Preprocessing
Model data for a regional ice flow models often comes from many different sources, each with its own gridding, projection, and geodetic choices. Manipulating these data into a common format suitable for a numerical model takes a great deal of time and is error prone. Although often not present in the raw data files due to lack of standardized formats (or not adhering to standards), someone has usually provided the semantic information to GIS software in order to visualize each data set. To reduce the number of intermediate steps, our software reads input in any coordinate system from a variety of GIS formats (any supported by the GDAL package) and converts on the fly to a chosen model coordinate system. Each input data set can use a different format and coordinate system, but any necessary conversions are transparent to the user. In parallel, each process reads only the necessary part of each input file.

Geometry and Mesh Generation
Constructing an unstructured hexahedral mesh with refinement in a region of interest typically involves several steps, due to the general-purpose nature of meshing packages. We automate the steps by simplifying the input to bed and surface in any format/coordinate system, a closed polygon defining the horizontal is done by either CAMALPaver (from CUBIT) or the new open source library. This triangulation, which combined with the exterior boundary, defines the volume to be meshed. The meshing in the horizontal is done by either CAMALPaver (from CUBIT) or the new open source library.

Conservative polymeric ice formulation
We solve field equations for total momentum, pressure, and total energy density \( (\rho \mathbf{u} p) \in W^{2,\infty} \times L^2 \times H^1 \) in the following form:

\[
\begin{align*}
\rho \left( \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \mathbf{f} \right) &= 0 \\
\n\n\end{align*}
\]

where \( \mathbf{f} \) is the body force, \( \mathbf{u} \) the velocity field, \( p \) the hydrostatic pressure, \( \rho \) the density of ice. The equations above are defined on the entire computational domain. The velocity \( \mathbf{u} \) is constrained as follows:

\[
\mathbf{u} \cdot \mathbf{n} = 0 \quad \text{on the boundary of the domain},
\]

\[
\mathbf{u} = \mathbf{u}_b \quad \text{on the boundary of the domain},
\]

where \( \mathbf{u}_b \) is the prescribed boundary velocity.

Parallel scalability on Blue Gene/P

Coupled velocity, surface erosion, and erosion
Here, we solve the hydrostatic equations coupled to surface erosion and evolution.

\[
\begin{align*}
\n\end{align*}
\]

where \( \gamma \) is the nonlinear effective viscosity with regularization and \( p \) the reaction-dominated \( \gamma \) is the second invariant. The momentum equations are discretized using Ch. finite elements and the surface evolution equation uses a node-centered upwind finite volume scheme with coupled volumes computed using centered reconstruction.

Why implicit?
The evolution of glaciers and ice sheets occurs on multiple time scales and frequently the slower of these scales are of great scientific importance. Most models of ice sheets and other climate systems are based on methods in which crucial components of the physics are treated explicitly. In addition to reducing the accuracy, time splitting errors produced by such methods may radically change steady state solutions.

Tightly coupled solvers, loosely coupled software
It is desirable to reuse exactly the same ‘physics’ code to run single-physics models, coupled models using semi-implicit methods, fully implicit coupled models using split preconditioners, and fully implicit coupled models using monolithic preconditioners. A generic interface has been added to PETSc [2] and the results on this poster are early client applications. It provides efficient assembly with arbitrary subphysics nesting and parallel decomposition, using a natural interface based on ‘local submatrices’. When the local submatrices interface is used for assembly, subphysics modules can be composed without recompilation into arbitrarily deep hierarchies, the hierarchy is flattened by the library so that performance is not affected. An example use case is the preconditioner for \( \mathbf{J} \).


g_{\mathbf{J},\mathbf{J}^{\omega}_{-1}} = \mathbf{J} \quad \text{is a vector containing all the discretized state variables,} \\
\mathbf{g}^{\omega}_{-1} \quad \text{is the stich part of the equation,} \\
r \quad \text{represents the non-stich part,} \\
\mathbf{J} \quad \text{the classical physics module preconditioners,} \\
\mathbf{g}^{\omega}_{-1} \quad \text{become available. For example, the same physics code can be used for steady state analysis using Newton or pseudotransient continuation and for transient analysis using Rosenbrock-W. Runge-Kutta-IMEX, or linear multistep methods, all with adaptive error control, with no modification to the solution code.}

References