Bathymetry and stickyness distribution

- **Bathymetry:**
  - Aspect ratio $\varepsilon = [H]/[x] \ll 1$
  - Need surface and bed slopes to be small

- **Stickyness distribution:**
  - Limiting cases of plug flow versus vertical shear
  - Stress ratio: $\lambda = [\tau_{xz}]/[\tau_{membrane}]$
  - Discontinuous: frozen to slippery transition at ice stream margins

- **Standard approach in glaciology:**
  Taylor expand in $\varepsilon$ and sometimes $\lambda$, drop higher order terms.

  $\lambda \gg 1$ Shallow Ice Approximation (SIA), no horizontal coupling
  $\lambda \ll 1$ Shallow Shelf Approximation (SSA), 2D elliptic solve in map-plane
    - Hydrostatic and various hybrids, 2D or 3D elliptic solves

- **Bed slope is discontinuous and of order 1.**
  - Taylor expansions no longer valid
  - Numerics require high resolution, subgrid parametrization, short time steps
  - Still get low quality results in the regions of most interest.

- **Basal sliding parameters are discontinuous.**
Hydrostatic equations for ice sheet flow

- Valid when \( w_x \ll u_z \), independent of basal friction (Schoof&Hindmarsh 2010)
- Eliminate \( p \) and \( w \) from Stokes by incompressibility:
  3D elliptic system for \( u = (u, v) \)

\[
- \nabla \cdot \left[ \eta \begin{pmatrix} 4u_x + 2v_y & u_y + v_x & u_z \\ u_y + v_x & 2u_x + 4v_y & v_z \end{pmatrix} \right] + \rho g \nabla h = 0
\]

\[
\eta(\theta, \gamma) = \frac{B(\theta)}{2} (\gamma_0 + \gamma)^{\frac{1-n}{2n}}, \quad n \approx 3
\]

\[
\gamma = u_x^2 + v_y^2 + u_x v_y + \frac{1}{4} (u_y + v_x)^2 + \frac{1}{4} u_z^2 + \frac{1}{4} v_z^2
\]

and slip boundary \( \sigma \cdot n = \beta^2 u \) where

\[
\beta^2(\gamma_b) = \beta_0^2 \left( \varepsilon_b^2 + \gamma_b \right)^{\frac{m-1}{2}}, \quad 0 < m \leq 1
\]

\[
\gamma_b = \frac{1}{2} (u^2 + v^2)
\]

- \( Q_1 \) FEM with Newton-Krylov-Multigrid solver in PETSc:
  src/snes/examples/tutorials/ex48.c
Grid-sequenced Newton-Krylov solution of test \( X \). The solid lines denote nonlinear iterations, and the dotted lines with \( \times \) denote linear residuals.
- Bathymetry is essentially discontinuous on any grid
- Shallow ice approximation produces oscillatory solutions
- Nonlinear and linear solvers have major problems or fail
- Grid sequenced Newton-Krylov multigrid works as well as in the smooth case
Figure: Grid sequenced Newton-Krylov convergence for test $Y$. The “cliff” has $58^\circ$ angle in the red line ($12 \times 125$ meter elements), $73^\circ$ for the cyan line ($6 \times 62$ meter elements).
Polythermal ice

- Interface tracking methods (Greve’s SICOPOLIS)
  - Different fields for temperate and cold ice.
  - Lagrangian or Eulerian, problems with changing topology
  - No discrete conservation

- Interface capturing
  - Enthalpy: Aschwanden, Buerler, Khroulev, Blatter (J. Glac. 2012)
    - Not in conservation form
    - Only conservative for infinitesimal melt fraction
  - Energy
    - Conserves mass, momentum, and energy for arbitrary melt fraction
    - Implicit equation of state
Conservative (non-Boussinesq) two-phase ice flow

Find momentum density $\rho u$, pressure $p$, and total energy density $E$:

$$(\rho u)_t + \nabla \cdot (\rho u \otimes u - \eta Du_i + p1) - \rho g = 0$$

$$\rho_t + \nabla \cdot \rho u = 0$$

$$E_t + \nabla \cdot ((E + p)u - k_T \nabla T - k_\omega \nabla \omega) - \eta Du_i : Du_i - \rho u \cdot g = 0$$

- Solve for density $\rho$, ice velocity $u_i$, temperature $T$, and melt fraction $\omega$ using constitutive relations.
  - Simplified constitutive relations can be solved explicitly.
  - Temperature, moisture, and strain-rate dependent rheology $\eta$.
  - High order FEM, typically $Q_3$ momentum & energy
- DAEs solved implicitly after semidiscretizing in space.
- Preconditioning using nested fieldsplit
## The Great Solver Schism: Monolithic or Split?

<table>
<thead>
<tr>
<th>Monolithic</th>
<th>Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct solvers</td>
<td>Physics-split Schwarz (based on relaxation)</td>
</tr>
<tr>
<td>Coupled Schwarz</td>
<td>Physics-split Schur (based on factorization)</td>
</tr>
<tr>
<td>Coupled Neumann-Neumann</td>
<td>approximate commutators SIMPLE, PCD, LSC</td>
</tr>
<tr>
<td>(need unassembled matrices)</td>
<td>segregated smoothers</td>
</tr>
<tr>
<td>Coupled multigrid</td>
<td>Augmented Lagrangian</td>
</tr>
<tr>
<td>X Need to understand local</td>
<td>“parabolization” for stiff waves</td>
</tr>
<tr>
<td>spectral and compatibility</td>
<td></td>
</tr>
<tr>
<td>properties of the coupled system</td>
<td></td>
</tr>
<tr>
<td>X Need to understand global</td>
<td></td>
</tr>
<tr>
<td>coupling strengths</td>
<td></td>
</tr>
</tbody>
</table>

- Preferred data structures depend on which method is used.
- Interplay with geometric multigrid.
Multi-physics coupling in PETSc

- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting
Multi-physics coupling in PETSc

- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting
Multi-physics coupling in PETSc

- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting
Multi-physics coupling in PETSc

- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting
Multi-physics coupling in PETSc

- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting
Splitting for Multiphysics

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} =
\begin{bmatrix}
f \\
g
\end{bmatrix}
\]

- Relaxation: \texttt{-pc_fieldsplit_type [additive,multiplicative,symmetric_multiplicative]}

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\begin{bmatrix}
A & 1 \\
C & D
\end{bmatrix}
\begin{bmatrix}
A & B \\
1 & 1
\end{bmatrix}
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}^{-1}
\]

- Gauss-Seidel inspired, works when fields are loosely coupled

- Factorization: \texttt{-pc_fieldsplit_type schur}

\[
\begin{bmatrix}
A & B \\
S & C
\end{bmatrix}
\begin{bmatrix}
1 & 1 \\
C A^{-1} & 1
\end{bmatrix}
S = D - C A^{-1} B
\]

- robust (exact factorization), can often drop lower block
- how to precondition \(S\) which is usually dense?
  - interpret as differential operators, use approximate commutators
Work in Split Local space, matrix data structures reside in any space.
Assembly code is independent of matrix format
- No-copy fieldsplit:
  -pack_dm_mat_type nest -pc_type fieldsplit
- Coupled direct solve:
  -pack_dm_mat_type aij -pc_type lu -pc_factor_mat_solver_package mumps
The common block preconditioners for Stokes require only options:

- `pc_type fieldsplit`
- `pc_field_split_type`
- `fieldsplit_0_pc_type ml`
- `fieldsplit_0_ksp_type preonly`
Stokes Example

The common block preconditioners for Stokes require only options:

```
- pc_type fieldsplit
- pc_field_split_type additive
- fieldsplit_0_pc_type ml
- fieldsplit_0_ksp_type preonly
- fieldsplit_1_pc_type jacobi
- fieldsplit_1_ksp_type preonly
```

\[
\begin{pmatrix}
\hat{A} & 0 \\
0 & I
\end{pmatrix}
\]

Stokes Example

The common block preconditioners for Stokes require only options:

- `pc_type fieldsplit`
- `pc_field_split_type` multiplicative
- `fieldsplit_0_pc_type ml`
- `fieldsplit_0_ksp_type preonly`
- `fieldsplit_1_pc_type jacobi`
- `fieldsplit_1_ksp_type preonly`

Stokes Example

The common block preconditioners for Stokes require only options:

- `pc_type fieldsplit`
- `pc_field_split_type schur`
- `fieldsplit_0_pc_type ml`
- `fieldsplit_0_ksp_type preonly`
- `fieldsplit_1_pc_type none`
- `fieldsplit_1_ksp_type minres`
- `pc_fieldsplit_schur_factorization_type diag`

\[
\begin{pmatrix}
\hat{A} & 0 \\
0 & -\hat{S}
\end{pmatrix}
\]


Olshanskii, Peters, and Reusken *Uniform preconditioners for a parameter dependent saddle point problem with application to generalized Stokes interface equations*, 2006.
Stokes Example

The common block preconditioners for Stokes require only options:

- `pc_type fieldsplit`
- `pc_field_split_type schur`
- `fieldsplit_0_pc_type ml`
- `fieldsplit_0_ksp_type preonly`
- `fieldsplit_1_pc_type none`
- `fieldsplit_1_ksp_type minres`
- `pc_fieldsplit_schur_factorization_type lower`

\[
\begin{pmatrix}
\hat{A} & 0 \\
B^T & \hat{S}
\end{pmatrix}
\]

The common block preconditioners for Stokes require only options:

- `pc_type fieldsplit`
- `pc_field_split_type schur`
- `fieldsplit_0_pc_type ml`
- `fieldsplit_0_ksp_type preonly`
- `fieldsplit_1_pc_type none`
- `fieldsplit_1_ksp_type minres`
- `pc_fieldsplit_schur_factorization_type upper`

Stokes Example

The common block preconditioners for Stokes require only options:

- `pc_type fieldsplit`
- `pc_field_split_type schur`
- `fieldsplit_0_pc_type ml`
- `fieldsplit_0_ksp_type preonly`
- `fieldsplit_1_pc_type lsc`
- `fieldsplit_1_ksp_type minres`
- `pc_field_split_schur_factorization_type full`


The common block preconditioners for Stokes require only options:

- `pc_type fieldsplit`
- `pc_field_split_type schur`
- `pc_fieldsplit_schur_factorization_type`

\[
\begin{bmatrix}
I & 0 \\
B^T A^{-1} & I
\end{bmatrix}
\begin{bmatrix}
\hat{A} & 0 \\
0 & \hat{S}
\end{bmatrix}
\begin{bmatrix}
I & A^{-1} B \\
0 & I
\end{bmatrix}
\]
Coupled MG for Stokes, split smoothers

\[ J = \begin{pmatrix} A & B^T \\ B & C \end{pmatrix} \]

\[ P_{\text{smooth}} = \begin{pmatrix} A_{\text{SOR}} & 0 \\ B & M \end{pmatrix} \]

-\text{pc\_type mg -pc\_mg\_levels 5 -pc\_mg\_galerkin}
-\text{mg\_levels\_pc\_type fieldsplit}
-\text{mg\_levels\_pc\_fieldsplit\_block\_size 3}
-\text{mg\_levels\_pc\_fieldsplit\_0\_fields 0,1}
-\text{mg\_levels\_pc\_fieldsplit\_1\_fields 2}
-\text{mg\_levels\_fieldsplit\_0\_pc\_type sor}
Nonlinear solvers in PETSc SNES

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS, TR</td>
<td>Newton-type with line search and trust region</td>
</tr>
<tr>
<td>NRichardson</td>
<td>Nonlinear Richardson, usually preconditioned</td>
</tr>
<tr>
<td>VIRS, VIRSAUG, and VISS</td>
<td>Reduced space and semi-smooth methods for variational inequalities</td>
</tr>
<tr>
<td>QN</td>
<td>Quasi-Newton methods like BFGS</td>
</tr>
<tr>
<td>NGMRES</td>
<td>Nonlinear GMRES</td>
</tr>
<tr>
<td>NCG</td>
<td>Nonlinear Conjugate Gradients</td>
</tr>
<tr>
<td>SORQN</td>
<td>Multiplicative Schwarz quasi-Newton</td>
</tr>
<tr>
<td>GS</td>
<td>Nonlinear Gauss-Seidel/multiplicative Schwarz sweeps</td>
</tr>
<tr>
<td>FAS</td>
<td>Full approximation scheme (nonlinear multigrid)</td>
</tr>
<tr>
<td>MS</td>
<td>Multi-stage smoothers, often used with FAS for hyperbolic problems</td>
</tr>
<tr>
<td>Shell</td>
<td>Your method, often used as a (nonlinear) preconditioner</td>
</tr>
</tbody>
</table>
Quasi-Newton revisited: ameliorating setup costs

- **Newton-Krylov with analytic Jacobian**

<table>
<thead>
<tr>
<th>Lag</th>
<th>FunctionEval</th>
<th>JacobianEval</th>
<th>PCSetUp</th>
<th>PCApply</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bt</td>
<td>12</td>
<td>8</td>
<td>8</td>
<td>31</td>
</tr>
<tr>
<td>1 cp</td>
<td>31</td>
<td>6</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>2 bt</td>
<td>— diverged</td>
<td>— diverged</td>
<td>— diverged</td>
<td>— diverged</td>
</tr>
<tr>
<td>2 cp</td>
<td>41</td>
<td>4</td>
<td>4</td>
<td>35</td>
</tr>
<tr>
<td>3 cp</td>
<td>50</td>
<td>4</td>
<td>4</td>
<td>44</td>
</tr>
</tbody>
</table>

- **Jacobian-free Newton-Krylov with lagged preconditioner**

<table>
<thead>
<tr>
<th>Lag</th>
<th>FunctionEval</th>
<th>JacobianEval</th>
<th>PCSetUp</th>
<th>PCApply</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bt</td>
<td>23</td>
<td>11</td>
<td>11</td>
<td>31</td>
</tr>
<tr>
<td>2 bt</td>
<td>48</td>
<td>4</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>3 bt</td>
<td>64</td>
<td>3</td>
<td>3</td>
<td>52</td>
</tr>
<tr>
<td>4 bt</td>
<td>87</td>
<td>3</td>
<td>3</td>
<td>75</td>
</tr>
</tbody>
</table>

- **Limited-memory Quasi-Newton/BFGS with lagged solve for $H_0$**

<table>
<thead>
<tr>
<th>Restart</th>
<th>$H_0$</th>
<th>FunctionEval</th>
<th>JacobianEval</th>
<th>PCSetUp</th>
<th>PCApply</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 cp</td>
<td>$10^{-5}$</td>
<td>17</td>
<td>4</td>
<td>4</td>
<td>35</td>
</tr>
<tr>
<td>1 cp</td>
<td>preonly</td>
<td>21</td>
<td>5</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>3 cp</td>
<td>$10^{-5}$</td>
<td>21</td>
<td>3</td>
<td>3</td>
<td>43</td>
</tr>
<tr>
<td>3 cp</td>
<td>preonly</td>
<td>23</td>
<td>3</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>6 cp</td>
<td>$10^{-5}$</td>
<td>29</td>
<td>2</td>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>6 cp</td>
<td>preonly</td>
<td>29</td>
<td>2</td>
<td>2</td>
<td>14</td>
</tr>
</tbody>
</table>
Relative effect of the blocks

\[ J = \begin{pmatrix}
J_{uu} & J_{up} & J_{uE} \\
J_{pu} & 0 & 0 \\
J_{Eu} & J_{Ep} & J_{EE}
\end{pmatrix}. \]

- \( J_{uu} \): Viscous/momentum terms, nearly symmetric, variable coefficients, anisotropy from Newton.
- \( J_{up} \): Weak pressure gradient, viscosity dependence on pressure (small), gravitational contribution (pressure-induced density variation). Large, nearly balanced by gravitational forcing.
- \( J_{uE} \): Viscous dependence on energy, very nonlinear, not very large.
- \( J_{pu} \): Divergence (mass conservation), nearly equal to \( J_{up}^T \).
- \( J_{Eu} \): Sensitivity of energy on momentum, mostly advective transport. Large in boundary layers with large thermal/moisture gradients.
- \( J_{Ep} \): Thermal/moisture diffusion due to pressure-melting, \( u \cdot \nabla \).
- \( J_{EE} \): Advection-diffusion for energy, very nonlinear at small regularization. Advection-dominated except in boundary layers and stagnant ice, often balanced in vertical.
How much nesting?

\[ P_1 = \begin{pmatrix} J_{uu} & J_{up} & J_{uE} \\ 0 & B_{pp} & 0 \\ 0 & 0 & J_{EE} \end{pmatrix} \]

- \( B_{pp} \) is a mass matrix in the pressure space weighted by inverse of kinematic viscosity.
- Elman, Mihajlović, Wathen, JCP 2011 for non-dimensional isoviscous Boussinesq.
- Works well for non-dimensional problems on the cube, not for realistic parameters.
- Low-order preconditioning full-accuracy unassembled high order operator.
- Build these on command line with PETSc PCFieldSplit.

\[ P = \begin{bmatrix} (J_{uu} & J_{up}) \\ (J_{pu} & 0) \\ (J_{Eu} & J_{Ep}) & J_{EE} \end{bmatrix} \]

- Inexact inner solve using upper-triangular with \( B_{pp} \) for Schur.
- Another level of nesting.
- GCR tolerant of inexact inner solves.
- Outer converges in 1 or 2 iterations.
Performance of assembled versus unassembled

- High order Jacobian stored unassembled using coefficients at quadrature points, can use local AD
- Choose approximation order at run-time, independent for each field
- Precondition high order using assembled lowest order method
- Implementation $> 70\%$ of FPU peak, SpMV bandwidth wall $< 4\%$
## Hardware Arithmetic Intensity

<table>
<thead>
<tr>
<th>Operation</th>
<th>Arithmetic Intensity (flops/B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse matrix-vector product</td>
<td>1/6</td>
</tr>
<tr>
<td>Dense matrix-vector product</td>
<td>1/4</td>
</tr>
<tr>
<td>Unassembled matrix-vector product</td>
<td>≈ 8</td>
</tr>
<tr>
<td>High-order residual evaluation</td>
<td>&gt; 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processor</th>
<th>BW (GB/s)</th>
<th>Peak (GF/s)</th>
<th>Balanced AI (F/B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandy Bridge 6-core</td>
<td>21*</td>
<td>150</td>
<td>7.2</td>
</tr>
<tr>
<td>Magny Cours 16-core</td>
<td>42*</td>
<td>281</td>
<td>6.7</td>
</tr>
<tr>
<td>Blue Gene/Q node</td>
<td>43</td>
<td>205</td>
<td>4.8</td>
</tr>
<tr>
<td>Tesla M2050</td>
<td>144</td>
<td>515</td>
<td>3.6</td>
</tr>
</tbody>
</table>
One level of $p$-multigrid

- Want to skip assembly on finest level (for better throughput)
- High order operators lack $h$-ellipticity
  - Necessary and sufficient condition for existence of pointwise smoother
- Use embedded low-order operator as smoother
  - Rescaled to recover a consistent inner product
  - Does not destroy symmetry for point-block Jacobi
- Polynomial smoothers
  - Target upper part of PBJacobi-preconditioned spectrum
  - Efficient GPU implementation
- Reordered incomplete factorization to couple “columns”
- Operator-dependent interpolation is more delicate
- Strict semi-coarsening requires semi-structured grid
Construction of conservative nodal normals

\[ n^i = \int_{\Gamma} \phi^i n \]

- Exact conservation even with rough surfaces
- Definition is robust in 2D and for first-order elements in 3D
- \( \int_{\Gamma} \phi^i = 0 \) for corner basis function of undeformed \( P_2 \) triangle
- May be negative for sufficiently deformed quadrilaterals
- Mesh motion should use normals from CAD model
  - Difference between CAD normal and conservative normal introduces correction term to conserve mass within the mesh
  - Anomalous velocities if disagreement is large
    (fast moving mesh, rough surface)
- Normal field not as smooth/accurate as desirable
  (and achievable with non-conservative normals)
  - Mostly problematic for surface tension
Need for well-balancing

(Behr, *On the application of slip boundary condition on curved surfaces*, 2004)
“No” boundary condition

Integration by parts produces

\[ \int_{\Gamma} v \cdot T \sigma \cdot n, \quad \sigma = \eta Du - p 1, \quad T = 1 - n \otimes n \]

Continuous weak form requires either

- Dirichlet: \( u|_{\Gamma} = f \implies v|_{\Gamma} = 0 \)
- Neumann/Robin: \( \sigma \cdot n|_{\Gamma} = g(u,p) \)

Discrete problem allows integration of \( \sigma \cdot n \) “as is”

- Extends validity of equations to include \( \Gamma \)
- Not valid for continuum equations

Introduced by Papanastasiou, Malamataris, and Ellwood, 1992 for Navier-Stokes outflow boundaries

Griffiths, *The ‘no boundary condition’ outflow boundary condition*, 1997

- Proves \( L^\infty \) order of accuracy \( \mathcal{O}((h + 1/Pe)^{p+1}) \) for Galerkin finite elements of order \( p \) (linear advection-diffusion)
- Demonstrates equivalence with collocation at Radau points in outflow element

Used in slip boundary conditions by Behr 2004
Outlook

- Unintrusive composition of multigrid and block preconditioning
- We can build many preconditioners from the literature on the command line
- User code does not depend on matrix format, preconditioning method, nonlinear solution method, time integration method (implicit or IMEX), or size of coupled system (except for driver).
- Similar infrastructure extends to nonlinear methods
- Preliminary implementations on GPU

In development

- Distributive relaxation and Vanka smoothers
- Improved operator-dependent semi-geometric multigrid
- Automated support for mixing analysis/UQ into levels
- IMEX time stepping for geometry evolution
- Special basis functions for corners