The parallel solution of ice flow problems: Block preconditioning strategy for variable-viscosity Stokes equations

Mika Malinen

CSC – IT Center for Science Ltd.
P.O. Box 405, FI-02101 Espoo, Finland

Elmer/Ice workshop
Nov 5, 2013

0. Contents

1. Motivation and background

- **The theme of this presentation**: Efficient ways to obtain scalar coefficients (referred to as degrees of freedom) $\mathbf{u}_j = [u_j \; v_j \; w_j]^T$ and p_j such that the associated finite element (FE) expansions of velocity and pressure fields

  $$\mathbf{u}_h : \Omega \to \mathbb{R}^3 \text{ and } p_h : \Omega \to \mathbb{R},$$

  which are expressed as

  $$\mathbf{u}_h(\mathbf{x}) = \sum_{j=1}^{N_u} \mathbf{u}_j \phi_j(\mathbf{x}), \quad p_h(\mathbf{x}) = \sum_{k=1}^{N_p} p_j \psi_k(\mathbf{x}),$$

  solve a FE version of variable-viscosity Stokes equations posed on the ice region Ω.

- A bottom line is that this is basically a linear algebra problem

- Special techniques are required to obtain efficiency!

The scope of ideas

- The special methods to be described are suitable for obtaining discrete approximations to the solution characterized by

$$\begin{aligned} -\operatorname{div}[2\mu(\mathbf{D})\mathbf{D}(\mathbf{u})] + \nabla p &= \rho \mathbf{g}, \\ -\operatorname{div} \mathbf{u} &= 0 \end{aligned} \tag{1}$$

  subject to the boundary conditions

$$\begin{aligned} \mathbf{u} &= \hat{\mathbf{u}} \quad \text{on } \Gamma_D, \\ 2\mu(\mathbf{D})\mathbf{D}(\mathbf{u})\mathbf{n} - p\mathbf{n} &= \hat{\mathbf{s}} \quad \text{on } \Gamma_N, \\ \mathbf{u}\cdot\mathbf{n} = 0 \quad \text{and} \quad \mathbf{n} \times [2\mu(\mathbf{D})\mathbf{D}(\mathbf{u})\mathbf{n}] \times \mathbf{n} &= -\beta \mathbf{n} \times \mathbf{u} \times \mathbf{n} \quad \text{on } \Gamma_S. \end{aligned} \tag{2}$$

- Here $\hat{\mathbf{u}}$ is the specified velocity, $\hat{\mathbf{s}}$ is the specified traction and also $\beta > 0$ is given as initial data.

Obtaining the linear algebra problem associated with the FE approximation

- The associated FE problem is of the form: Find $(\mathbf{u}_h, p_h) \in \mathcal{U}_h$ such that

$$\int_\Omega 2\mu(\mathbf{D}(\mathbf{u}_h))\mathbf{D}(\mathbf{u}_h) \cdot \mathbf{D}(\mathbf{v}_h)\, d\Omega + \int_{\Gamma_S} \beta(\mathbf{n} \times \mathbf{u}_h \times \mathbf{n}) \cdot \mathbf{v}_h\, dS - \int_\Omega p_h \nabla \cdot \mathbf{v}_h\, d\Omega$$

$$= \int_\Omega \rho \mathbf{g} \cdot \mathbf{v}_h\, d\Omega + \int_{\Gamma_N} \hat{\mathbf{s}} \cdot \mathbf{v}_h\, dS$$

$$-\int_\Omega \nabla \cdot \mathbf{u}_h q_h\, d\Omega = 0$$

  for any $(\mathbf{v}_h, q_h) \in \mathcal{V}_h$.

- This gives a nonlinear system of algebraic equations (with $\mathbf{U}$ and $\mathbf{P}$ containing the velocity and pressure degrees of freedom)

$$\begin{bmatrix} \mathbf{A}(\mathbf{U}) & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}. \tag{3}$$

Obtaining the linear algebra problem...

- The final step is linearization to obtain a sequence of linear algebra problems for solving the nonlinear problem iteratively.

- The simplest strategy is to use the lagged value approximation of the viscosity, so that at the step $k+1$ of the nonlinear iteration we need to solve

$$\begin{bmatrix} \mathbf{A}_k & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{k+1} \\ \mathbf{P}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}, \tag{4}$$

  with the entries of $\mathbf{A}_k$ arising from the computation of integrals

$$\int_\Omega 2\mu(\mathbf{D}(\mathbf{u}_h^k))\mathbf{D}(\mathbf{u}_h^{k+1}) \cdot \mathbf{D}(\mathbf{v}_h)\, d\Omega.$$

- The Newton linearization is another option (some details to follow), but also then the system has the natural (2×2) block structure

Solution strategies for the linear algebra problem

- Applying direct solution methods to

$$\begin{bmatrix} \mathbf{A}_k & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{k+1} \\ \mathbf{P}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix} \tag{5}$$

  is not an optimal strategy.

- Using iterative linear solvers, in combination with an effective preconditioner, is the best approach.

- **A key issue:** If the linear system (5) is abbreviated as $\mathbf{K}\mathbf{x} = \mathbf{b}$, we need an efficient preconditioner P which makes solving

$$\mathbf{K}\mathrm{P}^{-1}\mathbf{z} = \mathbf{b}, \text{ with } \mathrm{P}\mathbf{z} = \mathbf{x},$$

  quick and which is also amenable for a parallel implementation.

On flow solvers of Elmer

- Elmer/Ice simulations have traditionally been based on the general Navier–Stokes (NS) flow solver of Elmer.

- The general preconditioning strategies available in connection with the standard NS solver are not best suited for handling saddle point problems of the type we need to handle here

- To circumvent this performance bottleneck, a separate Stokes flow solver has been written in order to enable the use of an alternate preconditioning strategy which is effective.

- The first version of this alternate solver was made public in January, 2012.

- The solver code is contained in the file `../fem/src/modules/ParStokes.src`

- Utilizing the solver in parallel has been in mind from the very beginning (the file name comes from Parallel $Stokes$ solver)

2. The block preconditioning strategy

- **An alternate view on preconditioning:** Instead of handling the preconditioned system $\mathbf{K}\mathsf{P}^{-1}\mathbf{z} = \mathbf{b}$ to produce iterates of $\mathbf{z} = \mathsf{P}\mathbf{x}$, we consider generating iterates $\mathbf{x}^{(k)}$ (via applying the generalized conjugate residual, GCR, method) that minimize the residual 2-norm

$$||\mathbf{b} - \mathbf{K}\mathbf{x}^{(k)}||$$

over the space

$$\mathcal{X}_k = \mathbf{x}^{(0)} + \mathrm{span}\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(k)}\}$$

- In this setting, we define the preconditioner to be the operator which, given the previous iterate, produces the new search direction $\mathbf{s}^{(k)}$.

- The standard choice is to solve the residual correction system

$$\mathsf{P}\mathbf{s}^{(k+1)} = \mathbf{b} - \mathbf{K}\mathbf{x}^{(k)}$$

with $\mathsf{P} \approx \mathbf{K}$, but approximating the original coefficient matrix is not a necessity.

- For systems of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{V} \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix},$$

a common choice for the preconditioner is

$$\mathsf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}. \tag{6}$$

- An optimal $\mathbf{Q}$ corresponds to the Schur complement matrix $\mathbf{C} - \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T$.

- In the case of discrete Stokes equations a usual way is to select

$$\mathbf{Q} = \varepsilon^{-1}\mathbf{M}, \quad \mathbf{M} = [M_{ij}],\ M_{ij} = \int\limits_{\Omega} \psi_j \psi_j \, d\Omega \tag{7}$$

with ε the viscosity from the previous nonlinear iteration and $\mathbf{M}$ is the pressure mass matrix.

- It is known that this choice of the preconditioner is optimal when the viscosity is constant.

- However, no sharp theory exists for predicting the preconditioner performance when variable-viscosity flows are solved.

- Anyhow, we have experimental evidence that the scaled mass matrix approximation of the Schur complement may work surprisingly well.

- More advanced approximations are possible but not motivated in the light of experimental results.

3. The overview of the `ParStokes` module

- The block preconditioned GCR iteration is in-built into the `ParStokes` module.
- Otherwise the alternate Stokes solver contained in the `ParStokes` module basically mimics the standard NS solver of Elmer
- However, it does not provide all features available in the standard NS solver.
- Stable finite element approximation of the weak formulation necessitates using different approximation spaces for the velocity and pressure.
- In the `ParStokes` solver, the velocity approximation is augmented by elementwise bubble functions to obtain stability.
- Using the Newton iteration is also an option.

Applying the block preconditioner

- At each block preconditioned GCR iteration step we need to solve approximately the system

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{Q} \end{bmatrix} \begin{bmatrix} \delta\mathbf{V}^{(k+1)} \\ \delta\mathbf{P}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{F} - \mathbf{A}\mathbf{V}^{(k)} - \mathbf{B}^T\mathbf{P}^{(k)} \\ \mathbf{G} - \mathbf{B}\mathbf{V}^{(k)} - \mathbf{C}\mathbf{P}^{(k)} \end{bmatrix}. \tag{8}$$

- The user must thus define methods for solving subproblems of the type

$$\mathbf{Q}\delta\mathbf{P}^{(k+1)} = \mathbf{R}_P^{(k)} \quad \text{and} \quad \mathbf{A}\delta\mathbf{V}^{(k+1)} = \mathbf{R}_A^{(k)} - \mathbf{B}^T\delta\mathbf{V}^{(k+1)}$$

or, in practice, their preconditioned versions

$$(\mathbf{Q}\mathsf{P}_Q^{-1})\delta\hat{\mathbf{P}}^{(k+1)} = \mathbf{R}_P^{(k)} \quad \text{and} \quad (\mathbf{A}\mathsf{P}_A^{-1})\delta\hat{\mathbf{V}}^{(k+1)} = \mathbf{R}_A^{(k)} - \mathbf{B}^T\delta\mathbf{V}^{(k+1)}$$

- Selecting the preconditioner P_Q is an easy task. The choice of P_A is more difficult and critical to the performance.

The requirements for an ideal block preconditioning strategy

It is hoped that

- The block preconditioner P is such that the outer GCR iteration counts do not severely depend on the problem size and variations of essential model parameters
- The subsidiary computations corresponding to the application of the preconditioner can be done efficiently (in an ideal case by exploiting optimal complexity solvers).

If the block preconditioner P is robust in the above sense, parallel scalability (weak) depends primarily on the scalability of the subsidiary computations. **Ways to improve the efficiency of the subsidiary computations:**

- Interfacing with external linear algebra libraries (Hypre, Trilinos)
- Inaccurate solves of the subsidiary problems: a high accuracy is not needed

Bubble stabilization utilized in the `ParStokes` solver

- The velocity FE space is taken to be $U_h = S_h + B_h$, with S_h the standard space based on the first-order polynomials $P_1(\hat{K})$ and B_h the bubble space

$$B_h = \{v_h \mid v_{h|K} = (\hat{v} \circ \mathbf{f}_K^{-1})(\mathbf{x}), \hat{v} \in P_r(\hat{K}) \text{ and } v_{h|\partial K} = 0 \ \forall K = \mathbf{f}_K(\hat{K})\}.$$

- Some possible choices for the order r of the space $P_r(\hat{K})$ in the definition of the bubble space (1 = not recommended, 2 = typical choice)

element	r	#element bubbles/dof
brick^1	6	1
brick^2	7	4
tetrahedron^1	4	1
tetrahedron^2	5	4
wedge^1	5	1
wedge^2	6	4

- That is, in the sif file (in 3-D) use the element definition `Element = p:1 b:4`

Bubble stabilization continued

- We eliminate additional bubble degrees of freedom via the static condensation so that the unknowns to be solved correspond to standard nodal values.

- The static condensation $\Rightarrow$ the (2,2)-block of the system matrix is nontrivial, i.e. $\mathbf{C} \neq \mathbf{0}$.

- Assembly is expensive due to integrating bubble contributions accurately

- The "degree of stability" may be adjusted in a flexible manner by increasing the polynomial order of bubble functions.

Newton linearization for Glen's flow law $\mu(\mathbf{D}) = 1/2A^{-k}[I_2(\mathbf{D})]^{(k-1)/2}$

- Define $\mathbf{g}(\mathbf{D}) = [I_2(\mathbf{D})]^{(k-1)/2}\mathbf{D}$, with $I_2(\mathbf{D}) = 1/2(\mathbf{D} \cdot \mathbf{D})$, to obtain the derivative

$$\mathsf{D}\mathbf{g}(\mathbf{D})[\mathbf{U}] = \mu(\mathbf{D})\mathbf{U} + \{\frac{k-1}{2}[I_2(\mathbf{D})]^{(k-3)/2}\mathbf{D} \cdot \mathbf{U}\}\mathbf{D}$$

  and the linearization

$$\mathbf{g}(\mathbf{D}_{k+1}) = \mathbf{g}(\mathbf{D}_k) + \mathsf{D}\mathbf{g}(\mathbf{D}_k)[\mathbf{D}_{k+1} - \mathbf{D}_k].$$

- This leads to the Newton approximation

$$\begin{aligned}\mathbf{g}(\mathbf{D}_{k+1}) = & -\frac{k-1}{2}[I_2(\mathbf{D}_k)]^{(k-3)/2}(\mathbf{D}_k \cdot \mathbf{D}_k)\mathbf{D}_k + \\ & \frac{k-1}{2}[I_2(\mathbf{D}_k)]^{(k-3)/2}(\mathbf{D}_k \cdot \mathbf{D}_{k+1})\mathbf{D}_k + [I_2(\mathbf{D}_k)]^{(k-1)/2}\mathbf{D}_{k+1}.\end{aligned}$$

- This alters the definition of the $\mathbf{A}$-block and the RHS vector.

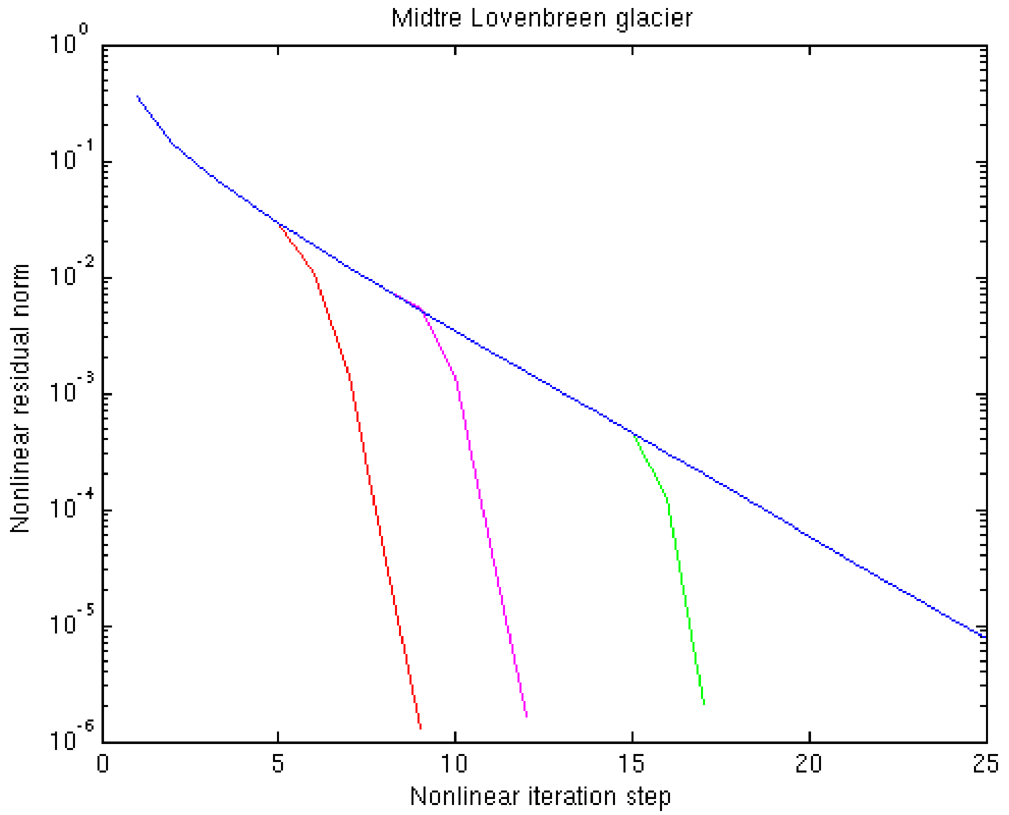


Figure 1: The nonlinear residual norm for different linearization strategies to solve a Midtre Lovénbreen flow ($A = 10^{-16} Pa^{-3} a^{-1}$): the blue curve corresponds to Picard linearization, while the red, magenta and green curves correspond to hybrid linearization based on threshold values $\delta_{NL} = 10^{-1}/2, 10^{-2}, 10^{-3}$ for switching from the Picard linearization to the Newton scheme.

4. Creating the solver input file and keywords

With the block preconditioned solver iterations are done at three levels:

- **Nonlinear iteration** to satisfy

$$\left\| \begin{bmatrix} \mathbf{F}_k \\ \mathbf{G}_k \end{bmatrix} - \begin{bmatrix} \mathbf{A}(\mathbf{U}_k) & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{U}_k \\ \mathbf{P}_k \end{bmatrix} \right\| = \varepsilon_N \left\| \begin{bmatrix} \mathbf{F}_k \\ \mathbf{G}_k \end{bmatrix} \right\|$$

- **Linear solver iteration** (inside the nonlinear iteration) to satisfy

$$\left\| \begin{bmatrix} \mathbf{F}_k \\ \mathbf{G}_k \end{bmatrix} - \begin{bmatrix} \mathbf{A}(\mathbf{U}_k) & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{k+1}^{(j)} \\ \mathbf{P}_{k+1}^{(j)} \end{bmatrix} \right\| = \varepsilon_L \left\| \begin{bmatrix} \mathbf{F}_k \\ \mathbf{G}_k \end{bmatrix} \right\|$$

- **Linear solver iterations for the preconditioning subproblems** to satisfy

$$\|\mathbf{R}_P^{(k)} - \mathbf{Q}\delta\mathbf{P}^{(k+1)}\| = \varepsilon_Q \|\mathbf{R}_P^{(k)}\| \quad \text{and} \quad \|\hat{\mathbf{R}}_A^{(k)} - \mathbf{A}\delta\mathbf{V}^{(k+1)}\| = \varepsilon_A \|\hat{\mathbf{R}}_A^{(k)}\|,$$

with $\hat{\mathbf{R}}_A^{(k)} = \mathbf{R}_A^{(k)} - \mathbf{B}^T\delta\mathbf{V}^{(k+1)}$.

Three solver sections (at least) are needed in the Elmer solver input file:

```
Solver 1
 Equation = "Pressure Preconditioning"
 !  The following commands control the solution of pressure
 !  preconditioning problems
 ...
End

Solver 2
 Equation = "Velocity Preconditioning"
 !  The following commands control the solution of velocity
 !  preconditioning problems
 ...
End

Solver 3
 Equation = "Stokes equations"
 !  Commands for solving the Stokes system
 ...
End
```

```
Solver 3
 Equation = "Stokes equations"
 Procedure = "ParStokes" "StokesSolver"
 Variable = FlowVar
 Variable DOFs = 4
 Element = "p:1 b:4"
 Bubbles in Global System = Logical False
 Nonlinear System Convergence Tolerance = 1.0e-5 !  equals ε_N
 Nonlinear System Max Iterations = 50
 Nonlinear System Newton After Tolerance = 1.0e-3
 Linear System Row Equilibration = Logical True
 Linear System Solver = "Iterative"
 Linear System Iterative Method = "GCR"
 Linear System Max Iterations = 200
 Linear System Convergence Tolerance = 1.0e-7 !  equals ε_L
 Linear System GCR Restart = Integer 50
 Block Preconditioning = Logical True
End
```

The solver section for the Stokes equations: Additional details

- The command `Bubbles in Global System = Logical False` is used to eliminate the bubble degrees of freedom before the linear solve.

- The command `Block Preconditioning = Logical True` activates the use of block preconditioning.

- Giving `Linear System Row Equilibration = Logical True` is recommended. The row equilibration of the system matrix is then done, so that the condition number of the scaled system matrix is minimal (with respect to the ∞-norm).

- The `Linear System GCR Restart` command defines the iteration count m after which the GCR iteration is restarted (to constrain the dimension of the search space $\mathcal{X}_k$). The linear solver is then GCR(m).

- Compile with `elmerf90 ParStokes.f90 -o ParStokes.so`

```
Solver 1
 Equation = "Pressure Preconditioning"
 Procedure = "PressurePrecond" "PressurePrecond"
 Exec Solver = "before simulation"
 Variable = -dofs 1 "P"
 Variable Output = False
 Element = "p:1"
 Linear System Solver = iterative
 Linear System Iterative Method = BiCGStabL
 Linear System Max Iterations = 1000
 Linear System Convergence Tolerance = 1.0e-6 !  equals ε_Q
 Linear System Preconditioning = None
 Skip Compute Nonlinear Change = Logical True
 Back Rotate N-T Solution = Logical False
End
```

The solver section for pressure preconditioning: Additional details

- The equation name must be `"Pressure Preconditioning"`
- If the normal-tangential BCs are given, `Back Rotate N-T Solution = Logical False` must be given (explanation for this is technical).
- This solver section is used to generate the matrix structure for the pressure preconditioning before attempting to solve the Stokes system, so giving `Exec Solver = "before simulation"` is enough.
- The module file `PressurePrecond.so` is obtained by compiling a dummy solver:
  - ▷ Get `PressurePrecond.src` from
    `../fem/src/modules/PressurePrecond.src`
  - ▷ Rename it as `PressurePrecond.f90`
  - ▷ Compile with `elmerf90 PressurePrecond.f90 -o PressurePrecond.so`
- The above values of the keywords should provide a good starting point.

```
Solver 2
 Equation = "Velocity Preconditioning"
 Procedure = "VelocityPrecond" "VelocityPrecond"
 Exec Solver = "before simulation"
 Variable = -dofs 3 "V"
 Variable Output = False
 Element = "p:1"
 Linear System Scaling = Logical True
 Linear System Row Equilibration = Logical True
 Linear System Solver = Iterative
 Linear System Iterative Method = BiCGStabL
 Linear System Max Iterations = 1000
 Linear System Convergence Tolerance = 1.0e-3 !  equals ε_A
 Linear System Preconditioning = ILU0
 Skip Compute Nonlinear Change = Logical True
 Back Rotate N-T Solution = Logical False
End
```

The solver section for velocity preconditioning: Additional details

- The equation name must be `"Velocity Preconditioning"`
- If the normal-tangential BCs are given, `Back Rotate N-T Solution = Logical False` must again be given.
- This solver section is used to generate the matrix structure for the velocity preconditioning before attempting to solve the Stokes system, so giving `Exec Solver = "before simulation"` is enough.
- The module file `VelocityPrecond.so` is obtained by compiling a dummy solver:
  - ▷ Get `VelocityPrecond.src` from
    `../fem/src/modules/VelocityPrecond.src`
  - ▷ Rename it as `VelocityPrecond.f90`
  - ▷ Compile with `elmerf90 VelocityPrecond.f90 -o VelocityPrecond.so`
- Some of the above keyword commands starting with `Linear System` may need to be changed to obtain a scalable solver.

Giving boundary conditions for the velocity preconditioner

- If the Dirichlet boundary condition of the type $\mathbf{u} = \hat{\mathbf{u}}$ is specified, the variable of the velocity preconditioning equation $\delta\mathbf{V}$ (the Elmer variable `V` above) must also be constrained similarly by using the homogeneous Dirichlet condition:

```
Boundary Condition 1
  ...
 FlowVar 1 = Real 0.0
 FlowVar 2 = Real 0.0
 FlowVar 3 = Real 0.0
 V 1 = Real 0.0
 V 2 = Real 0.0
 V 3 = Real 0.0
End
```

- Usually no need to specify boundary conditions for the pressure preconditioning variable.

Adaptive convergence tolerances

- If the error corresponding to the nonlinear system is large, using a high accuracy to solve the linearized system which defines the next nonlinear iterate is not necessary.

- The stopping tolerance for the linear system can be adjusted adaptively:

```
Solver 3
 ...
 Linear System Convergence Tolerance = 1.0e-7 !  equals ε_L
 Linear System Adaptive Tolerance = Logical True
 Linear System Relative Tolerance = Real 1e-2 !  to define η_R
 Linear System Base Tolerance = Real 1.0e-3 !  to define ε_B
 ...
```

- At the step $k+1$ of the nonlinear iteration this replaces the fixed tolerance ε_L by $\varepsilon_L^{(k+1)} = \eta_R \cdot \eta_N^{(k)}$, with $\eta_N^{(k)}$ the previous nonlinear error.

- However, $\varepsilon_L^{(k+1)}$ is never made larger than ε_B or smaller than the given ε_L.

5. Monitoring convergence and some practical tips

- The nonlinear iteration outputs $\|\mathbf{b}_k - \mathbf{K}(\mathbf{x}_k)\mathbf{x}_k\|_2/\|\mathbf{b}_k\|_2$ (available at the beginning of the step $k+1$)

- The preconditioned GCR outputs $\|\mathbf{b}_k - \mathbf{K}(\mathbf{x}_k)\mathbf{x}_{k+1}^{(j)}\|_2/\|\mathbf{b}_k\|_2$

```
...
StokesSolver:  Residual for nonlinear iterate 2 7.750E-02
...
Outer Iteration:  GCR residual for iterate 0 7.750E-02
...
Outer Iteration:  GCR residual for iterate 1 5.171E-02
...
Outer Iteration:  GCR residual for iterate 6 5.113E-04
...
StokesSolver:  Residual for nonlinear iterate 3 4.363E-02
```

- Having high element aspect ratios, etc., may affect the performance and make solving the subsidiary problems less straightforward

- If you have a mesh ready, it may be a good practice to start by solving the Stokes equations with a constant viscosity on it in order to identify a good solver for the $\mathbf{A}$-block.

- The block preconditioner should be ideal for the constant viscosity case.

- If the convergence of the outer GCR iteration is not satisfactory then, the mesh may contain pathologies. Consider improving mesh, as things are not likely to improve when viscosity variations are taken into account.

6. Examples

Key issues in performance

- A thin domain $\Rightarrow$ high element aspect ratios $\Rightarrow$ weakened finite element stability may have an effect on the effectiveness of the preconditioner
- The robustness of the preconditioner with respect to natural/large variations of the ice viscosity
- The solver performance for different linearization strategies
- The efficiency and scalability depends critically on subproblem solves with the coefficient matrix $\mathbf{A}$.
- Note also that the use of the stress-divergence form couples the solution of the components of the velocity, i.e. $\mathbf{A}$ is not block diagonal

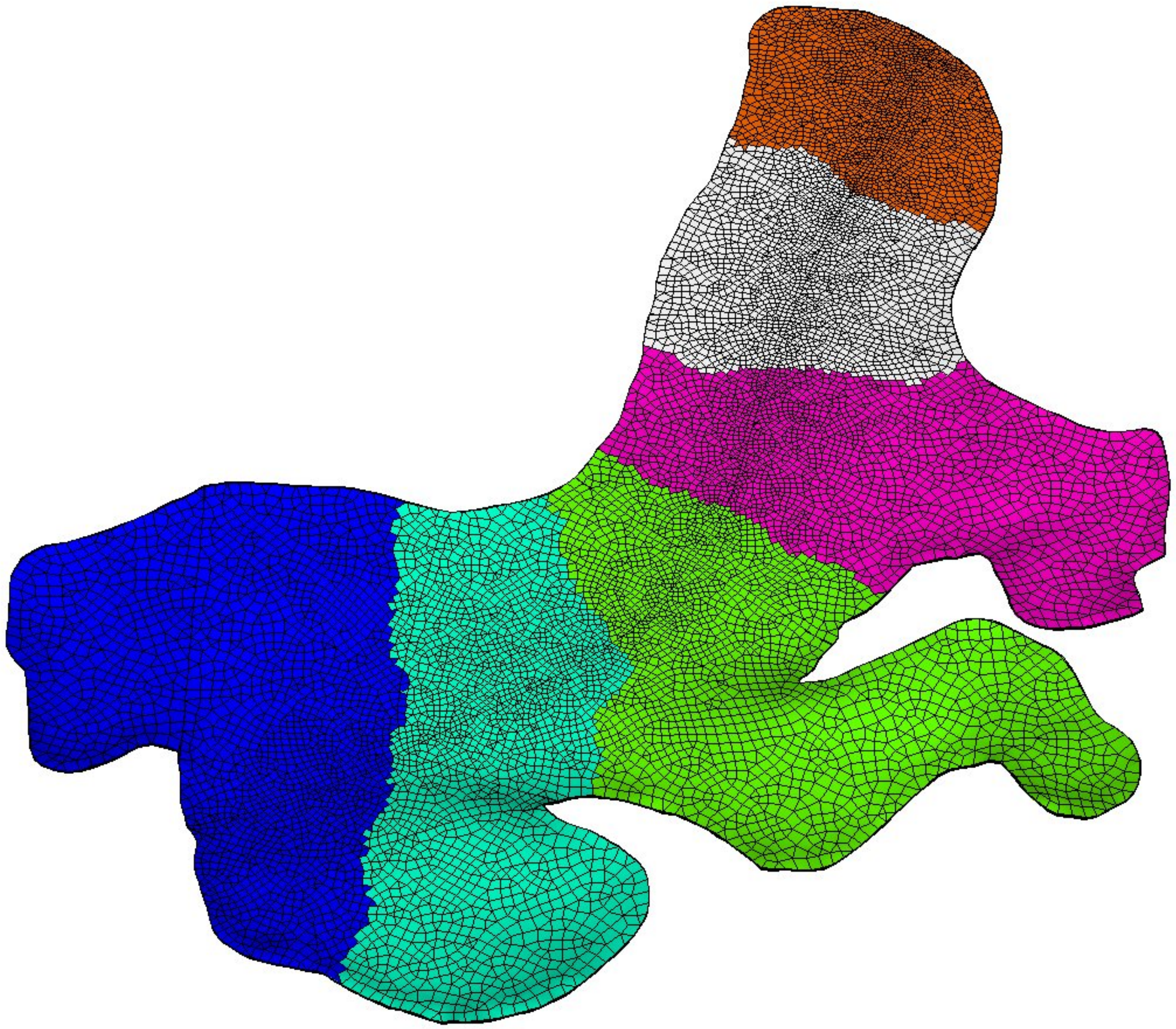


Figure 2: The partitioned mesh for Midtre Lovénbreen glacier (114080 nodes).

Preconditioned iteration counts for systems arising from different linearizations to solve a Midtre Lovénbreen flow ($A = 10^{-16} Pa^{-3} a^{-1}$)

	Picard	Hybrid $\delta_{NL} = 10^{-1}/2$	Hybrid $\delta_{NL} = 10^{-2}$
Nonlin Step	Iters	Linearization/Iters	Linearization/Iters
0	24	Picard/24	Picard/24
1	20	Picard/20	Picard/20
2	19	Picard/19	Picard/19
3	18	Picard/18	Picard/18
4	17	Picard/17	Picard/17
5	15	Newton/20	Picard/15
6	14	Newton/19	Picard/14
7	13	Newton/15	Picard/13
8	13	Newton/7	Picard/13
9	12	Convergence	Newton/16
10	11		Newton/14
11	10		Newton/7
12	9		Convergence
13. . .24	4.5 (Aver.)		
25	Convergence		

Preconditioned iteration counts for systems arising from different linearizations to solve ISMIP-HOM benchmark problem A with $L = 5$ km

	Picard	Hybrid $\delta_{NL} = 10^{-2}/2$	Hybrid $\delta_{NL} = 10^{-2}$
Nonlin Step	Iters	Linearization/Iters	Linearization/Iters
0	16	Picard/16	Picard/16
1	9	Picard/9	Picard/9
2	9	Picard/9	Picard/9
3	9	Picard/9	Picard/9
4	9	Picard/9	Newton/9
5	8	Newton/9	Newton/13
6	8	Newton/8	Newton/14
7	7	Newton/7	Newton/9
8	6	Convergence	Newton/7
9	6		Convergence
10	5		
11	5		
12	4		
13	4		
14	Convergence		

The effect of the element aspect ratio α_K on the efficiency of the preconditioner

- The ISMIP-HOM benchmark problem A with $L = 80$ km is solved on $N \times N \times M$ mesh
- The element aspect ratio $\alpha_K \sim$ diam(K)/(shortest element edge) varied
- The number n of mesh nodes kept approximately the same.
- N^0_{GCR} is the number of preconditioned GCR iterations to solve the initial guess.
- The initial guess is characterized by the Stokes equations with a constant viscosity (taking $I_2(\mathbf{D}) = 1$).

N	M	α_K	n	N^0_{GCR}
160	10	5	285131	18
128	16	10	282897	23
104	25	19.2	286650	25
90	34	30.2	289835	29
82	41	40	289338	30
65	65	80	287496	34

The robustness with respect to the problem size

- The ISMIP-HOM benchmark problem A with $L = 5$ km is solved on $N \times N \times M$ mesh.
- The element aspect ratio $\alpha_K = 5$ is kept to be the same.
- N^0_{GCR} is the number of preconditioned GCR iterations to solve the initial guess characterized by the Stokes equations with a constant viscosity.

N	n	N^0_{GCR}
20	9261	23
30	29791	25
40	68921	27
50	132651	29
60	226981	30

Key observations

- The robustness is not perfect but fairly satisfying
- The element aspect ratio affects the performance more clearly than the problem size
- If weak scalability is tested by refining the mesh only in the horizontal directions, super-linear scaling may be possible due to the decreasing α_K
- The preconditioner is effective for Newton systems also
- The scaled mass matrix approximation of the Schur complement work surprisingly well. More advanced approximations are possible but not motivated in the light of experimental results.

Opportunities to build a fully scalable solver

- The efficiency and scalability depends critically on solves with the coefficient matrix $\mathbf{A}$
- $\mathbf{A}$ is not block diagonal even for Picard scheme

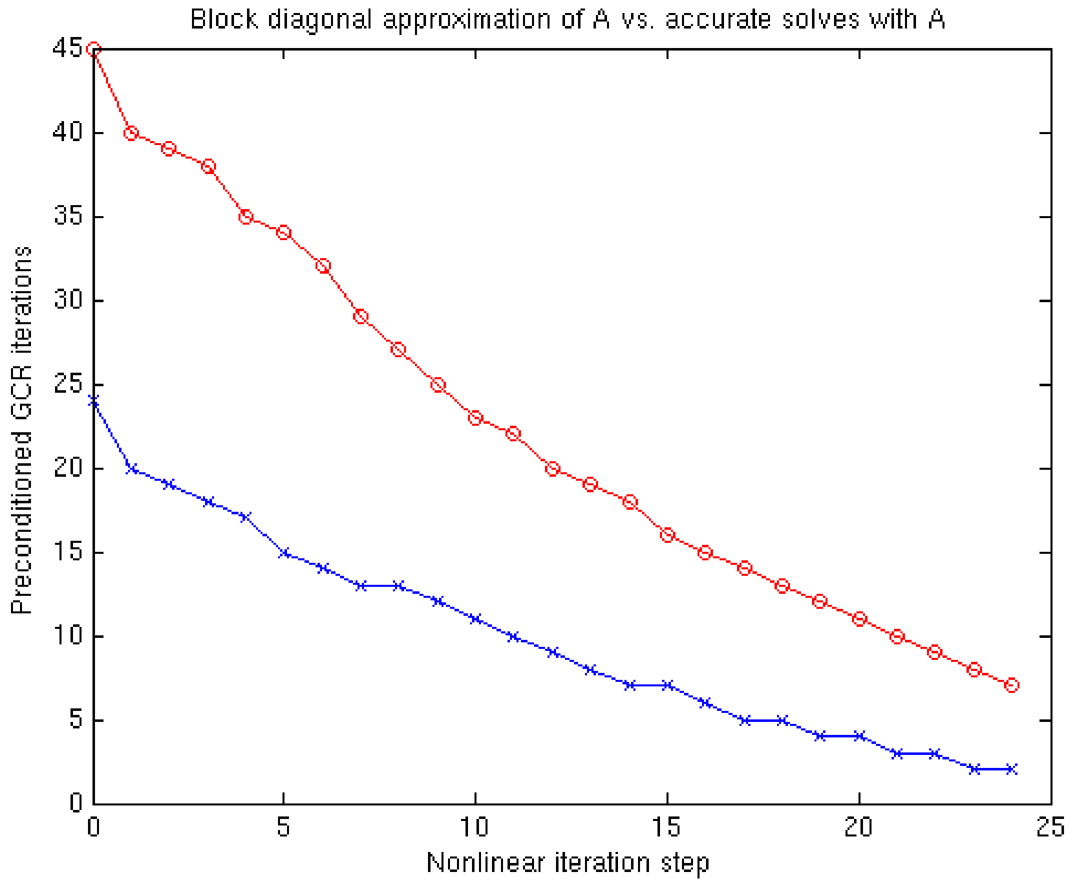
Many options available for solving systems $\mathbf{A}\delta\mathbf{V} = \hat{\mathbf{R}}_A$ (ongoing research)

- Preconditioned Krylov solvers based on a block-diagonal approximation P_A of $\mathbf{A}$, with the inverse of P_A approximated with multilevel methods
- Direct multilevel method acceleration of Krylov solvers
- FETI

Interfacing with external libraries (for example Hypre, Trilinos) is then usually needed.

Block diagonal approximation of $\mathbf{A}$ for Picard systems

$$\mathsf{P} = \begin{bmatrix} \mathsf{P}_A & \mathbf{B}^T \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}, \text{ with } P_A = \mathbf{A} \text{ (blue) or } P_A \sim \operatorname{diag}(\varepsilon\Delta\mathbf{v}_1, \varepsilon\Delta\mathbf{v}_2, \varepsilon\Delta\mathbf{v}_3)$$


Block diagonal approximation of A vs. accurate solves with A
Preconditioned GCR iterations
Nonlinear iteration step