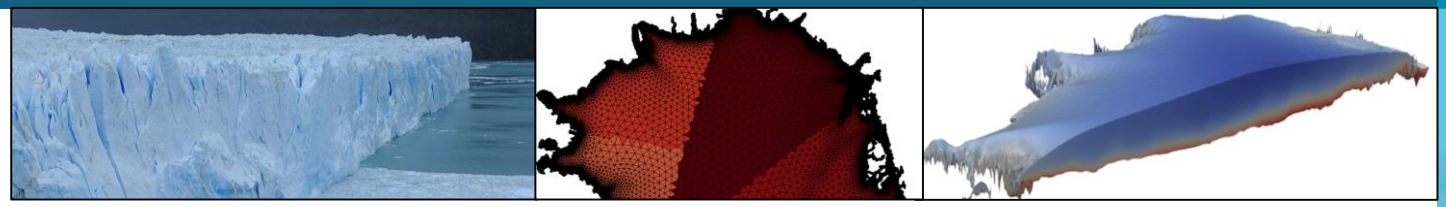
PDE-Constrained Optimization for Ice-Sheet Model Initialization in MALI



Mauro Perego and Kim Liegeois

Main collaborators:

L. Bertagna, M. Carlson, I Tezaur, J. Watkins, (Albany team, Sandia)
E. Phipps, D. Kouri, D. Ridzal, (Sandia National Labs)
G. Stadler (Courant Institute)
T. Hillebrand, M. Hoffman, S. Price (Los Alamos National Lab)

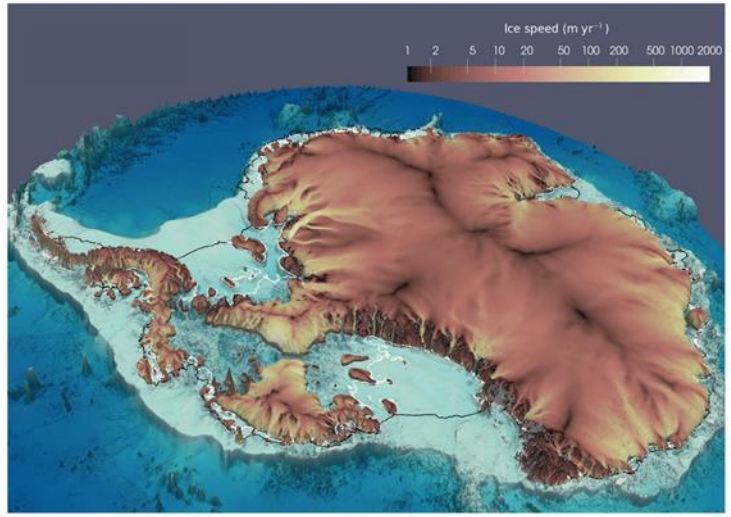
TUG 2023

SAND2023-11908PE

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Talk Outline

- Brief motivation and introduction to ice sheet models
- MALI ice sheet model
- Ice sheet initialization
  - ➢ Problem definition
  - ➢ Hessian computation
  - ➢ PDE-constrained optimization approach
- Final considerations



Supported by US DOE Office of Science projects:

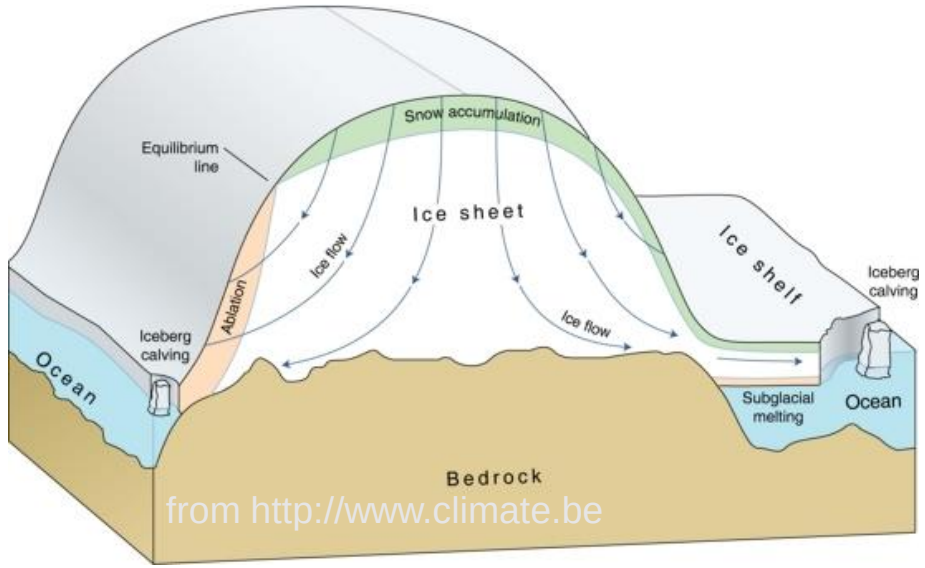
- ➢ ***ProSPect**: Probabilistic Sea-Level Projections from ice sheets and Earth System Models*
- ➢ ***FASTMath**: Frameworks, Algorithms and Scalable Technologies for Mathematics*
- ➢ ***E3SM**: Energy Exascale Earth System Model*
- ➢ ***FAnSSIE**: Framework for Antarctic System Science in E3SM*

Brief Motivation an basic physics

- Modeling ice sheets (Greenland and Antarctica) dynamics is essential to provide estimates for sea-level rise in next decades to centuries.

- Ice behaves like a very viscous shear-thinning fluid (similar to lava flow) driven by gravity.

- A critical step in ice-sheet modeling is to estimate the unknown or poorly known parameters (e.g. basal friction, bed topography) and the initial thermo-mechanical state of the ice - Initialization



Model: Ice velocity equations

Stokes equations:

$$\begin{cases} -\nabla \cdot \sigma = \rho \mathbf{g} \\ \nabla \cdot \mathbf{u} = 0 \end{cases}$$

gravit. acceleration

ice velocity

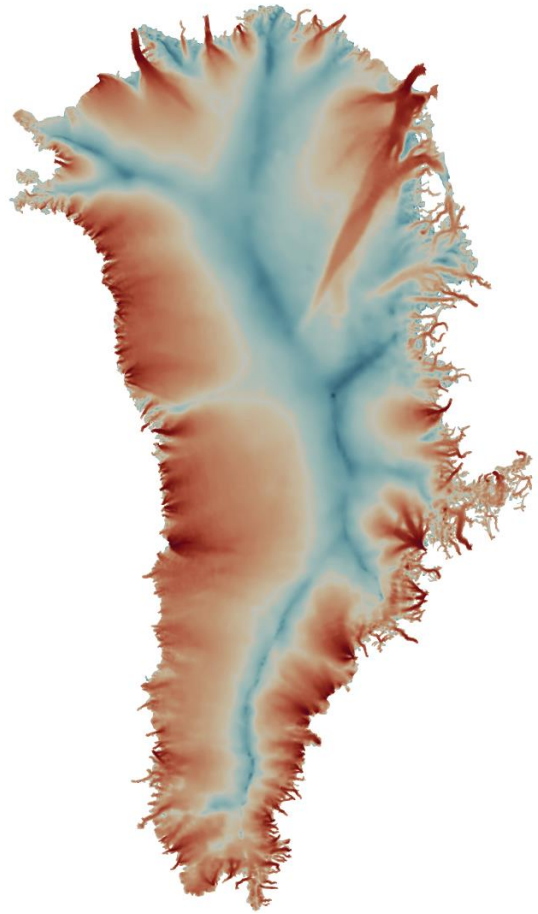
Stress tensor:

$$\sigma = 2\mu \mathbf{D} - pI, \qquad \mathbf{D}_{ij}(\mathbf{u}) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$

Ice viscosity (dependent on temperature):

$$\mu = \frac{1}{2} A(T) \, |\mathbf{D}(\mathbf{u})|^{\frac{1}{n}-1}, \quad n \geq 1, \quad \text{(tipically } n \simeq 3)$$

Modeled surface ice speed



In this work we use a simplification of Stokes equations, called **First Order** equations, obtained by scaling arguments given the shallow nature of the ice sheets and using hydrostatic pressure.

Model: Ice velocity equations

Stokes equations:

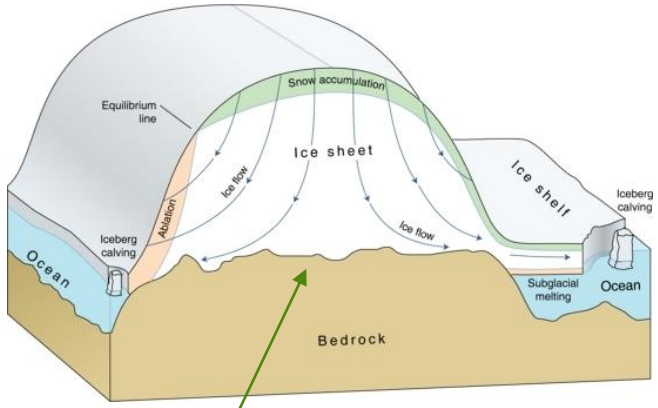
$$\begin{cases} -\nabla \cdot \sigma = \rho \mathbf{g} \\ \nabla \cdot \mathbf{u} = 0 \end{cases}$$

Sliding boundary condition at ice bed:

$$\begin{cases} \mathbf{u} \cdot \mathbf{n} = 0, \quad \text{(impenetrablity)} \\ (\sigma \mathbf{n})_{\|} = \beta \mathbf{u} \end{cases}$$

Free slip: $\beta = 0$

No slip: $\beta = \infty$



bed

Model: Temperature equation

Heat equation (for cold ice):

$$\rho c\, \partial_t T + \nabla \cdot (k \nabla T) + \rho c\, \mathbf{u} \cdot \nabla T = 4\mu |D(\mathbf{u})|^2$$

conductivity — heat capacity — dissipation heating

Boundary condition at the ice bed (includes melting and refreezing):

$$\mathrm{m} = \mathrm{G} + \beta |\mathbf{u}|^2 - k \nabla T \cdot \mathbf{n}$$

melting rate — geothermal heat flux — frictional heating — temperature flux

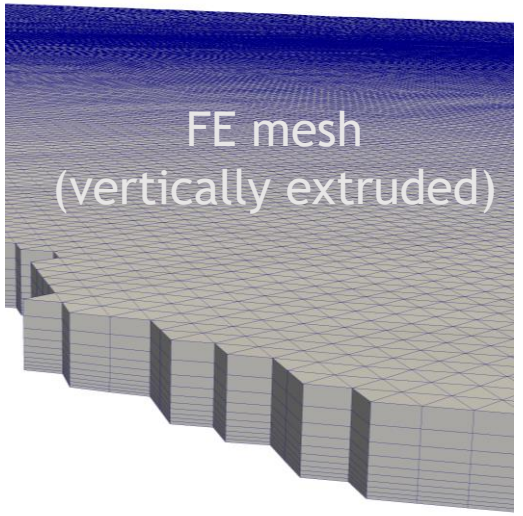
modeled temperature

In this work we use a enthalpy formulation that accounts for temperate ice as well.

[1] A. Aschwanden, E. Bueler, C. Khroulev, and H. Blatter, Journal of Glaciology, 2012
[2] J. Hewitt and C. Schoof, The Cryosphere, 2017

Software: MPAS-Albany Land Ice model (MALI)

ALGORITHM	SOFTWARE TOOLS
Thickness evolution / Temperature Finite Volumes on Voronoi meshes	MPAS (Model for Prediction Across Scales)
Velocity/ SS Enthalpy solvers: Finite Elements on prisms	Albany
Optimization	ROL
Nonlinear solver (Newton method)	NOX
Krylov linear solvers/Prec	Belos/MueLu, Belos/FROSch
Automatic differentiation	Sacado



MALI relies on **Trilinos** for achieving performance portability through ***Kokkos*** programming model. And for providing large-scale PDE constrained optimization capabilities.

References:

1. Watkins et al., IJHPCA 2023
2. Liegeois, Perego, Hartland, J. Comput. Appl., 2023
3. Heinlein, Perego, Rajamanickam, SISC, 2022
4. Hoffman et al. GMD, 2018
5. Tuminaro, Perego, Tezaur, Salinger, Price, SISC, 2016.
6. Tezaur, Perego, Salinger, Tuminaro, Price, Hoffman, GMD, 2015
7. Perego, Price, Stadler, JGR, 2014

Ice sheet initialization

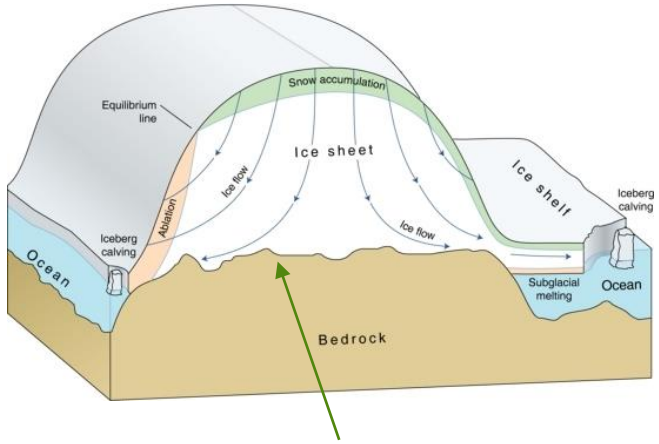
Goal: Find the initial/present-day thermo-mechanical state of the ice sheet and estimate the unknown/poorly known model parameters, by matching observations

Approach: **PDE-constrained optimization**

Find basal friction coefficient $\beta = \exp(p)$ that minimizes the mismatch with surface velocity:

$$\min_p \mathcal{J}(p,u) = \int_\Omega \frac{|u - u_{obs}|^2}{\sigma^2} + \mathcal{R}(p)$$

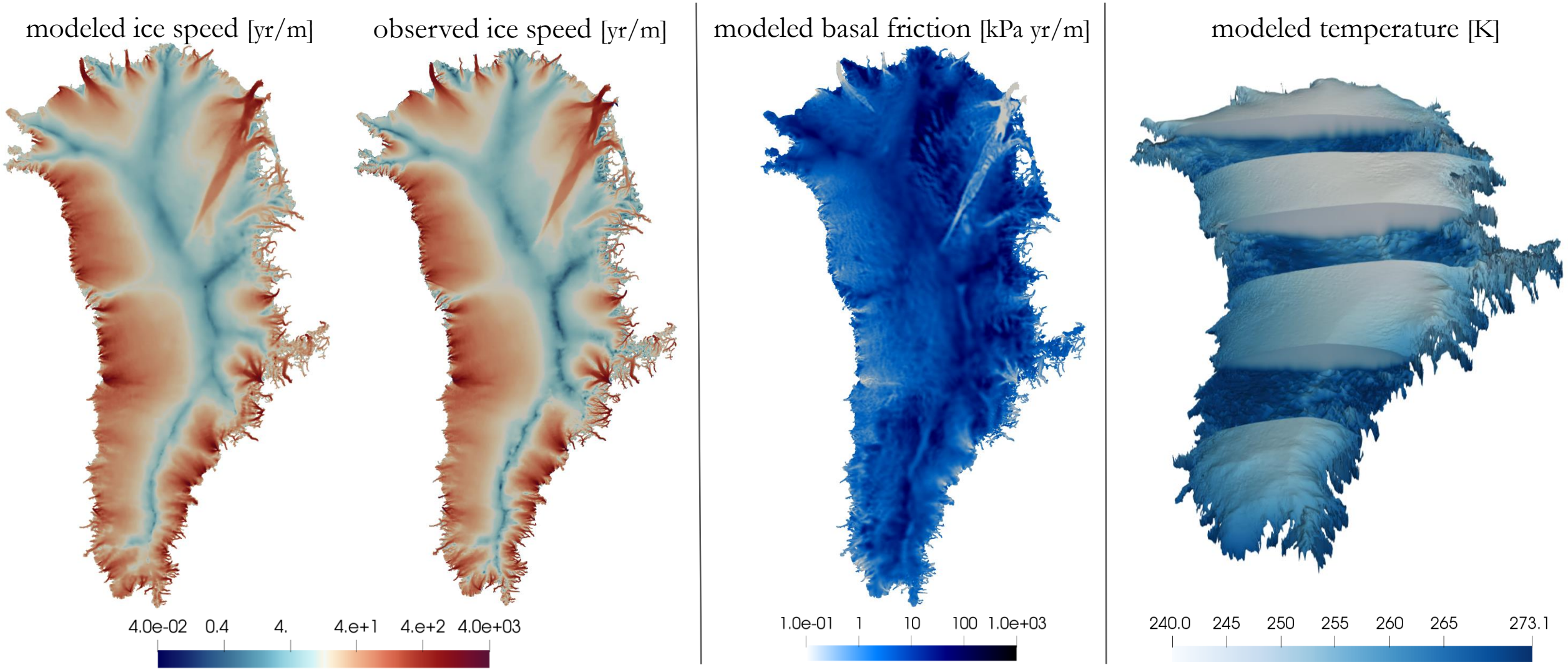
Subject to the coupled velocity/temperature problem (constraint).
The constraint maps $p \to u(p)$.



unknown sliding parameter $\beta = \exp(p)$

Typical regularization term: $\mathcal{R}(p) = \alpha_0 |p|^2 + \alpha_1 |\nabla p|^2$

Thermo-mechanical initialization of Greenland ice sheet



Variable resolution 1-10km mesh**, 300K parameters, 14M unknowns.** Initialization: ~10 hours on 2k nodes on NERSC Cori (Haswell)
The optimization is constrained by the **coupled velocity-temperature** solvers. As a byproduct of the optimization we get an initial temperature field that is consistent with the velocity

Ice sheet initialization

Numerical Optimization approach:

- **Reduced-space** approach (objective functional is considered a function of the parameter only, with the solution computed by solving the constraint for any given parameter) $\mathcal{J}(p) = \mathcal{J}(p, u(p))$

- **Trust Region method (Lin-Moré)**, using truncated CG for solving the quadratic subproblem. Requires computation of reduced gradient ($\partial_p \mathcal{J}$, first total derivative of $\mathcal{J}$) and reduced Hessian ($\partial_{pp} \mathcal{J}$, second total derivative of $\mathcal{J}$) to create a quadratic approximation of the objective $\mathcal{J}$.

- All first and second order derivatives are computed using **Automatic Differentiation** and **adjoints**.

Ice sheet initialization

Hessian-vector product computations using automatic differentiation

Trust Region optimization methods with Krylov solvers require Hessian mat-vec products:

Hessian of residual $\boldsymbol{f}$ dotted with the Lagrange multiplier $\boldsymbol{\lambda}$ in the direction $\boldsymbol{v}$:

$$\partial_{uu}(\lambda^T \boldsymbol{f}(\boldsymbol{u},\boldsymbol{p}))\,\boldsymbol{v}, \quad \partial_{up}(\lambda^T \boldsymbol{f}(\boldsymbol{u},\boldsymbol{p}))\,\boldsymbol{v},$$
$$\partial_{pu}(\lambda^T \boldsymbol{f}(\boldsymbol{u},\boldsymbol{p}))\,\boldsymbol{v}, \quad \partial_{pp}(\lambda^T \boldsymbol{f}(\boldsymbol{u},\boldsymbol{p}))\,\boldsymbol{v}$$

Computed w/ **automatic differentiation**, differentiating twice, based on the formula:

$$\partial_{pp}\,\mathcal{J}(\boldsymbol{p})\,\boldsymbol{v} = \partial_r \left(\partial_{\boldsymbol{p}}\,\mathcal{J}(\boldsymbol{p} + r\,\boldsymbol{v})\right)\Big|_{r=0}$$

The Hessian-vector product is evaluated using nested ***Sacado*** Forward automatic differentiation types. Any scalar value that implicitly depends on $\boldsymbol{u}$ or $\boldsymbol{p}$ is now a 2D array of data:

	value	**Dx(0)**	**Dx(1)**	**Dx(2)**	**…**
value	.value.value	.Dx(0).value	.Dx(1).value	.Dx(2).value	…
Dx(0)	.value.Dx(0)	.Dx(0).Dx(0)	.Dx(1).Dx(0)	.Dx(2).Dx(0)	…

Ice sheet initialization

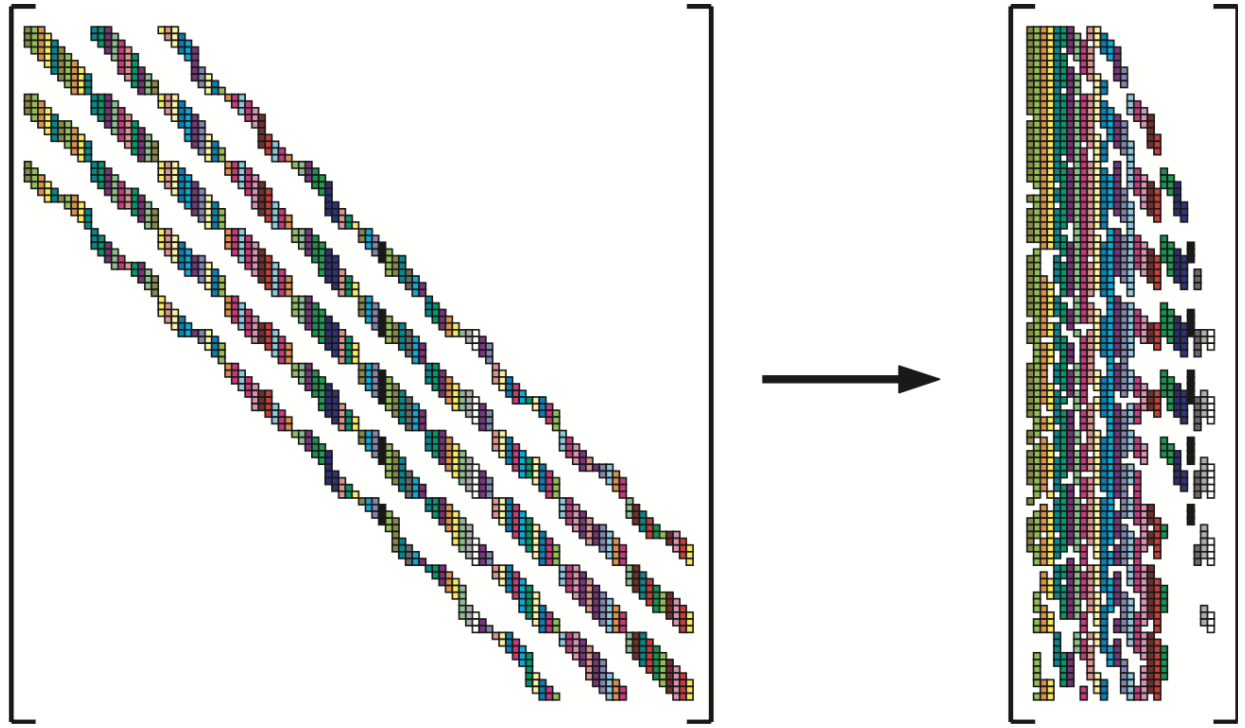
Partial Hessian reconstruction

It is sometimes convenient to explicitly compute the Hessian matrix.
This is the case when the Hessian is sparse with a known pattern (e.g. for a gradient squared regularization, the Hessian of regularization term is a Stiffness matrix).
In this case, we can reconstruct the matrix performing a small number of matrix-vector products.
This is achieved using coloring and seeding schemes.

In order to reconstruct a matrix representation of the partial Hessian of the response with respect to a nodal parameter, we:

1. Precompute the graph of the partial Hessian based on the finite element mesh,
2. Color the graph using Zoltan2,
3. Apply one Hessian-vector product per color,
4. Reconstruct the partial Hessian.

Coloring and seedling implemented by E. Phipps within ATDM project



A. H. Gebremedhin, F. Manne, A. Pothen, What Color Is Your Jacobian? Graph Coloring for Computing Derivatives, SIAM Review, 2005

Ice sheet initialization: PDE constrained optimization

Optimization loop, *O*(100) iterations

Value of objective ($\mathcal{J}$): (state solve)

Newton loop *O*(10) iters

Linear solve

Reduced Gradient ($\partial_p \mathcal{J}$):

Adjoint solve

Quadratic subproblem:
($\mathcal{H}\,\delta_p = -\partial_p \mathcal{J}, \quad \mathcal{H} = \partial_{pp}\mathcal{J}$)

Truncated CG loop, *O*(20) iters

Reduced Hessian action:

Adjoint solve 1

Adjoint solve 2

Data from a relatively simple optimization (Humboldt glacier on a coarse mesh):
state solves: 116
reduced gradients: 98
reduced Hessian: 1568
total linear solves (including adjoints): 3821

RAPID OPTIMIZATION LIBRARY

Ice sheet initialization: PDE constrained optimization

➢ Cost is often dominated by evaluation of reduced Hessian (and in particular by the assembly phase)

Possible strategies to reduce cost:

- Change the dot-product used to define the gradient $\mathcal{G}$, defined by $\partial_p \mathcal{J}\, v = (\mathcal{G}, v)$.
  Instead of $(u, v)_{l^2} = u^t v$, use $(u, v)_{L^2} = u^t M v$.
  Here M is the lumped mass matrix.
  Hence, $\mathcal{G} := M^{-1}\, \partial_p \mathcal{J}$.

  → Often reduces the number of optimization iterations. Particularly effective when using nonuniform meshes

- Find preconditioner for the reduced Hessian. Possibilities:
  - Use a low-rank approximation of the reduced Hessian (e.g., BFGS) as preconditioner,
  - Use $\partial_{pp} \mathcal{R}(p)$ to approximate reduced hessian (e.g. to initialize BFGS)

  → Improves convergence of CG solver for the subproblem

- Replace Hessian with low-rank approximation (e.g. BFGS)

  → Avoids expensive computation of Hessians but it can badly affect convergence of the optimization method

Ice sheet initialization: PDE constrained optimization

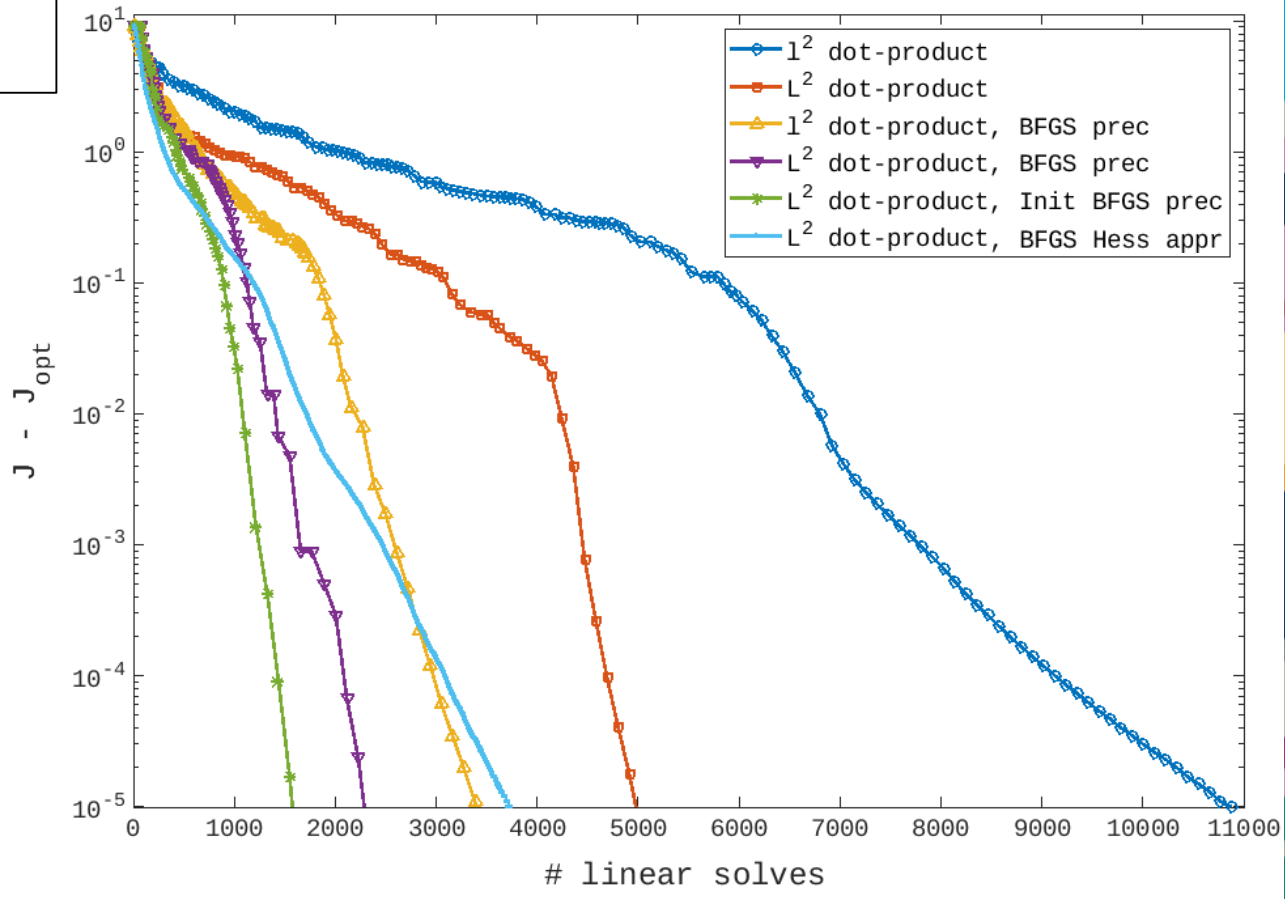
➢ Cost is often dominated by evaluation of reduced Hessian (and in particular by the assembly phase)

Possible strategies to reduce cost:

- Change the dot-product used to define the gradient $\mathcal{G}$, defined by $\partial_p \mathcal{J}\, v = (\mathcal{G}, v)$.
  Instead of $(u, v)_{l^2} = u^t v$, use $(u, v)_{L^2} = u^t M v$.
  Here M is the lumped mass matrix.
  Hence, $\mathcal{G} := M^{-1}\, \partial_p \mathcal{J}$.

- Find preconditioner for the reduced Hessian. Possibilities:
  - Use a low-rank approximation of the reduced Hessian (e.g., BFGS) as preconditioner,
  - Use $\partial_{pp}\mathcal{R}(p)$ to approximate reduced hessian (e.g. to initialize BFGS)

- Replace Hessian with low-rank approximation (e.g. BFGS)



Problem: Optimization to initialize Humboldt glacier
Ice velocity depends on friction parameter. Ice temperature held constant.

Ice sheet initialization: PDE constrained optimization

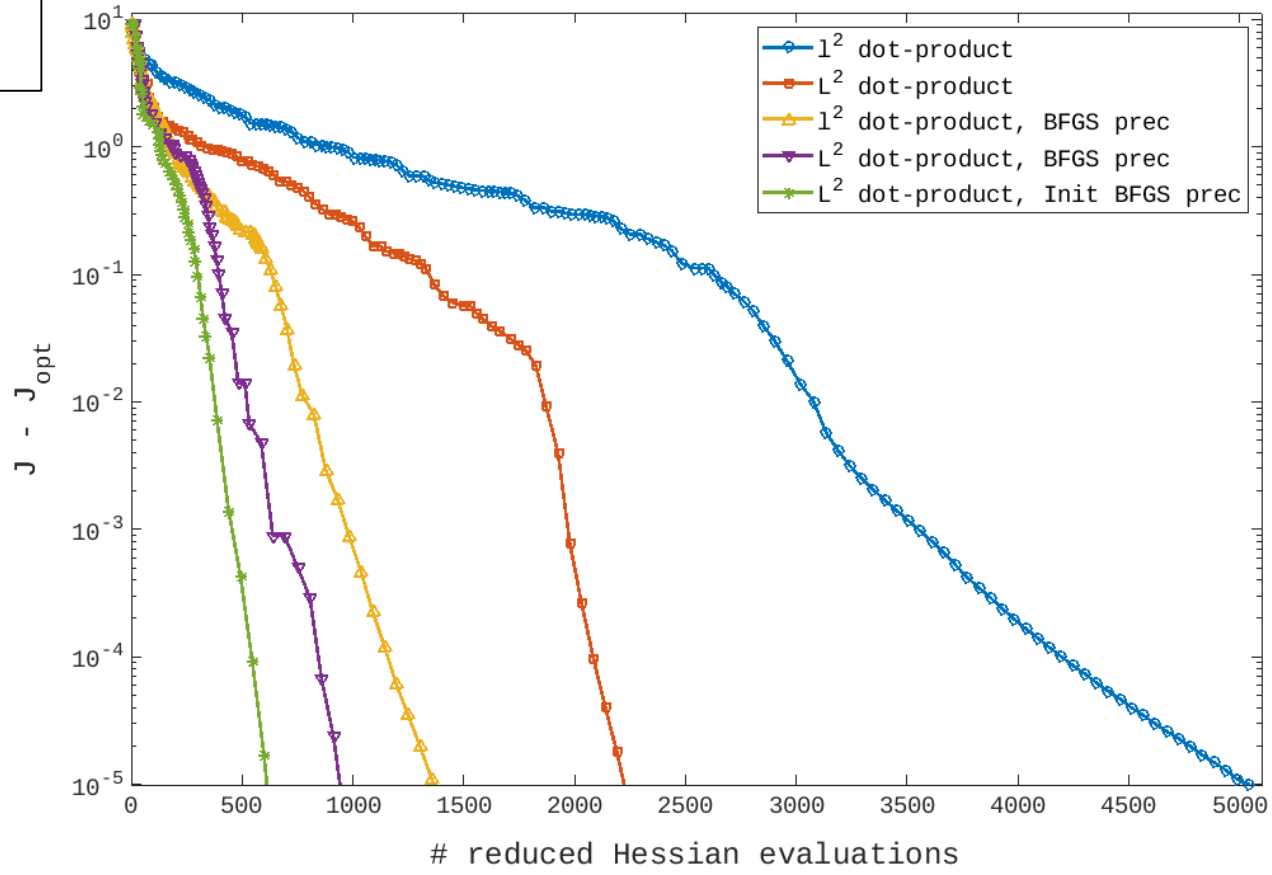
➢ Cost is often dominated by evaluation of reduced Hessian (and in particular by the assembly phase)

Possible strategies to reduce cost:

- Change the dot-product used to define the gradient $\mathcal{G}$, defined by $\partial_p \mathcal{J}\, v = (\mathcal{G}, v)$.
  Instead of $(u, v)_{l^2} = u^t v$, use $(u, v)_{L^2} = u^t M v$.
  Here M is the lumped mass matrix.
  Hence, $\mathcal{G} := M^{-1}\, \partial_p \mathcal{J}$.

- Find preconditioner for the reduced Hessian. Possibilities:
  - Use a low-rank approximation of the reduced Hessian (e.g., BFGS) as preconditioner,
  - Use $\partial_{pp} \mathcal{R}(p)$ to approximate reduced hessian (e.g. to initialize BFGS)

- Replace Hessian with low-rank approximation (e.g. BFGS)



Problem: Optimization to initialize Humboldt glacier
Ice velocity depends on friction parameter. Ice temperature held constant.

Ice sheet initialization: PDE constrained optimization

➢ Cost is often dominated by evaluation of reduced Hessian (and in particular by the assembly phase)

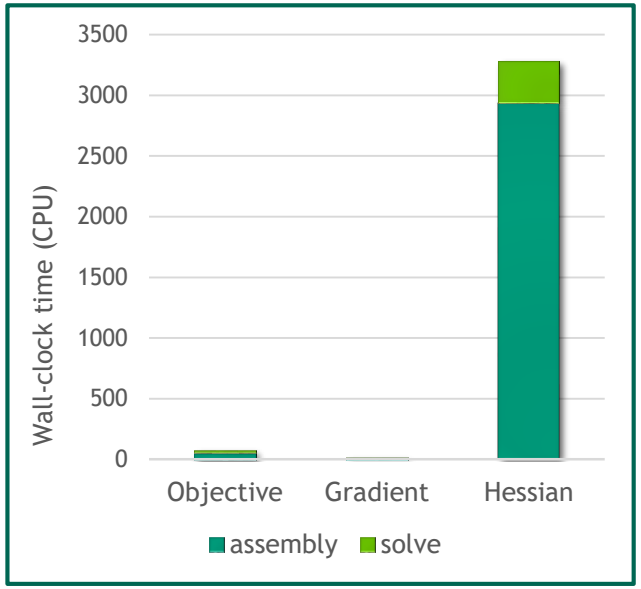
Possible strategies to reduce cost:

- Change the dot-product used to define the gradient $\mathcal{G}$, defined by $\partial_p \mathcal{J}\, v = (\mathcal{G}, v)$.
  Instead of $(u, v)_{l^2} = u^t v$, use $(u, v)_{L^2} = u^t M v$.
  Here M is the lumped mass matrix.
  Hence, $\mathcal{G} := M^{-1}\, \partial_p \mathcal{J}$.

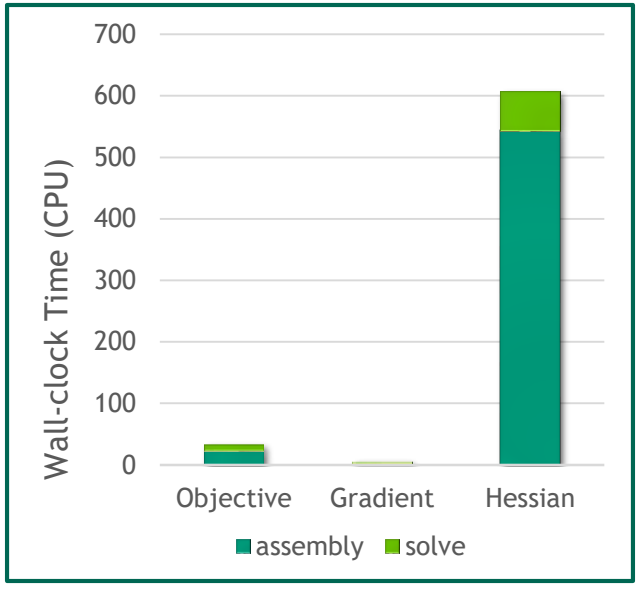
- Find preconditioner for the reduced Hessian. Possibilities:
  - Use a low-rank approximation of the reduced Hessian (e.g., BFGS) as preconditioner,
  - Use $\partial_{pp} \mathcal{R}(p)$ to approximate reduced hessian (e.g. to initialize BFGS)

- Replace Hessian with low-rank approximation (e.g. BFGS)

l^2 dot-product



L^2 dot-product, Init BFGS prec.



Problem: Optimization to initialize Humboldt glacier Ice velocity depends on friction parameter. Ice temperature held constant.

Final considerations

- Initialization via PDE-constrained optimization requires a large number of linear solves and computation of derivatives
- The cost of optimization with Trust Region method is in large part dominated by assembling the terms needed for computing the reduced Hessian
- Approaches that limit the number of reduced Hessian evaluations, like low-rank preconditioners, significantly improve performance
- The high assembly cost of computing Hessian derivatives should be greatly reduced on GPUs.