_velocity_simple.F for details of what the options do.

A.1.6 config_use_glp

Type:	logical
Units:	unitless
Default Value:	.true.
Possible Values:	.true. or .false.

Table A.6: config_use_glp: If true, then apply Albany's grounding line parameterization

A.1.7 config_beta_use_effective_pressure

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.7: config_beta_use_effective_pressure: If true, then multiply beta by effective pressure before passing to Albany. This allows, e.g., a Weertman basal friction law with an effective pressure term. Note that basal friction still needs to be selected in Albany xml file.

A.2 advection

A.2.1 config_thickness_advection

Type:	character
Units:	unitless

Default Value:	fo
Possible Values:	'fo', 'none'

Table A.8: config_thickness_advection: Selection of the method for advecting thickness ('fo' = first-order upwinding).

A.2.2 config_tracer_advection

Type:	character
Units:	unitless
Default Value:	none
Possible Values:	'fo', 'none'

Table A.9: config_tracer_advection: Selection of the method for advecting tracers.

A.3 calving

A.3.1 config_calving

Type:	character
Units:	unitless
Default Value:	none
Possible Values:	'none', 'floating', 'topographic_threshold', 'thickness_threshold', 'eigencalving'

Table A.10: config_calving: Selection of the method for calving ice (as defined further below).

A.3.2 config_calving_topography

Type:	real
Units:	m
Default Value:	-500.0
Possible Values:	Any non-positive real value

Table A.11: config_calving_topography: Defines the topographic height below which ice calves (for topographic_threshold option).

A.3.3 config_calving_thickness

Type:	real
Units:	m of ice
Default Value:	100.0
Possible Values:	Any positive real value

Table A.12: config_calving_thickness: Defines the ice thickness below which ice calves (for thickness_threshold option).

A.3.4 config_calving_eigencalving_parameter_source

Type:	character
Units:	none
Default Value:	scalar
Possible Values:	'data' ('eigencalvingParameter' field read from input file), 'scalar' (specified by config_calving_eigencalving_parameter_scalar_value)

Table A.13: config_calving_eigencalving_parameter_source: Source of the eigencalving parameter value

A.3.5 config_calving_eigencalving_parameter_scalar_value

Type:	real
Units:	m s
Default Value:	3.14e16
Possible Values:	any positive real number

Table A.14: config_calving_eigencalving_parameter_scalar_value: Value of eigencalving parameter if taken as a scalar by option config_calving_eigencalving_parameter_source. (Default value is 1.0e9 m a converted to units used here.)

A.3.6 config_data_calving

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.15: config_data_calving: Select whether or not to configure calving in a 'data' model mode (calc. calving flux but do not update ice geometry)

A.3.7 config_calving_timescale

Type:	real
Units:	s
Default Value:	0.0
Possible Values:	Any non-negative real value

Table A.16: config_calving_timescale: Defines the timescale for calving. The fraction of eligible ice that calves is min(dt/calving_timescale, 1.0). A value of 0 means that all eligible ice calves.

A.3.8 config_restore_calving_front

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.17: config_restore_calving_front: If true, then restore the calving front to its initial position. If ice grows beyond the initial extent, it is removed. If ice shrinks to an extent behind the initial extent, those locations are filled with thin ice (defined as 1/10th the value of config_dynamic_thickness). Note that this violates conservation of mass and energy.

A.4 thermal_solver

A.4.1 config_thermal_solver

Type:	character
Units:	unitless
Default Value:	none
Possible Values:	'none', 'temperature', 'enthalpy'

Table A.18: config_thermal_solver: Selection of the method for the vertical thermal solver (possible values are described further below).

A.4.2 config_thermal_calculate_bmb

Type:	logical
Units:	unitless
Default Value:	.true.
Possible Values:	.true. or .false.

Table A.19: config_thermal_calculate_bmb: Determines if basal and internal melting calculated by the thermal solver should contribute to basal mass balance or be ignored.

A.4.3 config_temperature_init

Type:	character
Units:	unitless
Default Value:	file
Possible Values:	'sfc_air_temperature', 'linear', 'file'

Table A.20: config_temperature_init: Selection of the method for initializing the ice temperature (as described further below).

A.4.4 config_thermal_thickness

Type:	real
Units:	m of ice
Default Value:	1.0
Possible Values:	Any positive real value

Table A.21: config_thermal_thickness: Defines the minimum ice thickness for conducting thermal calculations. Ice thinner than this value is ignored by the thermal solver.

A.4.5 config_surface_air_temperature_source

Type:	character
Units:	unitless
Default Value:	file
Possible Values:	'constant', 'file', 'lapse'

Table A.22: config_surface_air_temperature_source: Selection of the method for setting the surface air temperature. 'constant' uses the value set by config_surface_air_temperature_value. 'file' reads the field from an input or forcing file or ESM coupler. 'lapse' uses the value of config_surface_air_temperature_value at elevation 0 with a lapse rate applied from config_surface_air_temperature_lapse_rate.

A.4.6 config_surface_air_temperature_value

Type:	real
Units:	Kelvin
Default Value:	273.15
Possible Values:	Any positive real value

Table A.23: config_surface_air_temperature_value: Constant value of the surface air temperature.

A.4.7 config_surface_air_temperature_lapse_rate

Type:	real
Units:	$\mathrm{K\,m^{-1}}$
Default Value:	0.01
Possible Values:	Any real value

Table A.24: config_surface_air_temperature_lapse_rate: Lapse rate to apply to surface air temperature when config_surface_air_temperature_source='lapse'. Positive values lead to colder temperatures at higher elevations.

A.4.8 config_basal_heat_flux_source

Type:	character
Units:	unitless
Default Value:	file
Possible Values:	'constant', 'file' 'constant' uses the value set by config_basal_heat_flux_value. 'file' reads the field from an input or forcing file or ESM coupler.

Table A.25: config_basal_heat_flux_source: Selection of the method for setting the basal heat flux.

A.4.9 config_basal_heat_flux_value

Type:	real
Units:	$\mathrm{W\,m^{-2}}$
Default Value:	0.0
Possible Values:	Any positive real value

Table A.26: config_basal_heat_flux_value: Constant value of the basal heat flux (positive upward).

A.4.10 config_basal_mass_bal_float

Type:	character
Units:	unitless
Default Value:	none
Possible Values:	'none', 'file', 'constant', 'mismip', 'seroussi'

Table A.27: config_basal_mass_bal_float: Selection of the method for computing the basal mass balance of floating ice. 'none' sets the basalMassBal field to 0 everywhere. 'file' uses without modification whatever value was read in through an input or forcing file or the value set by an ESM coupler. 'constant', 'mismip', 'seroussi' use hardcoded fields defined in the code.

A.4.11 config_basal_mass_bal_seroussi_amplitude

Type:	real
Units:	m
Default Value:	0.0
Possible Values:	any positive real value

Table A.28: config_basal_mass_bal_seroussi_amplitude: amplitude on the depth adjustment applied to the Seroussi subglacial melt parameterization

A.4.12 config_basal_mass_bal_seroussi_period

Type:	real
Units:	a
Default Value:	1.0

Possible Values:	any positive real value

Table A.29: config_basal_mass_bal_seroussi_period: period of the periodic depth adjustment applied to the Seroussi subglacial melt parameterization

A.4.13 config_basal_mass_bal_seroussi_phase

Type:	real
Units:	cycles
Default Value:	0.0
Possible Values:	any positive real value

Table A.30: config_basal_mass_bal_seroussi_phase: phase of the periodic depth adjustment applied to the Seroussi subglacial melt parameterization. Units are cycles, i.e., 0-1

A.4.14 config_bmlt_float_flux

Type:	real
Units:	$\mathrm{W\,m^{-2}}$
Default Value:	0.0
Possible Values:	Any positive real value

Table A.31: config_bmlt_float_flux: Value of the constant heat flux applied to the base of floating ice (positive upward).

A.4.15 config_bmlt_float_xlimit

Type:	real
Units:	m
Default Value:	0.0
Possible Values:	Any positive real value

Table A.32: config_bmlt_float_xlimit: x value defining region where bmlt_float_flux is applied; melt only where abs(x) is greater than xlimit.

A.5 physical_parameters

A.5.1 config_ice_density

Type:	real
Units:	$\mathrm{kg\,m^{-3}}$
Default Value:	910.0
Possible Values:	Any positive real value

Table A.33: config_ice_density: ice density to use (assumed constant and uniform)

A.5.2 config_ocean_density

Type:	real
Units:	$\mathrm{kg\,m^{-3}}$
Default Value:	1028.0
Possible Values:	Any positive real value

Table A.34: config_ocean_density: ocean density to use for calculating floatation (assumed constant and uniform)

A.5.3 config_sea_level

Type:	real
Units:	m above datum
Default Value:	0.0
Possible Values:	Any real value

Table A.35: config_sea_level: sea level to use for calculating floatation (assumed constant and uniform)

A.5.4 config_default_flowParamA

Type:	real
Units:	$\mathrm{s^{-1}\,Pa^{-n}}$
Default Value:	3.1709792e-24
Possible Values:	Any positive real value

Table A.36: config_default_flowParamA: Defines the default value of the flow law parameter A to be used if it is not being calculated from ice temperature. This value will be used by either the sia or FO velocity solver if they are respectively configured to use a scalar A value. Defaults to the SI representation of 1.0e-16 yr^{-1} Pa^{-3}.

A.5.5 config_enhancementFactor

Type:	real
Units:	none
Default Value:	1.0
Possible Values:	Any positive real value

Table A.37: config_enhancementFactor: multiplier on the flow parameter A

A.5.6 config_flowLawExponent

Type:	real
Units:	none
Default Value:	3.0
Possible Values:	Any real value

Table A.38: config_flowLawExponent: Defines the value of the Glen flow law exponent, n. This value will be used by either the sia or FO velocity solver. A value other than 3.0 is untested.

A.5.7 config_dynamic_thickness

Type:	real
Units:	m of ice
Default Value:	10.0
Possible Values:	Any positive real value

Table A.39: config_dynamic_thickness: Defines the ice thickness below which dynamics are not calculated (and hence ice velocity is set to 0).

A.6 time_integration

A.6.1 config_dt

Type:	character
Units:	unitless
Default Value:	0001-00-00_00:00:00
Possible Values:	Any time interval of the format 'YYYY-MM-DD_HH:MM:SS', but limited by CFL condition.

Table A.40: config_dt: Length of model time step defined as a time interval.

A.6.2 config_time_integration

Type:	character
Units:	unitless
Default Value:	forward_euler
Possible Values:	'forward_euler'

Table A.41: config_time_integration: Time integration method (currently, only forward Euler is supported).

A.6.3 config_adaptive_timestep

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.42: config_adaptive_timestep: Determines if the time step should be adjusted based on the CFL condition or should be steady in time. If true, the config_dt_* options are ignored.

A.6.4 config_min_adaptive_timestep

Type:	real
Units:	s
Default Value:	0.0
Possible Values:	Any non-negative real value.

Table A.43: config_min_adaptive_timestep: The minimum allowable time step in seconds. If the CFL condition dictates the time step should be shorter than this, then the model aborts.

A.6.5 config_max_adaptive_timestep

Type:	real
Units:	s
Default Value:	3.15e9
Possible Values:	Any non-negative real value.

Table A.44: config_max_adaptive_timestep: The maximum allowable time step in seconds. If the allowable time step determined by the adaptive CFL calculation is longer than this, then the model will specify config_max_adaptive_timestep as the time step instead. Defaults to 100 years (in seconds).

A.6.6 config_adaptive_timestep_CFL_fraction

Type:	real
Units:	none
Default Value:	0.25
Possible Values:	Any positive real value less than 1.0.

Table A.45: config_adaptive_timestep_CFL_fraction: A multiplier on the minimum allowable time step calculated from the CFL condition. (Setting to 1.0 may be unstable, so smaller values are recommended.)

A.6.7 config_adaptive_timestep_include_DCFL

Type:	logical
Units:	none
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.46: config_adaptive_timestep_include_DCFL: Option of whether to include the diffusive CFL condition in the determination of the maximum allowable timestep. The diffusive CFL condition at any location is estimated based on the local ice flux and surface slope.

A.6.8 config_adaptive_timestep_force_interval

Type:	character
Units:	unitless
Default Value:	1000-00-00_00:00:00
Possible Values:	Any time interval of the format 'YYYY-MM-DD_HH:MM:SS'. (items in the format string may be dropped from the left if not needed, and the components on either side of the underscore may be replaced with a single integer representing the rightmost unit)

Table A.47: config_adaptive_timestep_force_interval: If adaptive timestep is enabled, the model will ensure a timestep ends at multiples of this interval. This is useful for ensuring that model output is written at a specific desired interval (rather than the closest time after) or when running coupled to an earth system model that expects a certain interval.

A.7 time_management

A.7.1 config_do_restart

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.48: config_do_restart: Determines if the initial conditions should be read from a restart file, or an input file. To perform a restart, set this to true in the namelist.input file. The restart time will be read from config_start_time (which can be set to 'file' to have the restart time read automatically from the file defined by config_restart_timestamp_name). A restart will read everything from the restart file - no information is read from the 'input' stream. It will perform a run normally, except velocity will not be solved on a restart.

A.7.2 config_restart_timestamp_name

Type:	character
Units:	unitless
Default Value:	restart_timestamp
Possible Values:	Path to a file.

Table A.49: config_restart_timestamp_name: Path to the filename for restart timestamps to be read and written from.

A.7.3 config_start_time

Type:	character
Units:	unitless
Default Value:	0000-01-01_00:00:00
Possible Values:	'YYYY-MM-DD_HH:MM:SS' (items in the format string may be dropped from the left if not needed, and the components on either side of the underscore may be replaced with a single integer representing the rightmost unit)

Table A.50: config_start_time: Timestamp describing the initial time of the simulation. If it is set to 'file', the initial time is read from the filename specified by config_restart_timestamp_name (defaults to 'restart_timestamp').

A.7.4 config_stop_time

Type:	character
Units:	unitless
Default Value:	0000-01-01_00:00:00
Possible Values:	'YYYY-MM-DD_HH:MM:SS' or 'none' (items in the format string may be dropped from the left if not needed, and the components on either side of the underscore may be replaced with a single integer representing the rightmost unit)

Table A.51: config_stop_time: Timestamp describing the final time of the simulation. If it is set to 'none' the final time is determined from config_start_time and config_run_duration. If config_run_duration is also specified, it takes precedence over config_stop_time. Set config_stop_time to be equal to config_start_time (and config_run_duration to 'none') to perform a diagnostic solve only.

A.7.5 config_run_duration

Type:	character
Units:	unitless
Default Value:	none

Possible Values:	'YYYY-MM-DD_HH:MM:SS' or 'none' (items in the format string may be dropped from the left if not needed, and the components on either side of the underscore may be replaced with a single integer representing the rightmost unit)

Table A.52: config_run_duration: Timestamp describing the length of the simulation. If it is set to 'none' the duration is determined from config_start_time and config_stop_time. config_run_duration overrides inconsistent values of config_stop_time. If a time value is specified for config_run_duration, it must be greater than 0.

A.7.6 config_calendar_type

Type:	character
Units:	unitless
Default Value:	gregorian_noleap
Possible Values:	'gregorian', 'gregorian_noleap'

Table A.53: config_calendar_type: Selection of the type of calendar that should be used in the simulation.

A.8 io

A.8.1 config_stats_interval

Type:	integer
Units:	unitless
Default Value:	0
Possible Values:	Any positive integer value greater than or equal to 0.

Table A.54: config_stats_interval: Integer specifying interval (number of timesteps) for writing global/local statistics. If set to 0, then statistics are not written (except perhaps at startup, as determined by 'config_write_stats_on_startup'). Applies to statistics written to log file and not analysis member output written to netCDF files.

A.8.2 config_write_stats_on_startup

Type:	logical
Units:	unitless

Default Value:	.false.
Possible Values:	.true. or .false.

Table A.55: config_write_stats_on_startup: Logical flag determining if statistics should be written prior to the first time step. Applies to statistics written to log file and not analysis member output written to netCDF files.

A.8.3 config_stats_cell_ID

Type:	integer
Units:	unitless
Default Value:	1
Possible Values:	Any positive integer value greater than or equal to 0.

Table A.56: config_stats_cell_ID: global ID for the cell selected for local statistics/diagnostics. Applies to statistics written to log file and not analysis member output written to netCDF files.

A.8.4 config_write_output_on_startup

Type:	logical
Units:	unitless
Default Value:	.true.
Possible Values:	.true. or .false.

Table A.57: config_write_output_on_startup: Logical flag determining if an output file should be written prior to the first time step.

A.8.5 config_pio_num_iotasks

Type:	integer
Units:	unitless
Default Value:	0
Possible Values:	Any positive integer value greater than or equal to 0.

Table A.58: config_pio_num_iotasks: Integer specifying how many IO tasks should be used within the PIO library. A value of 0 causes all MPI tasks to also be IO tasks. IO tasks are required to write contiguous blocks of data to a file. Optimal performance is typically found by having 1-2 tasks per node performing I/O. To do so, config_pio_num_iotasks must be manually set in conjunction with config_pio_stride as appropriate for the processor layout used. For example, running on 240 processors on a machine with 24 processors per node, setting config_pio_num_iotasks=20 and config_pio_stride=12 would configure two I/O tasks per node.

A.8.6 config_pio_stride

Type:	integer
Units:	unitless
Default Value:	1
Possible Values:	Any positive integer value greater than 0.

Table A.59: config_pio_stride: Integer specifying the stride of each IO task. See config_pio_num_iotasks for details.

A.8.7 config_year_digits

Type:	integer
Units:	unitless
Default Value:	4
Possible Values:	Any positive integer value greater than 0.

Table A.60: config_year_digits: Integer specifying the number of digits used to represent the year in time strings.

A.8.8 config_output_external_velocity_solver_data

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.61: config_output_external_velocity_solver_data: If .true., external velocity solvers (if enabled) will write their own output data in addition to any MPAS output that is configured.

A.8.9 config_write_albany_ascii_mesh

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.62: config_write_albany_ascii_mesh: Logical flag determining if ascii mesh files will be created. These files are written in a format that can be used by the standalone Albany velocity solver for optimization. If .true., the model initializes, writes the mesh files, and then terminates.

A.9 decomposition

A.9.1 config_num_halos

Type:	integer
Units:	unitless
Default Value:	2
Possible Values:	Any positive interger value.

Table A.63: config_num_halos: Determines the number of halo cells extending from a blocks owned cells (Called the 0-Halo). The default first-order upwinding advection requires a minimum of 2. Note that a minimum of 3 is required for incremental remapping advection on a quad mesh or for FCT advection (neither of which is currently supported for land ice).

A.9.2 config_block_decomp_file_prefix

Type:	character
Units:	unitless
Default Value:	graph.info.part.
Possible Values:	Any path/prefix to a block decomposition file.

Table A.64: config_block_decomp_file_prefix: Defines the prefix for the block decomposition file. Can include a path. The number of blocks is appended to the end of the prefix at run-time.

A.9.3 config_number_of_blocks

Type:	integer
Units:	unitless
Default Value:	0
Possible Values:	Any integer >= 0.

Table A.65: config_number_of_blocks: Determines the number of blocks a simulation should be run with. If it is set to 0, the number of blocks is the same as the number of MPI tasks at run-time.

A.9.4 config_explicit_proc_decomp

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.66: config_explicit_proc_decomp: Determines if an explicit processor decomposition should be used. This is only useful if multiple blocks per processor are used.

A.9.5 config_proc_decomp_file_prefix

Type:	character
Units:	unitless
Default Value:	graph.info.part.
Possible Values:	Any path/prefix to a processor decomposition file.

Table A.67: config_proc_decomp_file_prefix: Defines the prefix for the processor decomposition file. This file is only read if config_explicit_proc_decomp is .true. The number of processors is appended to the end of the prefix at run-time.

A.10 debug

A.10.1 config_print_thickness_advection_info

Type:	logical
Units:	unitless
Default Value:	.false.

Possible Values:	.true. or .false.

Table A.68: config_print_thickness_advection_info: Prints additional information about thickness advection.

A.10.2 config_print_calving_info

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.69: config_print_calving_info: Prints additional information about calving physics (if enabled).

A.10.3 config_print_thermal_info

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.70: config_print_thermal_info: Prints additional information about thermal calculations (if enabled).

A.10.4 config_always_compute_fem_grid

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.71: config_always_compute_fem_grid: Always compute finite-element grid information for external dycores rather than only doing so when the ice extent changes.

A.10.5 config_print_velocity_cleanup_details

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.72: config_print_velocity_cleanup_details: After velocity is calculated there are a few checks for appropriate values in certain geometric configurations. Setting this option to .true. will cause detailed information about those adjustments to be printed.

A.11 subglacial_hydro

A.11.1 config_SGH

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.73: config_SGH: activate subglacial hydrology model

A.11.2 config_SGH_adaptive_timestep_fraction

Type:	real
Units:	unitless
Default Value:	1.0
Possible Values:	positive real number

Table A.74: config_SGH_adaptive_timestep_fraction: fraction of adaptive CFL timestep to use

A.11.3 config_SGH_max_adaptive_timestep

Type:	real
Units:	s
Default Value:	3.15e9
Possible Values:	Any non-negative real value.

Table A.75: config_SGH_max_adaptive_timestep: The maximum allowable time step in seconds. If the allowable time step determined by the adaptive CFL calculation is longer than this, then the model will specify config_SGH_max_adaptive_timestep as the time step instead. Defaults to 100 years (in seconds).

A.11.4 config_SGH_tangent_slope_calculation

Type:	character
Units:	unitless
Default Value:	from_normal_slope
Possible Values:	'from_vertex_barycentric', 'from_vertex_barycentric_kiteareas', 'from_normal_slope'

Table A.76: config_SGH_tangent_slope_calculation: Selection of the method for calculating the tangent component of slope at edges. 'from_vertex_barycentric' interpolates scalar values from cell centers to vertices using the barycentric interpolation routine in operators (mpas_cells_to_points_using_baryweights) and then calculates the slope between vertices. It works for obtuse triangles, but will not work correctly across the edges of periodic meshes. 'from_vertex_barycentric_kiteareas' interpolates scalar values from cell centers to vertices using barycentric interpolation based on kiterea values and then calculates the slope between vertices. It will work across the edges of periodic meshes, but will not work correctly for obtuse triangles. 'from_normal_slope' uses the vector operator mpas_tangential_vector_1d to calculate the tangent slopes from the normal slopes on the edges of the adjacent cells. It will work for any mesh configuration, but is the least accurate method.

A.11.5 config_SGH_pressure_calc

Type:	character
Units:	unitless
Default Value:	cavity
Possible Values:	'cavity', 'overburden'

Table A.77: config_SGH_pressure_calc: Selection of the method for calculating water pressure. 'cavity' closes the hydrology equations by assuming cavities are always completely full. 'overburden' assumes water pressure is always equal to ice overburden pressure.

A.11.6 config_SGH_alpha

Type:	real
Units:	unitless
Default Value:	1.25
Possible Values:	positive real number

Table A.78: config_SGH_alpha: power of alpha parameter in subglacial water flux formula

A.11.7 config_SGH_beta

Type:	real
Units:	unitless
Default Value:	1.5
Possible Values:	positive real number

Table A.79: config_SGH_beta: power of beta parameter in subglacial water flux formula

A.11.8 config_SGH_conduc_coeff

Type:	real
Units:	$\mathrm{m}^{(2*\mathrm{beta}-\mathrm{alpha})}\,\mathrm{s}^{(2*\mathrm{beta}-3)}\,\mathrm{kg}^{(1-\mathrm{beta})}$
Default Value:	0.001
Possible Values:	positive real number

Table A.80: config_SGH_conduc_coeff: conductivity coefficient for subglacial water flux

A.11.9 config_SGH_till_drainage

Type:	real
Units:	$\mathrm{m\,s}^{-1}$
Default Value:	3.1709792e-11
Possible Values:	positive real number. Default value is 0.001 m/yr in SI units.

Table A.81: config_SGH_till_drainage: background subglacial till drainage rate

A.11.10 config_SGH_advection

Type:	character
Units:	none
Default Value:	fo
Possible Values:	'fo','fct'

Table A.82: config_SGH_advection: Advection method for SGH. 'fo'=first-order upwind; 'fct'=flux-corrected transport. FCT currently not enabled.

A.11.11 config_SGH_bed_roughness

Type:	real
Units:	m^{-1}
Default Value:	0.5
Possible Values:	positive real number

Table A.83: config_SGH_bed_roughness: cavitation coefficient

A.11.12 config_SGH_bed_roughness_max

Type:	real
Units:	m
Default Value:	0.1
Possible Values:	positive real number

Table A.84: config_SGH_bed_roughness_max: bed roughness scale

A.11.13 config_SGH_creep_coefficient

Type:	real
Units:	none
Default Value:	0.04
Possible Values:	positive real number

Table A.85: config_SGH_creep_coefficient: creep closure coefficient

A.11.14 config_SGH_englacial_porosity

Type:	real

Units:	none
Default Value:	0.01
Possible Values:	positive real number

Table A.86: config_SGH_englacial_porosity: notional englacial porosity

A.11.15 config_SGH_till_max

Type:	real
Units:	m
Default Value:	2.0
Possible Values:	positive real number

Table A.87: config_SGH_till_max: maximum water thickness in subglacial till

A.11.16 config_SGH_chnl_active

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.88: config_SGH_chnl_active: activate channels in subglacial hydrology model

A.11.17 config_SGH_chnl_alpha

Type:	real
Units:	unitless
Default Value:	1.25
Possible Values:	positive real number

Table A.89: config_SGH_chnl_alpha: power of alpha parameter in subglacial water flux formula (in channels)

A.11.18 config_SGH_chnl_beta

Type:	real
Units:	unitless

Default Value:	1.5
Possible Values:	positive real number

Table A.90: config_SGH_chnl_beta: power of beta parameter in subglacial water flux formula (in channels)

A.11.19 config_SGH_chnl_conduc_coeff

Type:	real
Units:	$\mathrm{m}^{(2*\mathrm{beta}-\mathrm{alpha})}\,\mathrm{s}^{(2*\mathrm{beta}-3)}\,\mathrm{kg}^{(1-\mathrm{beta})}$
Default Value:	0.1
Possible Values:	positive real number

Table A.91: config_SGH_chnl_conduc_coeff: conductivity coefficient (in channels)

A.11.20 config_SGH_chnl_creep_coefficient

Type:	real
Units:	none
Default Value:	0.04
Possible Values:	positive real number

Table A.92: config_SGH_chnl_creep_coefficient: creep closure coefficient (in channels)

A.11.21 config_SGH_incipient_channel_width

Type:	real
Units:	m
Default Value:	2.0
Possible Values:	positive real number

Table A.93: config_SGH_incipient_channel_width: width of sheet beneath/around channel that contributes to melt within the channel

A.11.22 config_SGH_include_pressure_melt

Type:	logical
Units:	none

Default Value:	.true.
Possible Values:	.true. or .false.

Table A.94: config_SGH_include_pressure_melt: whether to include the pressure melt term in the rate of channel opening

A.11.23 config_SGH_shmip_forcing

Type:	character
Units:	none
Default Value:	none
Possible Values:	'none', 'C1'-'C4', 'D1'-'D5'

Table A.95: config_SGH_shmip_forcing: calculate time-varying forcing specified by SHMIP experiments C or D

A.11.24 config_SGH_basal_melt

Type:	character
Units:	none
Default Value:	file
Possible Values:	'file', 'thermal', 'basal_heat'

Table A.96: config_SGH_basal_melt: source for the basalMeltInput term. 'file' takes whatever field was input and performs no calculation. 'thermal' uses the groundedBasalMassBal field calculated by the thermal model. 'basal_heat' calculates a melt rate assuming the entirety of the basal heat flux (basalFrictionFlux+basalHeatFlux) goes to melting ice at the bed. This is calculated in the SGH module and is independent of any calculations in the thermal model.

A.12 AM_globalStats

A.12.1 config_AM_globalStats_enable

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.97: config_AM_globalStats_enable: If true, landice analysis member globalStats is called.

A.12.2 config_AM_globalStats_compute_interval

Type:	character
Units:	unitless
Default Value:	output_interval
Possible Values:	Any valid time stamp, 'dt', or 'output_interval'

Table A.98: config_AM_globalStats_compute_interval: Timestamp determining how often analysis member computations should be performed.

A.12.3 config_AM_globalStats_stream_name

Type:	character
Units:	unitless
Default Value:	globalStatsOutput
Possible Values:	Any existing stream name or 'none'

Table A.99: config_AM_globalStats_stream_name: Name of the stream that the globalStats analysis member should be tied to.

A.12.4 config_AM_globalStats_compute_on_startup

Type:	logical
Units:	unitless
Default Value:	.true.
Possible Values:	.true. or .false.

Table A.100: config_AM_globalStats_compute_on_startup: Logical flag determining if analysis member computations occur on start-up.

A.12.5 config_AM_globalStats_write_on_startup

Type:	logical
Units:	unitless

Default Value:	.true.
Possible Values:	.true. or .false.

Table A.101: config_AM_globalStats_write_on_startup: Logical flag determining if an analysis member write occurs on start-up.

A.13 AM_regionalStats

A.13.1 config_AM_regionalStats_enable

Type:	logical
Units:	unitless
Default Value:	.false.
Possible Values:	.true. or .false.

Table A.102: config_AM_regionalStats_enable: If true, landice analysis member regionalStats is called.

A.13.2 config_AM_regionalStats_compute_interval

Type:	character
Units:	unitless
Default Value:	output_interval
Possible Values:	Any valid time stamp, 'dt', or 'output_interval'

Table A.103: config_AM_regionalStats_compute_interval: Timestamp determining how often analysis member computations should be performed.

A.13.3 config_AM_regionalStats_stream_name

Type:	character
Units:	unitless
Default Value:	regionalStatsOutput
Possible Values:	Any existing stream name or 'none'

Table A.104: config_AM_regionalStats_stream_name: Name of the stream that the regionalStats analysis member should be tied to.

A.13.4 config_AM_regionalStats_compute_on_startup

Type:	logical
Units:	unitless
Default Value:	.true.
Possible Values:	.true. or .false.

Table A.105: config_AM_regionalStats_compute_on_startup: Logical flag determining if analysis member computations occur on start-up.

A.13.5 config_AM_regionalStats_write_on_startup

Type:	logical
Units:	unitless
Default Value:	.true.
Possible Values:	.true. or .false.

Table A.106: config_AM_regionalStats_write_on_startup: Logical flag determining if an analysis member write occurs on start-up.

Appendix B

Variable definitions

Embedded links point to information in chapter 17

B.1 mesh

B.1.1 latCell

Type:	real
Units:	radians
Dimension:	nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % latCell

Table B.1: latCell: Latitude location of cell centers in radians.

B.1.2 lonCell

Type:	real
Units:	radians
Dimension:	nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % lonCell

Table B.2: lonCell: Longitude location of cell centers in radians.

B.1.3 xCell

Type:	real
Units:	unitless
Dimension:	nCells

Persistence:	persistent
Location in code:	domain % blocklist % mesh % xCell

Table B.3: xCell: X Coordinate in cartesian space of cell centers.

B.1.4 yCell

Type:	real
Units:	unitless
Dimension:	nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % yCell

Table B.4: yCell: Y Coordinate in cartesian space of cell centers.

B.1.5 zCell

Type:	real
Units:	unitless
Dimension:	nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % zCell

Table B.5: zCell: Z Coordinate in cartesian space of cell centers.

B.1.6 indexToCellID

Type:	integer
Units:	unitless
Dimension:	nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % indexToCellID

Table B.6: indexToCellID: List of global cell IDs.

B.1.7 latEdge

Type:	real
Units:	radians
Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % latEdge

Table B.7: latEdge: Latitude location of edge midpoints in radians.

B.1.8 lonEdge

Type:	real
Units:	radians
Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % lonEdge

Table B.8: lonEdge: Longitude location of edge midpoints in radians.

B.1.9 xEdge

Type:	real
Units:	unitless
Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % xEdge

Table B.9: xEdge: X Coordinate in cartesian space of edge midpoints.

B.1.10 yEdge

Type:	real
Units:	unitless
Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % yEdge

Table B.10: yEdge: Y Coordinate in cartesian space of edge midpoints.

B.1.11 zEdge

Type:	real
Units:	unitless
Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % zEdge

Table B.11: zEdge: Z Coordinate in cartesian space of edge midpoints.

B.1.12 indexToEdgeID

Type:	integer
Units:	unitless
Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % indexToEdgeID

Table B.12: indexToEdgeID: List of global edge IDs.

B.1.13 latVertex

Type:	real
Units:	radians
Dimension:	nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % latVertex

Table B.13: latVertex: Latitude location of vertices in radians.

B.1.14 lonVertex

Type:	real
Units:	radians
Dimension:	nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % lonVertex

Table B.14: lonVertex: Longitude location of vertices in radians.

B.1.15 xVertex

Type:	real
Units:	unitless
Dimension:	nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % xVertex

Table B.15: xVertex: X Coordinate in cartesian space of vertices.

B.1.16 yVertex

Type:	real
Units:	unitless
Dimension:	nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % yVertex

Table B.16: yVertex: Y Coordinate in cartesian space of vertices.

B.1.17 zVertex

Type:	real
Units:	unitless
Dimension:	nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % zVertex

Table B.17: zVertex: Z Coordinate in cartesian space of vertices.

B.1.18 indexToVertexID

Type:	integer
Units:	unitless
Dimension:	nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % indexToVertexID

Table B.18: indexToVertexID: List of global vertex IDs.

B.1.19 nEdgesOnCell

Type:	integer
Units:	unitless
Dimension:	nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % nEdgesOnCell

Table B.19: nEdgesOnCell: Number of edges that border each cell.

B.1.20 nEdgesOnEdge

Type:	integer
Units:	unitless
Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % nEdgesOnEdge

Table B.20: nEdgesOnEdge: Number of edges that surround each of the cells that straddle each edge. These edges are used to reconstruct the tangential velocities.

B.1.21 cellsOnEdge

Type:	integer
Units:	unitless
Dimension:	TWO nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % cellsOnEdge

Table B.21: cellsOnEdge: List of cells that straddle each edge.

B.1.22 edgesOnCell

Type:	integer
Units:	unitless
Dimension:	maxEdges nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % edgesOnCell

Table B.22: edgesOnCell: List of edges that border each cell.

B.1.23 edgesOnEdge

Type:	integer
Units:	unitless
Dimension:	maxEdges2 nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % edgesOnEdge

Table B.23: edgesOnEdge: List of edges that border each of the cells that straddle each edge.

B.1.24 cellsOnCell

Type:	integer
Units:	unitless
Dimension:	maxEdges nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % cellsOnCell

Table B.24: cellsOnCell: List of cells that neighbor each cell.

B.1.25 verticesOnCell

Type:	integer
Units:	unitless
Dimension:	maxEdges nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % verticesOnCell

Table B.25: verticesOnCell: List of vertices that border each cell.

B.1.26 verticesOnEdge

Type:	integer
Units:	unitless
Dimension:	TWO nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % verticesOnEdge

Table B.26: verticesOnEdge: List of vertices that straddle each edge.

B.1.27 edgesOnVertex

Type:	integer
Units:	unitless
Dimension:	vertexDegree nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % edgesOnVertex

Table B.27: edgesOnVertex: List of edges that share a vertex as an endpoint.

B.1.28 cellsOnVertex

Type:	integer
Units:	unitless
Dimension:	vertexDegree nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % cellsOnVertex

Table B.28: cellsOnVertex: List of cells that share a vertex.

B.1.29 weightsOnEdge

Type:	real
Units:	unitless
Dimension:	maxEdges2 nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % weightsOnEdge

Table B.29: weightsOnEdge: Reconstruction weights associated with each of the edgesOnEdge.

B.1.30 dvEdge

Type:	real
Units:	m

Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % dvEdge

Table B.30: dvEdge: Length of each edge, computed as the distance between verticesOnEdge.

B.1.31 dcEdge

Type:	real
Units:	m
Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % dcEdge

Table B.31: dcEdge: Length of each edge, computed as the distance between cellsOnEdge.

B.1.32 angleEdge

Type:	real
Units:	radians
Dimension:	nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % angleEdge

Table B.32: angleEdge: Angle the edge normal makes with local eastward direction.

B.1.33 areaCell

Type:	real
Units:	m^2
Dimension:	nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % areaCell

Table B.33: areaCell: Area of each cell in the primary grid.

B.1.34 areaTriangle

Type:	real
Units:	m^2
Dimension:	nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % areaTriangle

Table B.34: areaTriangle: Area of each cell (triangle) in the dual grid.

B.1.35 kiteAreasOnVertex

Type:	real
Units:	m^2
Dimension:	vertexDegree nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % kiteAreasOnVertex

Table B.35: kiteAreasOnVertex: Area of the portions of each dual cell that are part of each cellsOnVertex.

B.1.36 meshDensity

Type:	real
Units:	unitless
Dimension:	nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % meshDensity

Table B.36: meshDensity: The value of the generating density function at each cell center.

B.1.37 localVerticalUnitVectors

Type:	real
Units:	unitless
Dimension:	R3 nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % localVerticalUnitVectors

Table B.37: localVerticalUnitVectors: Unit surface normal vectors defined at cell centers.

B.1.38 edgeNormalVectors

Type:	real
Units:	unitless
Dimension:	R3 nEdges
Persistence:	persistent
Location in code:	domain % blocklist % mesh % edgeNormalVectors

Table B.38: edgeNormalVectors: Normal vector defined at an edge.

B.1.39 cellTangentPlane

Type:	real
Units:	unitless
Dimension:	R3 TWO nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % cellTangentPlane

Table B.39: cellTangentPlane: The two vectors that define a tangent plane at a cell center.

B.1.40 coeffs_reconstruct

Type:	real
Units:	unitless
Dimension:	R3 maxEdges nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % coeffs_reconstruct

Table B.40: coeffs_reconstruct: Coefficients to reconstruct velocity vectors at cell centers.

B.1.41 layerThicknessFractions

Type:	real
Units:	none
Dimension:	nVertLevels
Persistence:	persistent
Location in code:	domain % blocklist % mesh % layerThicknessFractions

Table B.41: layerThicknessFractions: Fractional thickness of each sigma layer

B.1.42 layerCenterSigma

Type:	real
Units:	none
Dimension:	nVertLevels
Persistence:	persistent
Location in code:	domain % blocklist % mesh % layerCenterSigma

Table B.42: layerCenterSigma: Sigma (fractional) level at center of each layer

B.1.43 layerInterfaceSigma

Type:	real
Units:	none
Dimension:	nVertInterfaces
Persistence:	persistent
Location in code:	domain % blocklist % mesh % layerInterfaceSigma

Table B.43: layerInterfaceSigma: Sigma (fractional) level at interface between each layer (including top and bottom)

B.1.44 edgeSignOnCell

Type:	integer
Units:	unitless
Dimension:	maxEdges nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % edgeSignOnCell

Table B.44: edgeSignOnCell: Sign of edge contributions to a cell for each edge on cell. Used for bit-reproducible loops. Represents directionality of vector connecting cells.

B.1.45 edgeSignOnVertex

Type:	integer
Units:	unitless
Dimension:	maxEdges nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % edgeSignOnVertex

Table B.45: edgeSignOnVertex: Sign of edge contributions to a vertex for each edge on vertex. Used for bit-reproducible loops. Represents directionality of vector connecting vertices.

B.1.46 cellProcID

Type:	integer
Units:	unitless
Dimension:	nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % cellProcID

Table B.46: cellProcID: processor number for each cell

B.1.47 baryCellsOnVertex

Type:	integer
Units:	unitless
Dimension:	R3 nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % baryCellsOnVertex

Table B.47: baryCellsOnVertex: Cell center indices to use for interpolating from cell centers to vertex locations. Note these are local indices!

B.1.48 baryWeightsOnVertex

Type:	real
Units:	unitless
Dimension:	R3 nVertices
Persistence:	persistent
Location in code:	domain % blocklist % mesh % baryWeightsOnVertex

Table B.48: baryWeightsOnVertex: Weights to interpolate from cell centers to vertex locations. Each weight is used with the corresponding cell center index identified by baryCellsOnVertex.

B.1.49 wachspressWeightVertex

Type:	real
Units:	unitless
Dimension:	maxEdges nCells
Persistence:	persistent
Location in code:	domain % blocklist % mesh % wachspressWeightVertex

Table B.49: wachspressWeightVertex: Wachspress weights used to interpolate from vertices to cell centers.

B.1.50 xtime

Type:	text
Units:	unitless
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % mesh % xtime

Table B.50: xtime: model time, with format 'YYYY-MM-DD_HH:MM:SS'

B.1.51 deltat

Type:	real
Units:	s
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % mesh % deltat

Table B.51: deltat: time step length, in seconds. Value on a given time is the value used between the previous time level and the current time level.

B.1.52 allowableDtACFL

Type:	real
Units:	s
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % mesh % allowableDtACFL

Table B.52: allowableDtACFL: The maximum allowable time step based on the advective CFL condition. Value on a given time is the value appropriate for between the previous time level and the current time level.

B.1.53 allowableDtDCFL

Type:	real
Units:	s
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % mesh % allowableDtDCFL

Table B.53: allowableDtDCFL: The maximum allowable time step based on the diffusive CFL condition. Value on a given time is the value appropriate for between the previous time level and the current time level.

B.1.54 simulationStartTime

Type:	text
Units:	unitless
Dimension:	
Persistence:	persistent
Location in code:	domain % blocklist % mesh % simulationStartTime

Table B.54: simulationStartTime: start time of first simulation, with format 'YYYY-MM-DD_HH:MM:SS'

B.1.55 daysSinceStart

Type:	real
Units:	days
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % mesh % daysSinceStart

Table B.55: daysSinceStart: Time since simulationStartTime in days, for plotting

B.1.56 timestepNumber

Type:	integer
Units:	none
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % mesh % timestepNumber

Table B.56: timestepNumber: time step number. initial time is 0.

B.2 geometry

B.2.1 bedTopography

Type:	real
Units:	m above datum
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % bedTopography

Table B.57: bedTopography: Elevation of ice sheet bed. Once isostasy is added to the model, this should become a state variable.

B.2.2 thickness

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % thickness

Table B.58: thickness: ice thickness

B.2.3 layerThickness

Type:	real
Units:	m
Dimension:	nVertLevels nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % layerThickness

Table B.59: layerThickness: layer thickness

B.2.4 lowerSurface

Type:	real
Units:	m above datum
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % lowerSurface

Table B.60: lowerSurface: elevation at bottom of ice

B.2.5 upperSurface

Type:	real
Units:	m above datum
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % upperSurface

Table B.61: upperSurface: elevation at top of ice

B.2.6 layerThicknessEdge

Type:	real
Units:	m
Dimension:	nVertLevels nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % layerThicknessEdge

Table B.62: layerThicknessEdge: layer thickness on cell edges

B.2.7 dHdt

Type:	real
Units:	$\mathrm{m\,a^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % dHdt

Table B.63: dHdt: diagnostic field of rate of thickness change with time (dH/dt). This includes all processes (flux divergence, SMB, BMB, calving, etc.) because it is calculated as the new thickness minus the old thickness divided by the time step.

B.2.8 thicknessOld

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % thicknessOld

Table B.64: thicknessOld: ice thickness from previous time level (only used to calculate thicknessTendency)

B.2.9 dynamicThickening

Type:	real
Units:	$\mathrm{m\,a^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % dynamicThickening

Table B.65: dynamicThickening: diagnostic field of dynamic thickening rate (calculated as negative of flux divergence)

B.2.10 cellMask

Type:	integer
Units:	none
Dimension:	nCells Time

Persistence:	persistent
Location in code:	domain % blocklist % geometry % cellMask

Table B.66: cellMask: bitmask indicating various properties about the ice sheet on cells. cellMask only needs to be a restart field if config_allow_additional_advance = false (to keep the mask of initial ice extent)

B.2.11 edgeMask

Type:	integer
Units:	none
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % edgeMask

Table B.67: edgeMask: bitmask indicating various properties about the ice sheet on edges.

B.2.12 vertexMask

Type:	integer
Units:	none
Dimension:	nVertices Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % vertexMask

Table B.68: vertexMask: bitmask indicating various properties about the ice sheet on vertices.

B.2.13 sfcMassBal

Type:	real
Units:	$\mathrm{kg\,m^{-2}\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % sfcMassBal

Table B.69: sfcMassBal: applied surface mass balance

B.2.14 basalMassBal

Type:	real
Units:	$\mathrm{kg\,m^{-2}\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % basalMassBal

Table B.70: basalMassBal: applied basal mass balance

B.2.15 groundedBasalMassBal

Type:	real
Units:	$\mathrm{kg\,m^{-2}\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % groundedBasalMassBal

Table B.71: groundedBasalMassBal: Basal mass balance on grounded regions

B.2.16 floatingBasalMassBal

Type:	real
Units:	$\mathrm{kg\,m^{-2}\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % floatingBasalMassBal

Table B.72: floatingBasalMassBal: Basal mass balance on floating regions

B.2.17 calvingThickness

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % calvingThickness

Table B.73: calvingThickness: thickness of ice that calves on a given timestep (less than or equal to ice thickness)

B.2.18 eigencalvingParameter

Type:	real
Units:	m s
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % eigencalvingParameter

Table B.74: eigencalvingParameter: proportionality constant K2+- used in eigencalving formulation

B.2.19 calvingVelocity

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % calvingVelocity

Table B.75: calvingVelocity: rate of calving front retreat due to calving, represented as a velocity normal to the calving front (in the x-y plane). This retreat rate is converted from a flux to a rate in the code requiredCalvingVolumeRate.

B.2.20 requiredCalvingVolumeRate

Type:	real
Units:	$\mathrm{m^3\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % requiredCalvingVolumeRate

Table B.76: requiredCalvingVolumeRate: total volume of ice that needs to be removed based on eigencalving rate at this margin cell

B.2.21 uncalvedVolume

Type:	real

Units:	m^3
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % uncalvedVolume

Table B.77: uncalvedVolume: volume of ice that was left uncalved from required calving flux due to only applying flux over immediate neighbors (diagnostic field to assess if this limitation is a problem)

B.2.22 basalWaterThickness

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % basalWaterThickness

Table B.78: basalWaterThickness: thickness of basal water

B.2.23 restoreThickness

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % restoreThickness

Table B.79: restoreThickness: thickness of ice added when the config_restore_calving_front option is set to .true. (in order to maintain the calving front at its initial position)

B.2.24 normalSlopeEdge

Type:	real
Units:	$m\,m^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % normalSlopeEdge

Table B.80: normalSlopeEdge: normal surface slope on edges

B.2.25 apparentDiffusivity

Type:	real
Units:	$m^2\,s^{-1}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % apparentDiffusivity

Table B.81: apparentDiffusivity: apparent diffusivity at cell centers (estimated based on the local ice flux and surface slope)

B.2.26 upperSurfaceVertex

Type:	real
Units:	m above datum
Dimension:	nVertices Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % upperSurfaceVertex

Table B.82: upperSurfaceVertex: elevation at top of ice on vertices

B.2.27 tangentSlopeEdge

Type:	real
Units:	$m\,m^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % tangentSlopeEdge

Table B.83: tangentSlopeEdge: tangent surface slope on edges

B.2.28 slopeEdge

Type:	real
Units:	$m\,m^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % geometry % slopeEdge

Table B.84: slopeEdge: surface slope magnitude on edges

B.3 velocity

B.3.1 flowParamA

Type:	real
Units:	$s^{-1}\,Pa^{-n}$
Dimension:	nVertLevels nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % flowParamA

Table B.85: flowParamA: flow law parameter, A, used by shallow-ice velocity solver

B.3.2 normalVelocity

Type:	real
Units:	$m\,s^{-1}$
Dimension:	nVertInterfaces nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % normalVelocity

Table B.86: normalVelocity: horizonal velocity, normal component to an edge, layer interface

B.3.3 layerNormalVelocity

Type:	real
Units:	$m\,s^{-1}$
Dimension:	nVertLevels nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % layerNormalVelocity

Table B.87: layerNormalVelocity: horizonal velocity, normal component to an edge, layer midpoint

B.3.4 normalVelocityInitial

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nVertInterfaces nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % normalVelocityInitial

Table B.88: normalVelocityInitial: horizonal velocity, normal component to an edge, computed at initialization

B.3.5 uReconstructX

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nVertInterfaces nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % uReconstructX

Table B.89: uReconstructX: x-component of velocity reconstructed on cell centers. Also, for higher-order dycores, on input: value of the x-component of velocity that should be applied where dirichletVelocityMask==1.

B.3.6 uReconstructY

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nVertInterfaces nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % uReconstructY

Table B.90: uReconstructY: y-component of velocity reconstructed on cell centers. Also, for higher-order dycores, on input: value of the y-component of velocity that should be applied where dirichletVelocityMask==1.

B.3.7 uReconstructZ

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nVertInterfaces nCells Time

Persistence:	persistent
Location in code:	domain % blocklist % velocity % uReconstructZ

Table B.91: uReconstructZ: z-component of velocity reconstructed on cell centers

B.3.8 uReconstructZonal

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nVertInterfaces nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % uReconstructZonal

Table B.92: uReconstructZonal: zonal velocity reconstructed on cell centers

B.3.9 uReconstructMeridional

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nVertInterfaces nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % uReconstructMeridional

Table B.93: uReconstructMeridional: meridional velocity reconstructed on cell centers

B.3.10 surfaceSpeed

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % surfaceSpeed

Table B.94: surfaceSpeed: ice surface speed reconstructed at cell centers

B.3.11 basalSpeed

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % basalSpeed

Table B.95: basalSpeed: ice basal speed reconstructed at cell centers

B.3.12 beta

Type:	real
Units:	$\mathrm{Pa\,yr\,m^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % beta

Table B.96: beta: input value of basal traction parameter for sliding law used with first-order momentum balance solver (NOTE non-SI units)

B.3.13 betaSolve

Type:	real
Units:	$\mathrm{Pa\,yr\,m^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % betaSolve

Table B.97: betaSolve: value of basal traction parameter for sliding law used with first-order momentum balance solver (NOTE non-SI units); differs from beta due to any necessary adjustments made for internal consistency (e.g., zeroed out where the ice is found to be floating)

B.3.14 exx

Type:	real
Units:	$\mathrm{s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % exx

Table B.98: exx: x-component of surface strain rate

B.3.15 eyy

Type:	real
Units:	s^{-1}
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % eyy

Table B.99: eyy: y-component of surface strain rate

B.3.16 exy

Type:	real
Units:	s^{-1}
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % exy

Table B.100: exy: shear component of surface strain rate

B.3.17 eTheta

Type:	real
Units:	radians
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % eTheta

Table B.101: eTheta: orientation of principal surface strain rate

B.3.18 eyx

Type:	real
Units:	s^{-1}
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % eyx

Table B.102: eyx: shear component of surface strain rate

B.3.19 eMax

Type:	real
Units:	s^{-1}
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % eMax

Table B.103: eMax: magnitude of first principal surface strain rate

B.3.20 eMin

Type:	real
Units:	s^{-1}
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % eMin

Table B.104: eMin: magnitude of second principal surface strain rate

B.3.21 anyDynamicVertexMaskChanged

Type:	integer
Units:	none
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % anyDynamicVertex-MaskChanged

Table B.105: anyDynamicVertexMaskChanged: flag needed by external velocity solvers that indicates if the region to solve on the block's domain has changed (treated as a logical)

B.3.22 dirichletVelocityMask

Type:	integer
Units:	none
Dimension:	nVertInterfaces nCells Time

Persistence:	persistent
Location in code:	domain % blocklist % velocity % dirichletVelocityMask

Table B.106: dirichletVelocityMask: mask of where Dirichlet boundary conditions should be applied to the velocity solution. 1 means apply a Dirichlet boundary condition, 0 means do not. (higher-order dycores only)

B.3.23 dirichletMaskChanged

Type:	integer
Units:	none
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % dirichletMaskChanged

Table B.107: dirichletMaskChanged: flag needed by external velocity solvers that indicates if the Dirichlet boundary condition mask has changed (treated as a logical)

B.3.24 floatingEdges

Type:	integer
Units:	unitless
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % velocity % floatingEdges

Table B.108: floatingEdges: edges which are floating have a value of 1. non floating edges have a value of 0.

B.4 observations

B.4.1 observedSurfaceVelocityX

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % observations % observedSurfaceVelocityX

Table B.109: observedSurfaceVelocityX: X-component of observed surface velocity

B.4.2 observedSurfaceVelocityY

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % observations % observedSurfaceVelocityY

Table B.110: observedSurfaceVelocityY: Y-component of observed surface velocity

B.4.3 observedSurfaceVelocityUncertainty

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % observations % observedSurfaceVelocityUncertainty

Table B.111: observedSurfaceVelocityUncertainty: uncertainty in observed surface velocity magnitude

B.4.4 observedThicknessTendency

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % observations % observedThicknessTendency

Table B.112: observedThicknessTendency: observed tendency in thickness (dH/dt)

B.4.5 observedThicknessTendencyUncertainty

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % observations % observedThicknessTendencyUncertainty

Table B.113: observedThicknessTendencyUncertainty: uncertainty in observed tendency in thickness (dH/dt)

B.4.6 sfcMassBalUncertainty

Type:	real
Units:	$\mathrm{kg\,m^{-2}\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % observations % sfcMassBalUncertainty

Table B.114: sfcMassBalUncertainty: uncertainty in observed surface mass balance

B.4.7 thicknessUncertainty

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % observations % thicknessUncertainty

Table B.115: thicknessUncertainty: uncertainty in observed thickness

B.4.8 floatingBasalMassBalUncertainty

Type:	real
Units:	$\mathrm{kg\,m^{-2}\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % observations % floatingBasalMassBalUncertainty

Table B.116: floatingBasalMassBalUncertainty: uncertainty in observed floating basal mass balance

B.5 thermal

B.5.1 temperature

Type:	real
Units:	K
Dimension:	nVertLevels nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % temperature

Table B.117: temperature: interior ice temperature

B.5.2 waterfrac

Type:	real
Units:	unitless
Dimension:	nVertLevels nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % waterfrac

Table B.118: waterfrac: interior ice water fraction

B.5.3 enthalpy

Type:	real
Units:	$\mathrm{J\,m^{-3}}$
Dimension:	nVertLevels nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % enthalpy

Table B.119: enthalpy: interior ice enthalpy

B.5.4 surfaceAirTemperature

Type:	real
Units:	K
Dimension:	nCells Time
Persistence:	persistent

Location in code:	domain % blocklist % thermal % surfaceAirTemperature

Table B.120: surfaceAirTemperature: air temperature at the ice sheet surface

B.5.5 surfaceTemperature

Type:	real
Units:	K
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % surfaceTemperature

Table B.121: surfaceTemperature: temperature at upper ice service

B.5.6 basalTemperature

Type:	real
Units:	K
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % basalTemperature

Table B.122: basalTemperature: temperature at lower ice surface

B.5.7 pmpTemperature

Type:	real
Units:	K
Dimension:	nVertLevels nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % pmpTemperature

Table B.123: pmpTemperature: pressure melt temperature

B.5.8 basalPmpTemperature

Type:	real
Units:	K

Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % basalPmpTemperature

Table B.124: basalPmpTemperature: pressure melt temperature at lower ice surface

B.5.9 surfaceConductiveFlux

Type:	real
Units:	$\mathrm{W\,m^{-2}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % surfaceConductiveFlux

Table B.125: surfaceConductiveFlux: conductive heat flux at the upper ice surface (positive downward)

B.5.10 basalConductiveFlux

Type:	real
Units:	$\mathrm{W\,m^{-2}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % basalConductiveFlux

Table B.126: basalConductiveFlux: conductive heat flux at the lower ice surface (positive downward)

B.5.11 basalHeatFlux

Type:	real
Units:	$\mathrm{W\,m^{-2}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % basalHeatFlux

Table B.127: basalHeatFlux: basal heat flux into the ice (positive upward)

B.5.12 basalFrictionFlux

Type:	real
Units:	$\mathrm{W\,m^{-2}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % basalFrictionFlux

Table B.128: basalFrictionFlux: basal frictional heat flux into the ice (positive upward)

B.5.13 heatDissipation

Type:	real
Units:	$\mathrm{deg\,s^{-1}}$
Dimension:	nVertLevels nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % thermal % heatDissipation

Table B.129: heatDissipation: interior heat dissipation rate, divided by rhoi*cp_ice

B.6 scratch

B.6.1 iceCellMask

Type:	integer
Units:	none
Dimension:	nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % iceCellMask

Table B.130: iceCellMask: mask set to 1 in cells where some criterion is satisfied and 0 otherwise

B.6.2 iceCellMask2

Type:	integer
Units:	none
Dimension:	nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % iceCellMask2

Table B.131: iceCellMask2: mask set to 1 in cells where some criterion is satisfied and 0 otherwise

B.6.3 iceCellMask3

Type:	integer
Units:	none
Dimension:	nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % iceCellMask3

Table B.132: iceCellMask3: mask set to 1 in cells where some criterion is satisfied and 0 otherwise

B.6.4 iceEdgeMask

Type:	integer
Units:	none
Dimension:	nEdges
Persistence:	scratch
Location in code:	domain % blocklist % scratch % iceEdgeMask

Table B.133: iceEdgeMask: mask set to 1 for edges adjacent to ice-covered cells and 0 otherwise

B.6.5 workLevelCell

Type:	real
Units:	none
Dimension:	nVertLevels nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % workLevelCell

Table B.134: workLevelCell: generic work array with dimensions of (nVertLevels nCells)

B.6.6 workLevelEdge

Type:	real
Units:	none
Dimension:	nVertLevels nEdges
Persistence:	scratch
Location in code:	domain % blocklist % scratch % workLevelEdge

Table B.135: workLevelEdge: generic work array with dimensions of (nVertLevels nEdges)

B.6.7 workLevelVertex

Type:	real
Units:	none
Dimension:	nVertLevels nVertices
Persistence:	persistent
Location in code:	domain % blocklist % scratch % workLevelVertex

Table B.136: workLevelVertex: generic work array with dimensions of (nVertLevels nVertices)

B.6.8 workCell

Type:	real
Units:	none
Dimension:	nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % workCell

Table B.137: workCell: generic work array with dimensions of (nCells)

B.6.9 workCell2

Type:	real
Units:	none
Dimension:	nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % workCell2

Table B.138: workCell2: generic work array with dimensions of (nCells)

B.6.10 workCell3

Type:	real
Units:	none
Dimension:	nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % workCell3

Table B.139: workCell3: generic work array with dimensions of (nCells)

B.6.11 workTracerCell

Type:	real
Units:	none
Dimension:	maxTracersAdvect nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % workTracerCell

Table B.140: workTracerCell: generic work array with dimensions of (maxTracersAdvect nCells)

B.6.12 workTracerCell2

Type:	real
Units:	none
Dimension:	maxTracersAdvect nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % workTracerCell2

Table B.141: workTracerCell2: generic work array with dimensions of (maxTracersAdvect nCells)

B.6.13 workTracerLevelCell

Type:	real
Units:	none
Dimension:	maxTracersAdvect nVertLevels nCells
Persistence:	scratch

Location in code:	domain % blocklist % scratch % workTracerLevelCell

Table B.142: workTracerLevelCell: generic work array with dimensions of (maxTracersAdvect nVertLevels nCells)

B.6.14 workTracerLevelCell2

Type:	real
Units:	none
Dimension:	maxTracersAdvect nVertLevels nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % workTracerLevelCell2

Table B.143: workTracerLevelCell2: generic work array with dimensions of (maxTracersAdvect nVertLevels nCells)

B.6.15 slopeCellX

Type:	real
Units:	none
Dimension:	nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % slopeCellX

Table B.144: slopeCellX: x-component of slope on cell centers

B.6.16 slopeCellY

Type:	real
Units:	none
Dimension:	nCells
Persistence:	scratch
Location in code:	domain % blocklist % scratch % slopeCellY

Table B.145: slopeCellY: y-component of slope on cell centers

B.6.17 vertexIndices

Type:	integer
Units:	none
Dimension:	nVertices
Persistence:	scratch
Location in code:	domain % blocklist % scratch % vertexIndices

Table B.146: vertexIndices: local indices of each vertex

B.7 regions

B.7.1 regionCellMasks

Type:	integer
Units:	unitless
Dimension:	nRegions nCells
Persistence:	persistent
Location in code:	domain % blocklist % regions % regionCellMasks

Table B.147: regionCellMasks: masks set to 1 in cells that fall within a given region and 0 otherwise

B.8 hydro

B.8.1 waterThickness

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterThickness

Table B.148: waterThickness: water layer thickness in subglacial hydrology system

B.8.2 waterThicknessOld

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent

Location in code:	domain % blocklist % hydro % waterThicknessOld

Table B.149: waterThicknessOld: water layer thickness in subglacial hydrology system from previous time step

B.8.3 waterThicknessTendency

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterThicknessTendency

Table B.150: waterThicknessTendency: rate of change in water layer thickness in subglacial hydrology system

B.8.4 tillWaterThickness

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % tillWaterThickness

Table B.151: tillWaterThickness: water layer thickness in subglacial till

B.8.5 tillWaterThicknessOld

Type:	real
Units:	m
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % tillWaterThicknessOld

Table B.152: tillWaterThicknessOld: water layer thickness in subglacial till from previous time step

B.8.6 waterPressure

Type:	real
Units:	Pa
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterPressure

Table B.153: waterPressure: pressure in subglacial hydrology system

B.8.7 waterPressureOld

Type:	real
Units:	Pa
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterPressureOld

Table B.154: waterPressureOld: pressure in subglacial hydrology system from previous time step

B.8.8 waterPressureTendency

Type:	real
Units:	$\mathrm{Pa\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterPressureTendency

Table B.155: waterPressureTendency: tendency in pressure in subglacial hydrology system

B.8.9 basalMeltInput

Type:	real
Units:	$\mathrm{kg\,m^{-2}\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % basalMeltInput

Table B.156: basalMeltInput: basal meltwater input to subglacial hydrology system

B.8.10 externalWaterInput

Type:	real
Units:	$\mathrm{kg\,m^{-2}\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % externalWaterInput

Table B.157: externalWaterInput: external water input to subglacial hydrology system

B.8.11 frictionAngle

Type:	real
Units:	None
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % frictionAngle

Table B.158: frictionAngle: subglacial till friction angle

B.8.12 effectivePressure

Type:	real
Units:	Pa
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % effectivePressure

Table B.159: effectivePressure: effective ice pressure in subglacial hydrology system

B.8.13 hydropotential

Type:	real
Units:	Pa
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % hydropotential

Table B.160: hydropotential: hydropotential in subglacial hydrology system

B.8.14 waterFlux

Type:	real
Units:	$m^2\,s^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterFlux

Table B.161: waterFlux: total water flux in subglacial hydrology system

B.8.15 waterFluxMask

Type:	integer
Units:	none
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterFluxMask

Table B.162: waterFluxMask: mask indicating how to handle fluxes on each edge: 0=calculate based on hydropotential gradient; 1=allow outflow based on hydropotential gradient, but no inflow (NOT YET IMPLEMENTED); 2=zero flux

B.8.16 waterFluxAdvec

Type:	real
Units:	$m^2\,s^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterFluxAdvec

Table B.163: waterFluxAdvec: advective water flux in subglacial hydrology system

B.8.17 waterFluxDiffu

Type:	real
Units:	$m^2\,s^{-1}$
Dimension:	nEdges Time
Persistence:	persistent

Location in code:	domain % blocklist % hydro % waterFluxDiffu

Table B.164: waterFluxDiffu: diffusive water flux in subglacial hydrology system

B.8.18 waterVelocity

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterVelocity

Table B.165: waterVelocity: water velocity in subglacial hydrology system

B.8.19 waterVelocityCellX

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterVelocityCellX

Table B.166: waterVelocityCellX: subglacial water velocity reconstructed on cell centers, x-component

B.8.20 waterVelocityCellY

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterVelocityCellY

Table B.167: waterVelocityCellY: subglacial water velocity reconstructed on cell centers, y-component

B.8.21 effectiveConducEdge

Type:	real
Units:	$m^2\,s^{-1}\,Pa^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % effectiveConducEdge

Table B.168: effectiveConducEdge: effective Darcy hydraulic conductivity on edges in subglacial hydrology system

B.8.22 waterThicknessEdge

Type:	real
Units:	m
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterThicknessEdge

Table B.169: waterThicknessEdge: water layer thickness on edges in subglacial hydrology system

B.8.23 waterThicknessEdgeUpwind

Type:	real
Units:	m
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterThicknessEdgeUpwind

Table B.170: waterThicknessEdgeUpwind: water layer thickness of cell upwind of edge in subglacial hydrology system

B.8.24 diffusivity

Type:	real
Units:	$m^2\,s^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % diffusivity

Table B.171: diffusivity: diffusivity of water sheet in subglacial hydrology system

B.8.25 hydropotentialBase

Type:	real
Units:	Pa
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % hydropotentialBase

Table B.172: hydropotentialBase: hydropotential in subglacial hydrology system without water thickness contribution

B.8.26 hydropotentialBaseVertex

Type:	real
Units:	Pa
Dimension:	nVertices Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % hydropotentialBaseVertex

Table B.173: hydropotentialBaseVertex: hydropotential without water thickness contribution on vertices. Only used for some choices of config_SGH_tangent_slope_calculation.

B.8.27 hydropotentialBaseSlopeNormal

Type:	real
Units:	$\mathrm{Pa\,m^{-1}}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % hydropotentialBaseSlopeNormal

Table B.174: hydropotentialBaseSlopeNormal: normal component of gradient of hydropotentialBase

B.8.28 hydropotentialBaseSlopeTangent

Type:	real

Units:	$\mathrm{Pa\,m^{-1}}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % hydropotentialBaseSlopeTangent

Table B.175: hydropotentialBaseSlopeTangent: tangent component of gradient of hydropotentialBase

B.8.29 gradMagPhiEdge

Type:	real
Units:	$\mathrm{Pa\,m^{-1}}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % gradMagPhiEdge

Table B.176: gradMagPhiEdge: magnitude of the gradient of hydropotentialBase, on Edges

B.8.30 waterPressureSlopeNormal

Type:	real
Units:	$\mathrm{Pa\,m^{-1}}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % waterPressureSlopeNormal

Table B.177: waterPressureSlopeNormal: normal component of gradient of waterPressure in subglacial hydrology system

B.8.31 divergence

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % divergence

Table B.178: divergence: flux divergence of water in subglacial hydrology system

B.8.32 openingRate

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % openingRate

Table B.179: openingRate: rate of cavity opening in subglacial hydrology system

B.8.33 closingRate

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % closingRate

Table B.180: closingRate: rate of ice creep closure in subglacial hydrology system

B.8.34 zeroOrderSum

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % zeroOrderSum

Table B.181: zeroOrderSum: sum of zero order terms in subglacial hydrology system

B.8.35 deltatSGHadvec

Type:	real
Units:	s
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % deltatSGHadvec

Table B.182: deltatSGHadvec: advective CFL limited time step length in subglacial hydrology system

B.8.36 deltatSGHdiffu

Type:	real
Units:	s
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % deltatSGHdiffu

Table B.183: deltatSGHdiffu: diffusive CFL limited time step length in subglacial hydrology system

B.8.37 deltatSGHpressure

Type:	real
Units:	s
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % deltatSGHpressure

Table B.184: deltatSGHpressure: time step length limited by pressure equation scheme in subglacial hydrology system

B.8.38 deltatSGH

Type:	real
Units:	s
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % deltatSGH

Table B.185: deltatSGH: time step used for evolving subglacial hydrology system

B.8.39 channelArea

Type:	real
Units:	m^2
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelArea

Table B.186: channelArea: area of channel in subglacial hydrology system

B.8.40 channelDischarge

Type:	real
Units:	$m^3\,s^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelDischarge

Table B.187: channelDischarge: discharge through channel in subglacial hydrology system

B.8.41 channelVelocity

Type:	real
Units:	$m\,s^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelVelocity

Table B.188: channelVelocity: water velocity in channel in subglacial hydrology system

B.8.42 channelMelt

Type:	real
Units:	$kg\,m^{-1}\,s^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelMelt

Table B.189: channelMelt: melt rate in channel in subglacial hydrology system

B.8.43 channelPressureFreeze

Type:	real
Units:	$\mathrm{kg\,m^{-1}\,s^{-1}}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelPressureFreeze

Table B.190: channelPressureFreeze: freezing rate in subglacial channel due to water pressure gradient (positive=freezing, negative=melting)

B.8.44 flowParamAChannel

Type:	real
Units:	$\mathrm{Pa^{-3}\,s^{-1}}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % flowParamAChannel

Table B.191: flowParamAChannel: flow parameter A on edges used for channel in subglacial hydrology system

B.8.45 channelEffectivePressure

Type:	real
Units:	Pa
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelEffectivePressure

Table B.192: channelEffectivePressure: effective pressure in the channel in subglacial hydrology system

B.8.46 channelClosingRate

Type:	real
Units:	$\mathrm{m^2\,s^{-1}}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelClosingRate

Table B.193: channelClosingRate: closing rate from creep of the channel in subglacial hydrology system

B.8.47 channelOpeningRate

Type:	real
Units:	$m^2\,s^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelOpeningRate

Table B.194: channelOpeningRate: opening rate from melt of the channel in subglacial hydrology system

B.8.48 channelChangeRate

Type:	real
Units:	$m^2\,s^{-1}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelChangeRate

Table B.195: channelChangeRate: rate of change of channel area in subglacial hydrology system

B.8.49 deltatSGHadvecChannel

Type:	real
Units:	s
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % deltatSGHadvecChannel

Table B.196: deltatSGHadvecChannel: time step length limited by channel advection

B.8.50 deltatSGHdiffuChannel

Type:	real
Units:	s
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % deltatSGHdiffuChannel

Table B.197: deltatSGHdiffuChannel: time step length limited by channel diffusion

B.8.51 divergenceChannel

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % divergenceChannel

Table B.198: divergenceChannel: divergence due to channel flow in subglacial hydrology system

B.8.52 channelAreaChangeCell

Type:	real
Units:	$\mathrm{m\,s^{-1}}$
Dimension:	nCells Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelAreaChangeCell

Table B.199: channelAreaChangeCell: change in channel area within each cell, averaged over cell area

B.8.53 channelDiffusivity

Type:	real
Units:	$\mathrm{m^2\,s^{-1}}$
Dimension:	nEdges Time
Persistence:	persistent
Location in code:	domain % blocklist % hydro % channelDiffusivity

Table B.200: channelDiffusivity: diffusivity in channel in subglacial hydrology system

B.9 globalStatsAM

B.9.1 totalIceVolume

Type:	real
Units:	m^3
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % totalIceVolume

Table B.201: totalIceVolume: total ice sheet volume

B.9.2 volumeAboveFloatation

Type:	real
Units:	m^3
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % volumeAboveFloatation

Table B.202: volumeAboveFloatation: total ice sheet volume above floatation

B.9.3 totalIceArea

Type:	real
Units:	m^2
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % totalIceArea

Table B.203: totalIceArea: total ice sheet area

B.9.4 floatingIceVolume

Type:	real
Units:	m^3
Dimension:	Time

Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % floatingIceVolume

Table B.204: floatingIceVolume: total floating ice sheet volume

B.9.5 floatingIceArea

Type:	real
Units:	m^2
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % floatingIceArea

Table B.205: floatingIceArea: total floating ice sheet area

B.9.6 groundedIceVolume

Type:	real
Units:	m^3
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % groundedIceVolume

Table B.206: groundedIceVolume: total grounded ice sheet volume

B.9.7 groundedIceArea

Type:	real
Units:	m^2
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % groundedIceArea

Table B.207: groundedIceArea: total grounded ice sheet area

B.9.8 iceThicknessMean

Type:	real

Units:	m
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % iceThicknessMean

Table B.208: iceThicknessMean: spatially averaged ice thickness

B.9.9 iceThicknessMax

Type:	real
Units:	m
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % iceThicknessMax

Table B.209: iceThicknessMax: maximum ice thickness in domain

B.9.10 iceThicknessMin

Type:	real
Units:	m
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % iceThicknessMin

Table B.210: iceThicknessMin: minimum ice thickness in domain

B.9.11 totalSfcMassBal

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % totalSfcMassBal

Table B.211: totalSfcMassBal: total, area integrated surface mass balance. Positive values represent ice gain.

B.9.12 avgNetAccumulation

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % avgNetAccumulation

Table B.212: avgNetAccumulation: average sfcMassBal, as a thickness rate. Positive values represent ice gain.

B.9.13 totalBasalMassBal

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % totalBasalMassBal

Table B.213: totalBasalMassBal: total, area integrated basal mass balance. Positive values represent ice gain.

B.9.14 totalGroundedBasalMassBal

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % totalGroundedBasalMassBal

Table B.214: totalGroundedBasalMassBal: total, area integrated grounded basal mass balance. Positive values represent ice gain.

B.9.15 avgGroundedBasalMelt

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent

Location in code:	domain % blocklist % globalStatsAM % avgGroundedBasalMelt

Table B.215: avgGroundedBasalMelt: average groundedBasalMassBal value, as a thickness rate. Positive values represent ice loss.

B.9.16 totalFloatingBasalMassBal

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % totalFloatingBasalMassBal

Table B.216: totalFloatingBasalMassBal: total, area integrated floating basal mass balance. Positive values represent ice gain.

B.9.17 avgSubshelfMelt

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % avgSubshelfMelt

Table B.217: avgSubshelfMelt: average floatingBasalMassBal value, as a thickness rate. Positive values represent ice loss.

B.9.18 totalCalvingFlux

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % totalCalvingFlux

Table B.218: totalCalvingFlux: total, area integrated mass loss due to calving. Positive values represent ice loss.

B.9.19 groundingLineFlux

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % groundingLineFlux

Table B.219: groundingLineFlux: total mass flux across all grounding lines. Note that flux from floating to grounded ice makes a negative contribution to this metric.

B.9.20 surfaceSpeedMax

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % surfaceSpeedMax

Table B.220: surfaceSpeedMax: maximum surface speed in the domain

B.9.21 basalSpeedMax

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	Time
Persistence:	persistent
Location in code:	domain % blocklist % globalStatsAM % basalSpeedMax

Table B.221: basalSpeedMax: maximum basal speed in the domain

B.10 regionalStatsAM

B.10.1 regionalIceArea

Type:	real
Units:	$\mathrm{m^2}$

Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalIceArea

Table B.222: regionalIceArea: total ice sheet area within region

B.10.2 regionalIceVolume

Type:	real
Units:	m^3
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalIceVolume

Table B.223: regionalIceVolume: total ice sheet volume within region

B.10.3 regionalVolumeAboveFloatation

Type:	real
Units:	m^3
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalVolume-AboveFloatation

Table B.224: regionalVolumeAboveFloatation: total ice sheet volume above floatation

B.10.4 regionalGroundedIceArea

Type:	real
Units:	m^2
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regional-GroundedIceArea

Table B.225: regionalGroundedIceArea: total grounded ice sheet area within region

B.10.5 regionalGroundedIceVolume

Type:	real
Units:	m^3
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalGroundedIceVolume

Table B.226: regionalGroundedIceVolume: total grounded ice sheet volume within region

B.10.6 regionalFloatingIceArea

Type:	real
Units:	m^2
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalFloatingIceArea

Table B.227: regionalFloatingIceArea: total floating ice sheet area within region

B.10.7 regionalFloatingIceVolume

Type:	real
Units:	m^3
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalFloatingIceVolume

Table B.228: regionalFloatingIceVolume: total floating ice sheet volume within region

B.10.8 regionalIceThicknessMin

Type:	real
Units:	m
Dimension:	nRegions Time
Persistence:	persistent

Location in code:	domain % blocklist % regionalStatsAM % regionalIceThicknessMin

Table B.229: regionalIceThicknessMin: min ice thickness within region

B.10.9 regionalIceThicknessMax

Type:	real
Units:	m
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalIceThicknessMax

Table B.230: regionalIceThicknessMax: max ice thickness within region

B.10.10 regionalIceThicknessMean

Type:	real
Units:	m
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalIceThicknessMean

Table B.231: regionalIceThicknessMean: mean ice thickness within region

B.10.11 regionalSumSfcMassBal

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalSumSfcMassBal

Table B.232: regionalSumSfcMassBal: area-integrated surface mass balance within region

B.10.12 regionalAvgNetAccumulation

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalAvgNetAccumulation

Table B.233: regionalAvgNetAccumulation: average sfcMassBal, as a thickness rate. Positive values represent ice gain.

B.10.13 regionalSumBasalMassBal

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalSumBasalMassBal

Table B.234: regionalSumBasalMassBal: area-integrated basal mass balance within region

B.10.14 regionalSumGroundedBasalMassBal

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalSumGroundedBasalMassBal

Table B.235: regionalSumGroundedBasalMassBal: total, area integrated grounded basal mass balance. Positive values represent ice gain.

B.10.15 regionalAvgGroundedBasalMelt

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	nRegions Time

Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalAvgGroundedBasalMelt

Table B.236: regionalAvgGroundedBasalMelt: average groundedBasalMassBal value, as a thickness rate. Positive values represent ice loss.

B.10.16 regionalSumFloatingBasalMassBal

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalSumFloatingBasalMassBal

Table B.237: regionalSumFloatingBasalMassBal: total, area integrated floating basal mass balance. Positive values represent ice gain.

B.10.17 regionalAvgSubshelfMelt

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalAvgSubshelfMelt

Table B.238: regionalAvgSubshelfMelt: average floatingBasalMassBal value, as a thickness rate. Positive values represent ice loss.

B.10.18 regionalSumCalvingFlux

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalSumCalvingFlux

Table B.239: regionalSumCalvingFlux: area-integrated calving flux within region

B.10.19 regionalSumGroundingLineFlux

Type:	real
Units:	$\mathrm{kg\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalSumGroundingLineFlux

Table B.240: regionalSumGroundingLineFlux: total mass flux across all grounding lines (note that flux from floating to grounded ice makes a negative contribution to this metric)

B.10.20 regionalSurfaceSpeedMax

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalSurfaceSpeedMax

Table B.241: regionalSurfaceSpeedMax: maximum surface speed in the domain

B.10.21 regionalBasalSpeedMax

Type:	real
Units:	$\mathrm{m\,yr^{-1}}$
Dimension:	nRegions Time
Persistence:	persistent
Location in code:	domain % blocklist % regionalStatsAM % regionalBasalSpeedMax

Table B.242: regionalBasalSpeedMax: maximum basal speed in the domain