

SEACISM: A scalable, efficient, and accurate Glimmer-CISM

Project Members:

John Burkardt, FSU

Kate Evans, ORNL

J.-F. Lemieux, New York U

Matt Norman, ORNL

Jeff Nichols, ORNL

Mauro Perego, FSU

Andy Salinger, Sandia

Pat Worley, ORNL

Consultation/Assistance from:

Jim Edwards, NCAR

Max Gunzburger, FSU

David Holland, NYU

Bill Lipscomb, Los Alamos

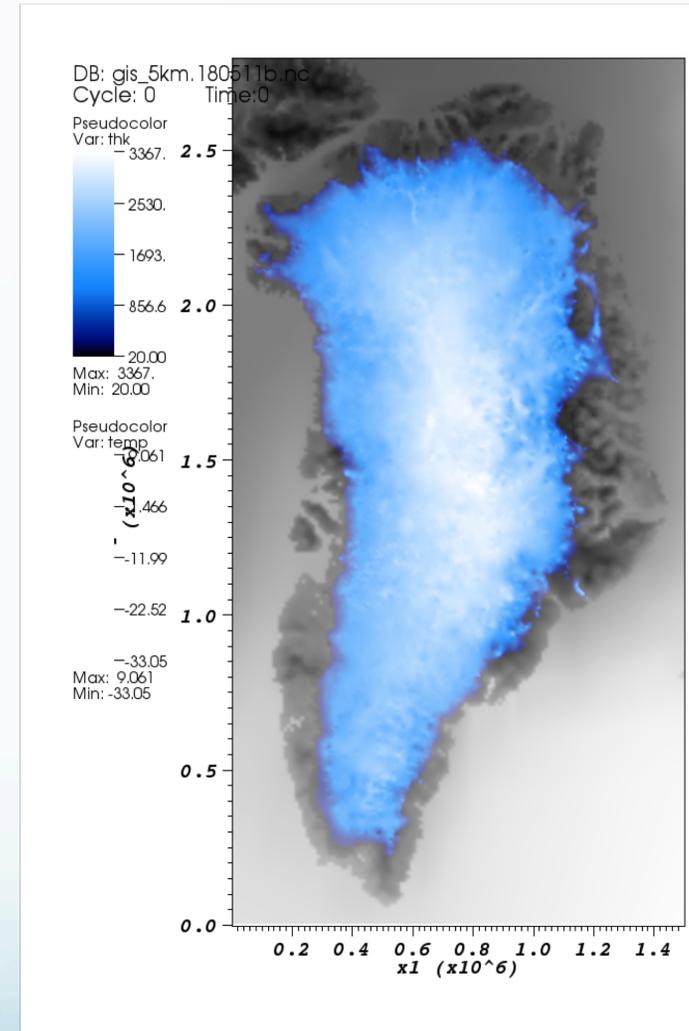
Steve Price, Los Alamos

Mariana Vertenstein, NCAR

GLIMMER Steering committee



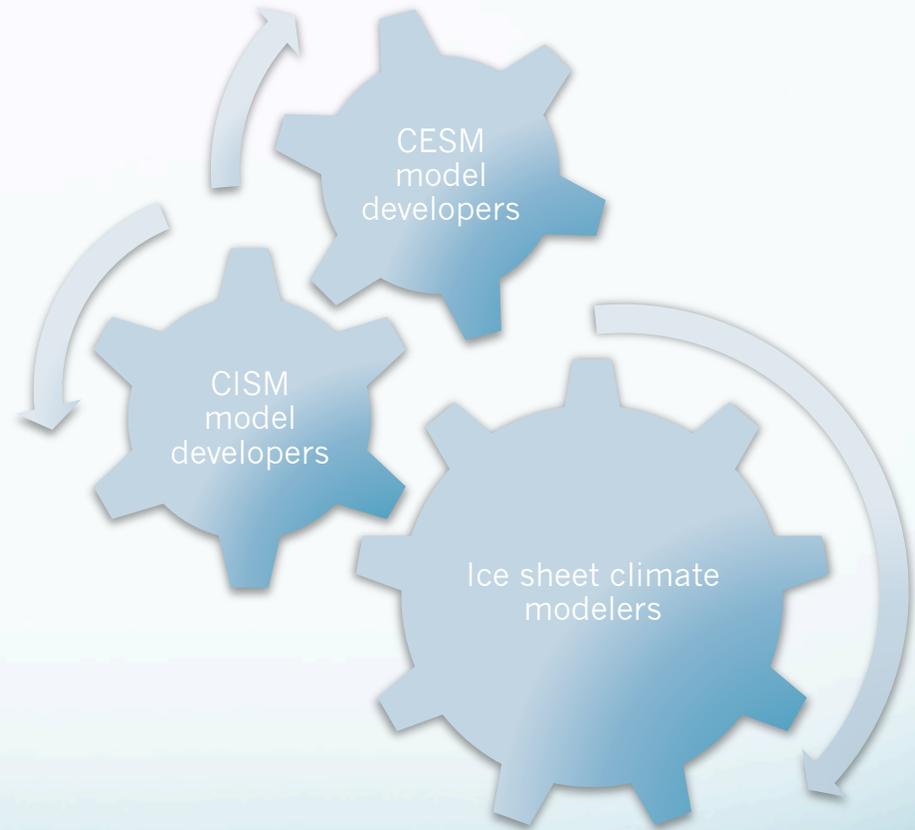
THE FLORIDA STATE UNIVERSITY



Thanks to DOE ASCR for supporting ice sheet model development!

Goal: Provide a state-of-the-art ice sheet model with continuous access by the climate community

- Implement parallel, scalable capability as soon as possible to allow efficient high-resolution simulations
- Design to allow ongoing code extensions by ice sheet modelers, e.g. new parameterizations and equations
- Maintain consistency and interaction with the production-level CESM



EVENTUAL GOAL: coupled simulations with other climate components

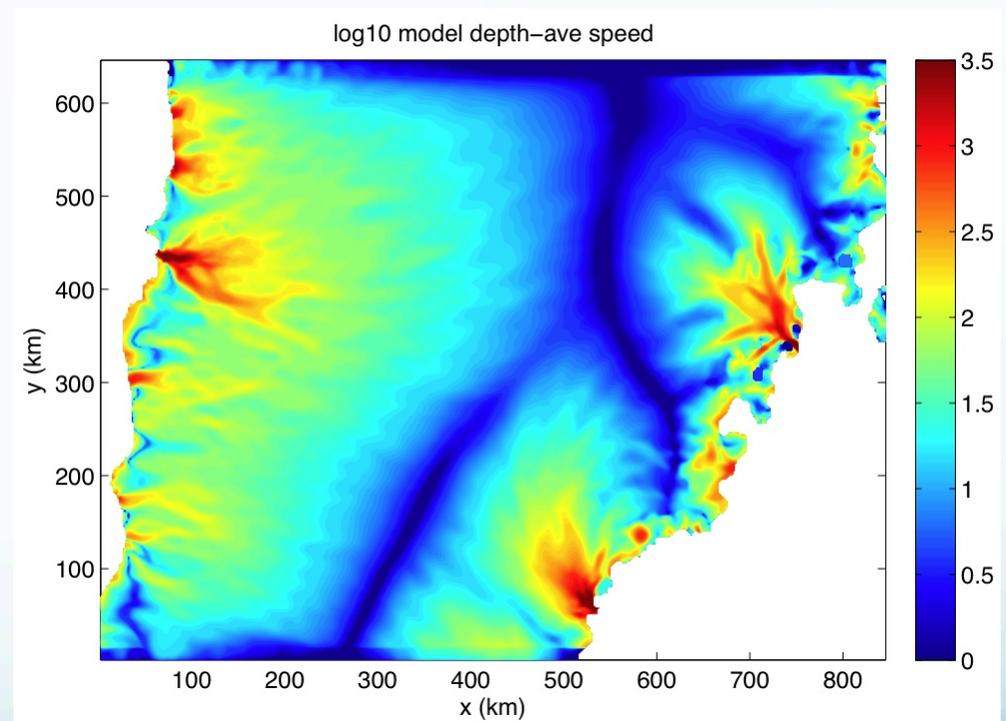
Recent ice sheet model results using parallel Trilinos solver in HO Glimmer-CISM

Greenland configuration with HO dycore:

5km simulations produced ice thinning in a radially propagating direction upstream

Experimental 2km resolution simulations show possible ice thinning occurring more along the ice channels as observed.

For this experiment: Faster model throughput allowed the finer resolution runs to occur, and higher resolutions produced simulations closer to observations



Courtesy S. Price, LANL, presented at AGU Nat'l meeting, Dec. 2010

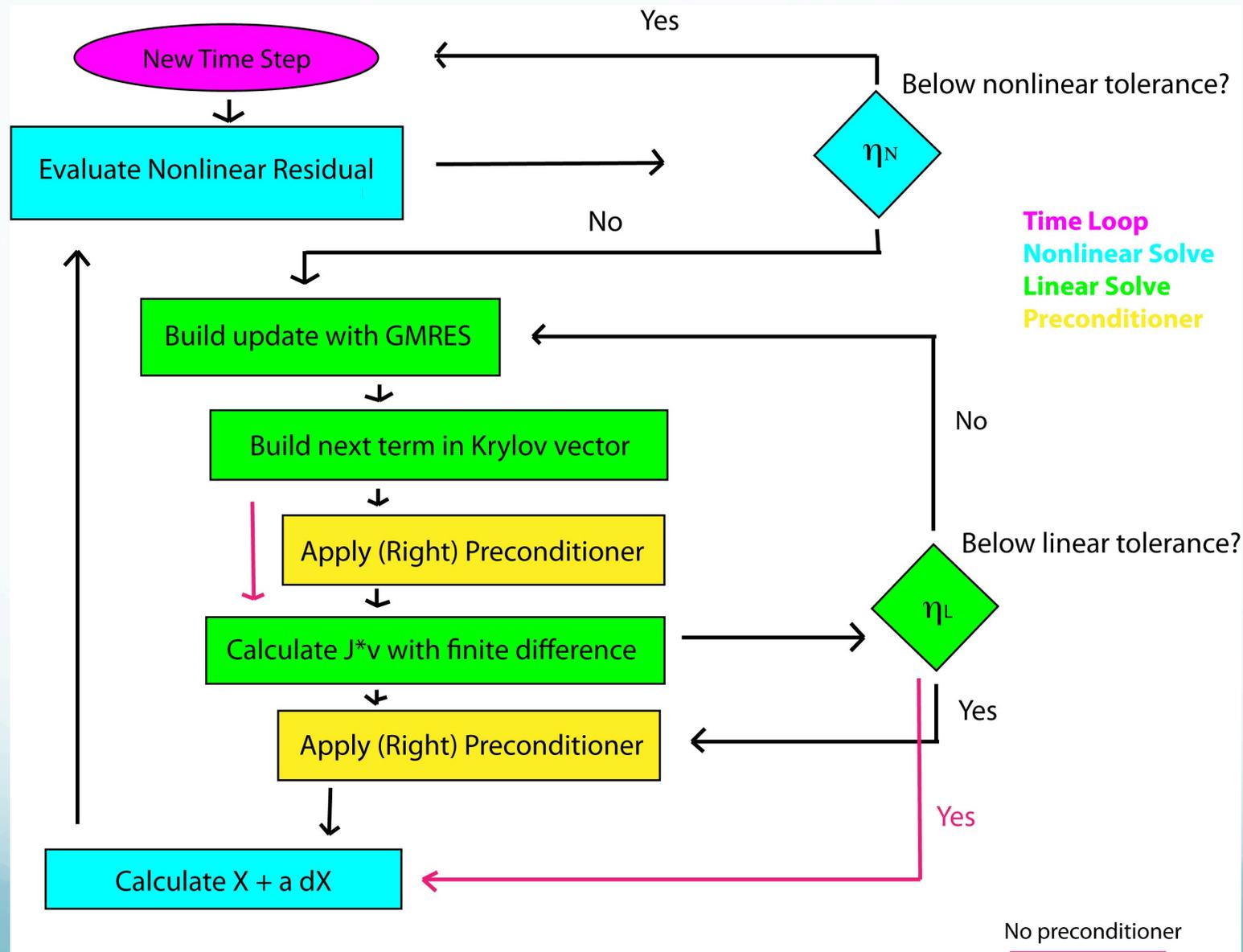
Solving the HO ice sheet momentum equations coherently to a set nonlinear tolerance

- Current G-CISM uses Picard, within which GMRES is called to solve velocity components sequentially
- Now: Uses Inexact Newton to solve $F(u) = A(u)u - b = 0$ system of nonlinear equations
- Picard: slow/cheap, Newton: fast/expensive (per iteration)

$$F'(x^{t+1})\delta x = -f(x^{t+1})$$

- We use Newton to solve, and use Picard with a loose tolerance as a preconditioner. Currently Picard preconditioner has a 1e-4 tolerance, IfPack with 0 overlap as preconditioner

Newton-Krylov solution method



Trilinos Interface in CISM for velocity solve

- Using new Piro package as a user-friendly wrapper
 - calls nonlinear solvers, time integrators, continuation etc.
 - U/Q and optimization around your simulation, e.g. Dakota
- Implemented C++ interface layer to expose Trilinos functions
- Allows transfer to new finite element version of code, LIFE-V
- Configure options added
e.g. `--with-trilinos` link to Trilinos libraries

Current Packages Being Used within Piro

NOX: nonlinear solvers

Stratimikos: allows user to specify solver options at runtime in an XML file

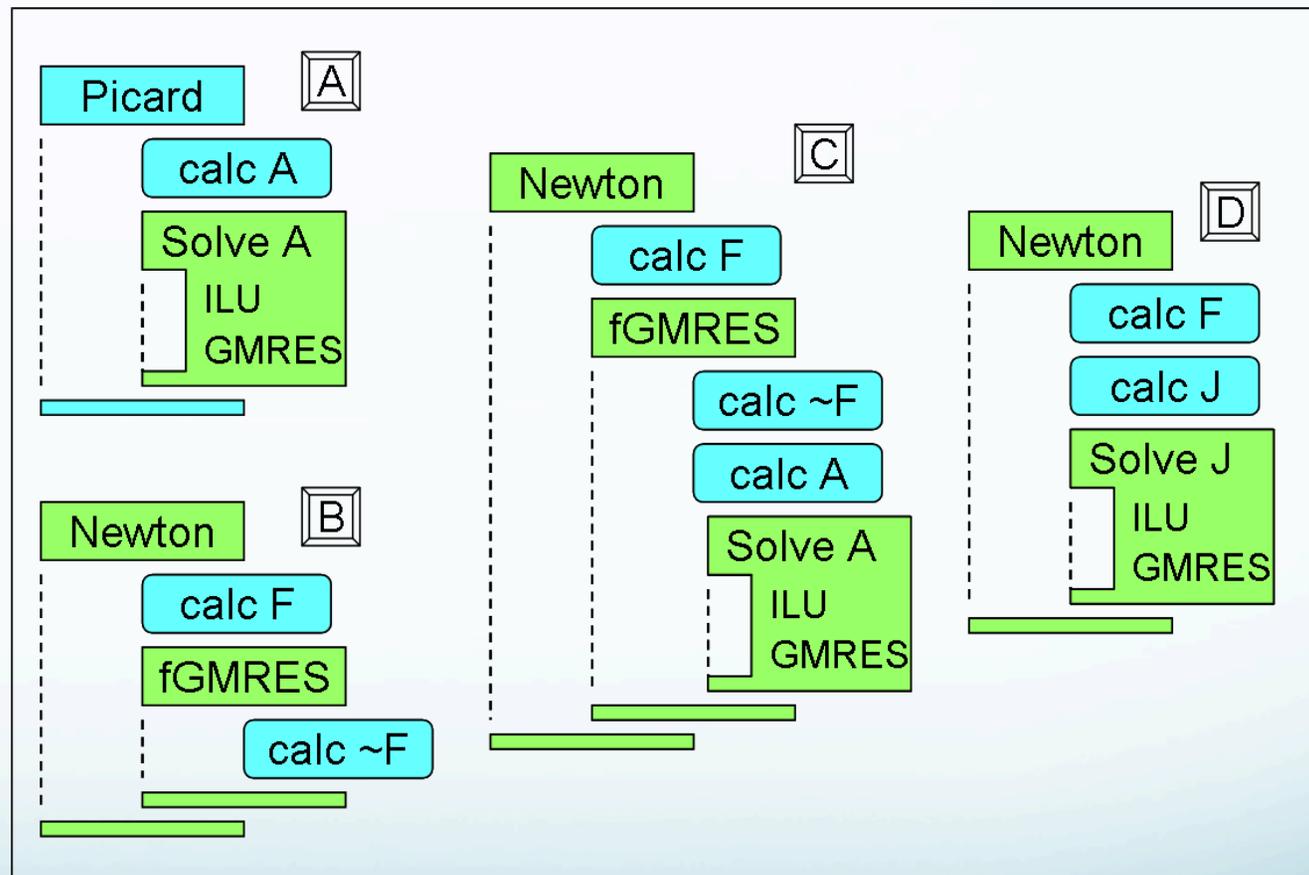
Belos: linear solvers – FGMRES, can use GPU through tpetra



Interfacing the Trilinos package with CISM, and eventually, CESM

A: Current version in the CESM repo

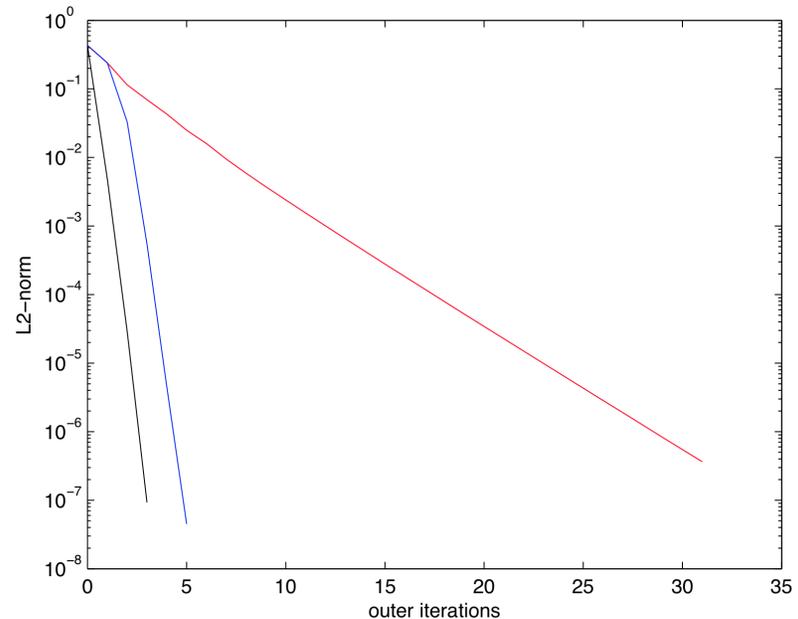
C: Working version being tested on CESM branch



4 methods for interfacing the solver to Glimmer-CISM. Version C uses function pointers to G-CISM to evaluate the nonlinear solution and call the preconditioner

Jacobian-Free Newton-Krylov as a solver in Glimmer-CISM

- **GMRES iterations are reduced with JFNK for a given level of tolerance**
- **For test cases evaluated, JFNK is ~2-3.5 times faster than Picard for a given # processors**
- **We use same tolerance for both solvers for testing, so accuracy is comparable. Results similar to original solver settings for test cases**
- **Picard as preconditioner works as designed, with more gains possible**
- **This behavior propagates to other test cases (e.g. confined shelf, ISMIP-HOM, low res GIS)**



Greenland 10km test case, one time step.

Red line: Picard

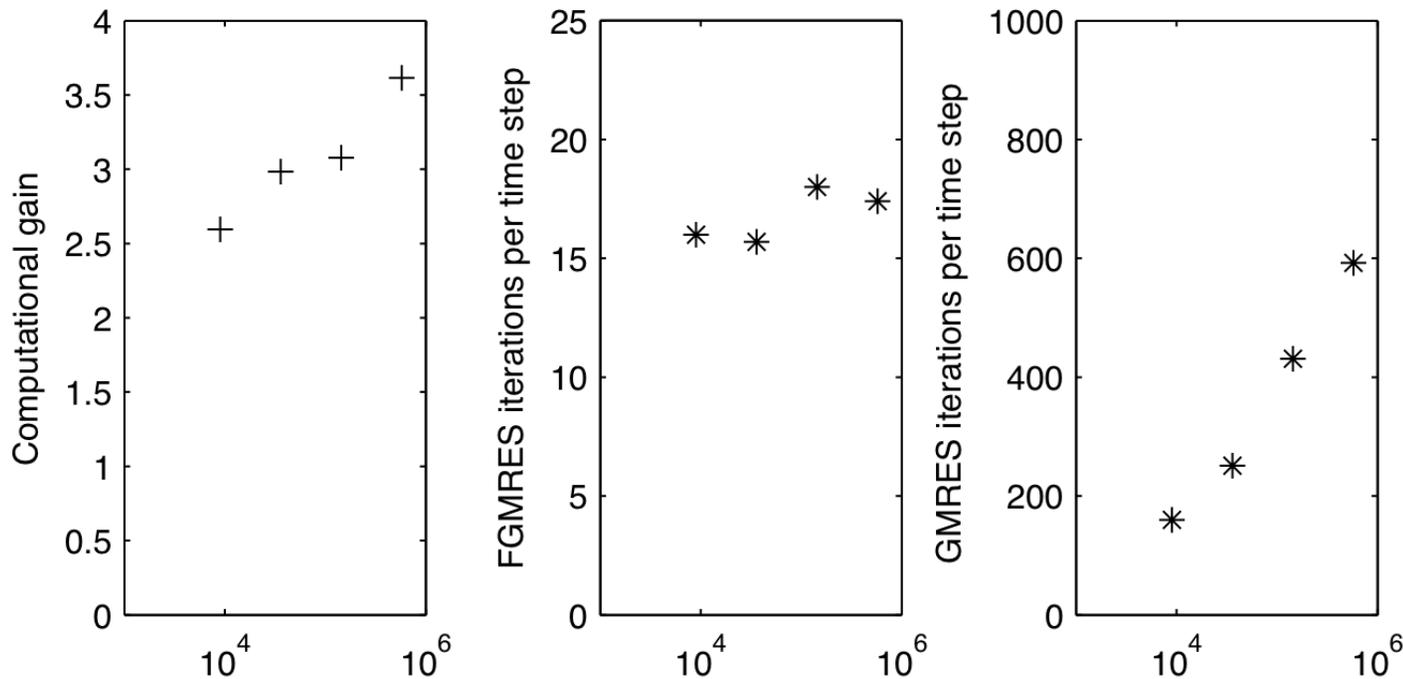
Blue line: JFNK

Black line: inexact Newton (looser linear tolerance)

Preconditioner: the key to solution efficiency

- Physics Based Preconditioning to JFNK produces robust and efficient solution updates for a number of multiphysics applications (fluids, phase transition, chemical transport)
- Reduce, reuse, recycle
 - Existing Picard solution method as preconditioner within new JFNK solver
 - As approximate update, use Picard and FGMRES with a loose tolerance and Ifpack (Jacobi) preconditioner
 - Current solver can become preconditioner to more complete models coming down the road, e.g. full Stokes
- Need to boost physics-based preconditioning with scalable algorithm: multilevel methods (multigrid, Schwarz)
 - Enhanced efficiency for a given problem
 - More linear scaling than physics-based preconditioning alone

Picard preconditioner: proof of principle



Relative efficiency improves with problem size, but is currently limited (in speed and memory) by use of GMRES iterations in the preconditioner.

Lemieux, J.F., S. Price, K. Evans, D. Knoll, A. Salinger, D. Holland, T. Payne, (2011) J. Comp. Phys. In Press.

Initial parallel Picard velocity solver in CESM

Linear Solver	Preconditioner	# Procs	GMRES iterations per solve	Time per solve [s]
GMRES	None	1	Fail	Fail
GMRES	ILU (Fill-in=0)	1	38	0.938
GMRES	ILU (Fill-in=1)	1	30	0.774
GMRES	Multi-Level (Default Settings)	1	45	3.31
KLU	None (KLU is a Direct Solver)	1	--	25.0
GMRES	DD-ILU (Overlap=0, Fill-in=0)	24	52	0.156
GMRES	DD-ILU (Overlap=0, Fill-in=1)	24	48	0.164
GMRES	DD-ILU (Overlap=2, Fill-in=3)	24	21	0.182
GMRES	DD (Overlap=0, Direct Solver on each Domain)	24	44	0.225
GMRES	DD (Overlap=3, Direct Solver on each Domain)	24	9	0.134
GMRES	Multi-Level (Default Settings)	24	46	0.208

Solving a time step for a model problem with 69792 unknowns for a set of linear solver and preconditioner choices, as selected through the Trilinos input file.

The iteration count and solution time are for the final linear solve out of the 82 needed to converge the nonlinear problem with the Picard iteration. Ideal settings are problem-dependent and open areas in algorithmic research, and the current interface provides a large degree of flexibility.

Distributed Parallel CISM

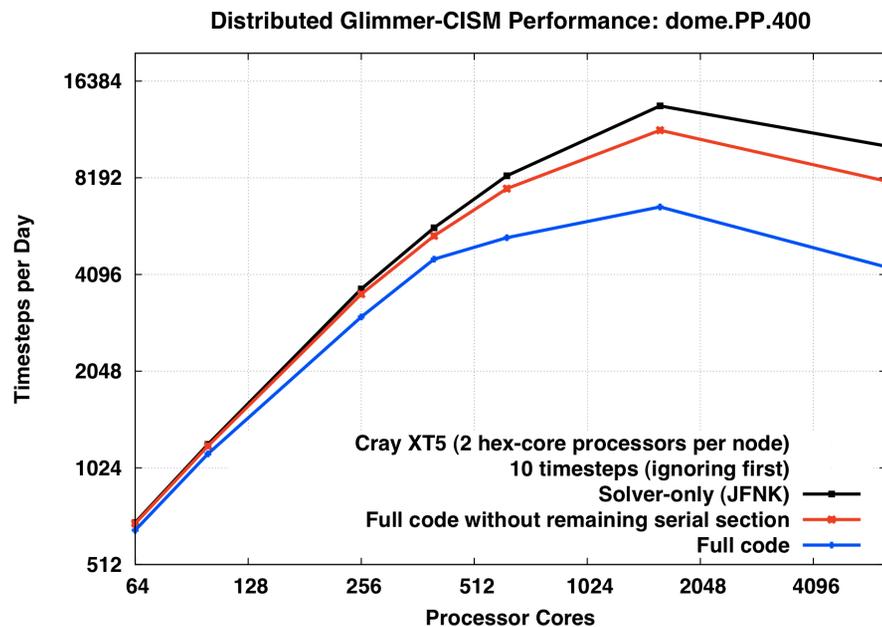
- Initial implementation
 - Ported to Jaguar* and Bluefire
 - Distributed-memory parallelism
 - Ice Dome test cases
- Improve performance/memory use
 - Trilinos interface, parameters
 - Take advantage of CESM
 - Parallel I/O
- Extend to temperature, vertical remapping (Bill Lipscomb)
- Next: Port to Intrepid at ALCF



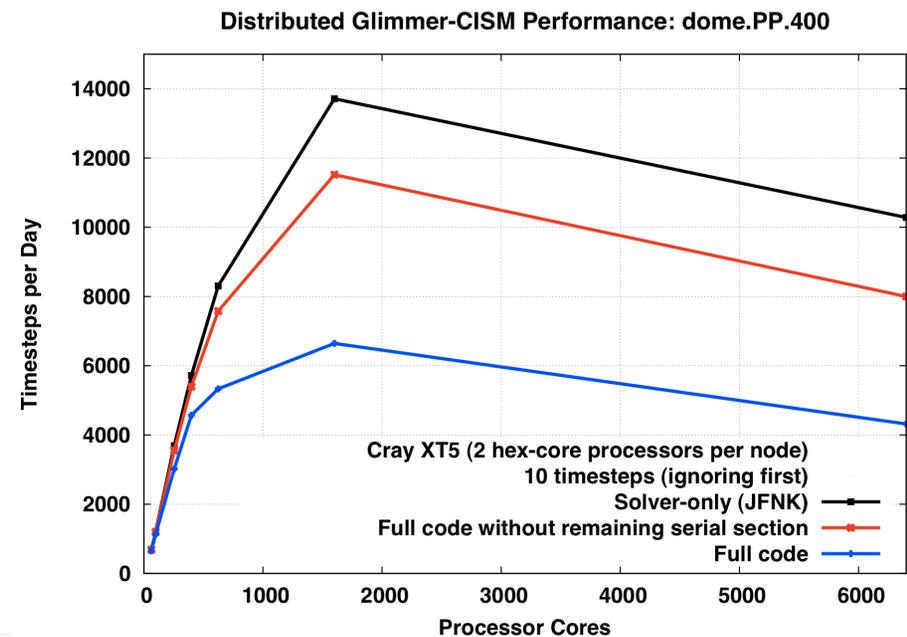
*SEACISM uses an ALCC allocation from DOE-ASCR to develop Glimmer CISM at scale.

Performance behavior of 'dome' case with JFNK, fully decomposed parallel velocity solve

JFNK solver scaling/performance



JFNK solver expense evaluation



- Parallelized the existing Picard velocity solver first & port to Bluefire to give quick gains to modelers (version V2.0 now in CESM repo)
- now fully decomposed HO for dome test case, save T and thickness advection
- Several 'to do' items limit scaling and/or memory: e.g. preconditioner, I/O

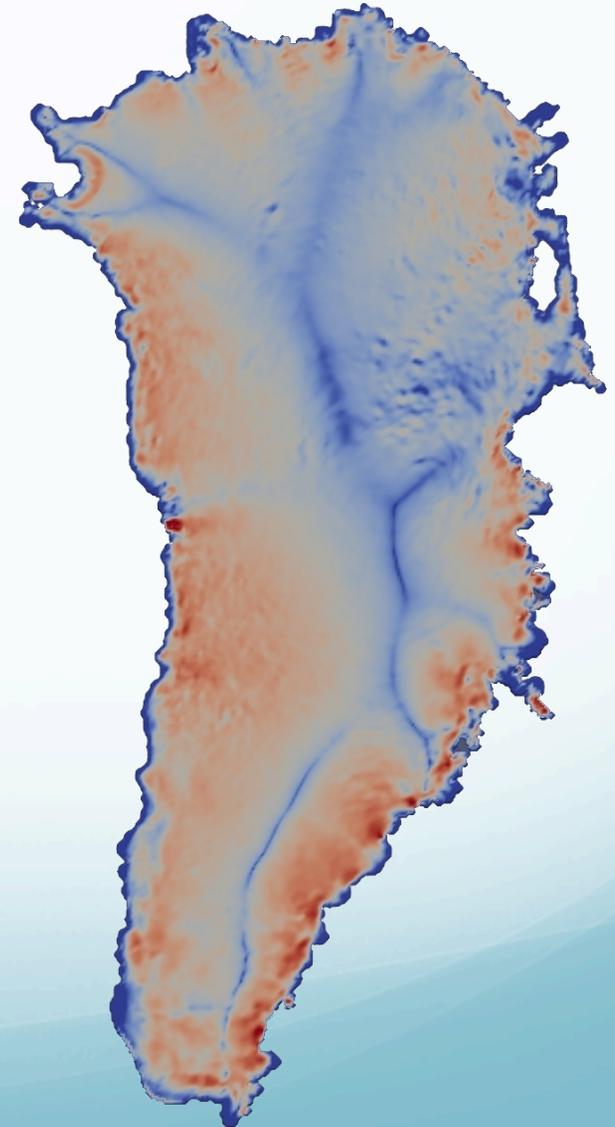
Finite Element implementation of higher-order models



Finite elements naturally handle:
complex geometries, unstructured
meshes and several type of boundary
conditions

- Linear or quadratic finite elements are implemented using the C++ library LifeV¹.
- nonlinear solver: Newton method, NOX.
- Models: First Order, L1L2, SSA, SIA.
- Boundary conditions: free slip, stress free, Dirichlet, Robin, Coulomb Friction like.

Different models and numerical approximations have been tested and compared on ISIMP-HOM benchmarks. First simulations on Greenland.



¹ www.lifev.org

Near term efforts within Glimmer-CISM

- Version with parallel Trilinos solver has been ported to Glimmer-CISM trunk.
- IEEE special issue paper submitted on the software frameworks with Glimmer-CISM explains software developments
- Extend parallelization refactored temperature, and vertical remap (CFL limited) portions of code
- Work with ice sheet modelers to design science sims of interest
- Optimize current preconditioner for JFNK solver to improve scalability (e.g.ML)
- Continue to optimize and tune existing code changes for robustness

Thank you. Cites mentioned in talk:

Price, S.F., A.J. Payne, I.M. Howat, B.E. Smith (2011) "Committed sea-level rise for the next century from Greenland ice sheet dynamics during the past decade," Proceedings of the National Academy of Sciences of the U.S., submitted.

Bamber et al. (2000) "An analysis of balance velocities over the Greenland ice sheet and comparison with synthetic aperture radar interferometry," Journal of Glaciology 46:67-74.

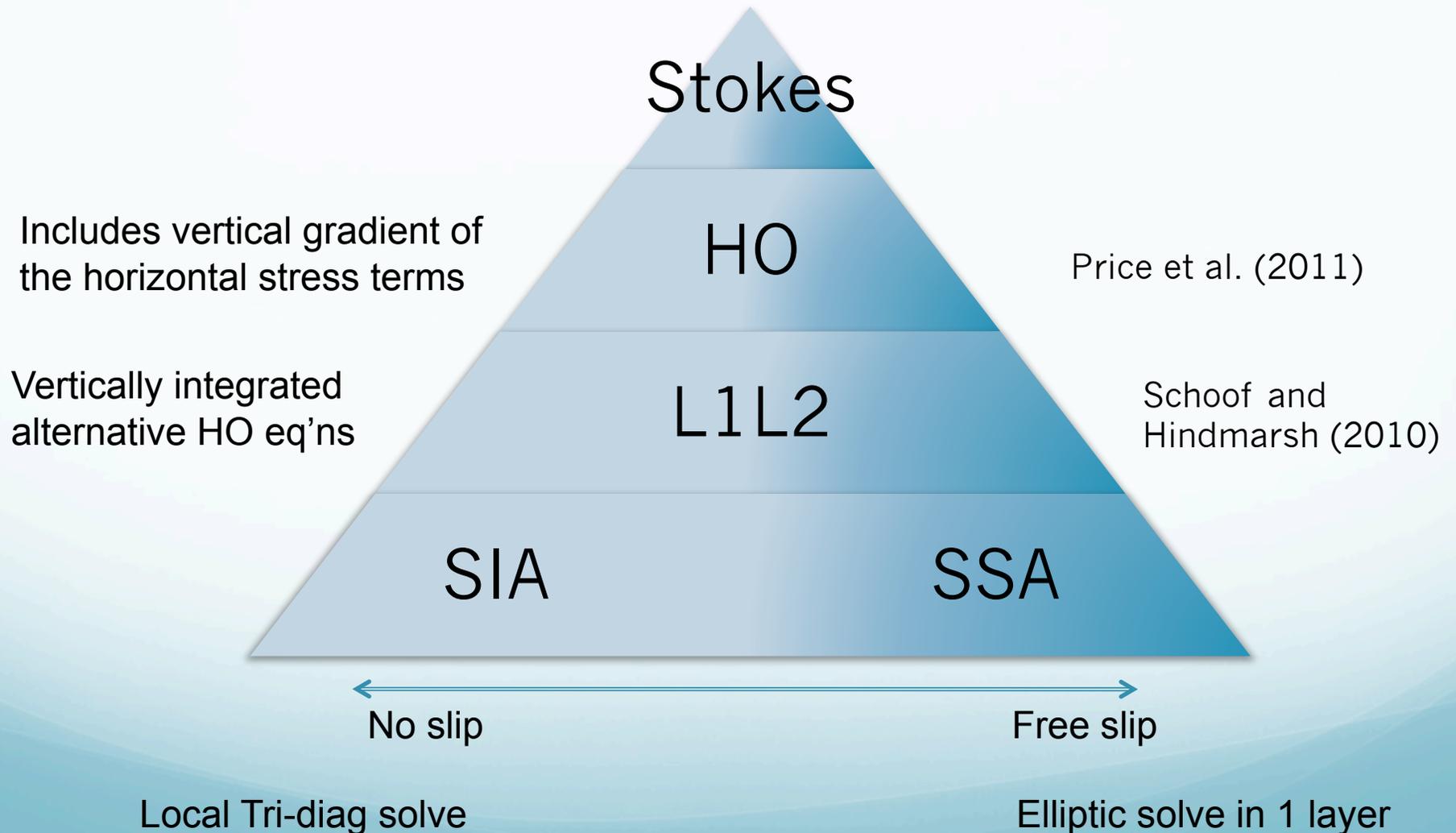
Lemieux, J.F., S. Price, K. Evans, D. Knoll, A. Salinger, D. Holland, T. Payne (2011) "Implementation of the Jacobian-free Newton-Krylov method for solving the first-order ice sheet momentum balance" J. Comp. Phys, in press.

Schoof, C. and R. Hindmarsh (2010) "Thin-film flows with wall slip: An asymptotic analysis of high-order glacier from models," Q. J. Appl. Mech. Math. 63:73-114.

Evans, K., A. Salinger, P. Worley, S. Price, W. Lipscomb, J. Nichols, J. White, M. Perego, M. Vertenstein, J. Edwards, and J.-F. Lemieux (2011). "A modern solver framework to manage solution algorithms in the Community Earth System Model," IEEE software, submitted.

Questions?

'Higher Order/First-order' set of 3D momentum and mass balance equations; intermediate complexity



Incorporate SEA-Solvers: Picard vs Newton

$$F'(x^{t+1})\delta x = -f(x^{t+1})$$

KEY: Picard is a simpler form of Newton

$$F'_p = A + \frac{1}{\delta t} F \quad \text{Picard}$$

$$F'_{ij} = A_{ij} + \frac{1}{\delta t} F_{ij} + \sum_s \frac{\partial A_{is}}{\partial x_j} x_s + \frac{1}{\delta t} \sum_s \frac{\partial F_{is}}{\partial x_j} (x_s^{t+1} - x_s^t) + \frac{\partial b_i}{\partial x_j} \quad \text{Newton}$$

Use Inexact Newton to solve for x:

solve top equation with preconditioned GMRES method

$$x^{k+1} = x^k + \delta u^k$$

if $\|F(x^{k+1})\| < \gamma_{nl} \|F(x^0)\|$ stop

end do

- Use JFNK approach: $J(x^k)v \sim (F(x^k + \varepsilon v) - F(x^k)) / \varepsilon$
- Develop a physics-based preconditioner and combine with multilevel options available through Trilinos

JFNK behavior for suite of test cases

