GMRES with Multigrid

Lukas Spies

24th October 2019

Scott MacLachlan, Thomas Benson, Luke Olson

Stokes Equations

$$-\nabla \cdot (\nu \epsilon(\mathbf{u})) + \nabla p = \mathbf{f} \tag{1}$$
$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

$\rightarrow$ assuming ν to be constant simplifies (1) to

$$-\nu \nabla^2 \mathbf{u} + \nabla p = \mathbf{f}$$

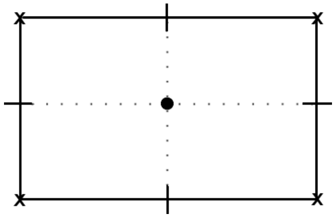
Weak form: Find $u \in H_0^1(\Omega)$ and $p \in L^2(\Omega)/\mathbb{R}$ such that

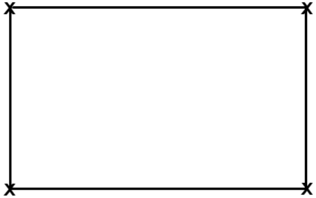
$$\begin{aligned} a(\mathbf{u},\mathbf{v}) + b(\mathbf{v},p) &= (\mathbf{f},\mathbf{v}) \quad \forall \mathbf{v} \in \mathbf{H}_0^1(\Omega) \\ b(\mathbf{u},q) &= 0 \qquad\quad \forall q \in L^2(\Omega)/\mathbb{R} \end{aligned}$$

$$\Rightarrow \begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}$$

Taylor-Hood elements

Q2 elements for velocity - Q1 elements for pressure

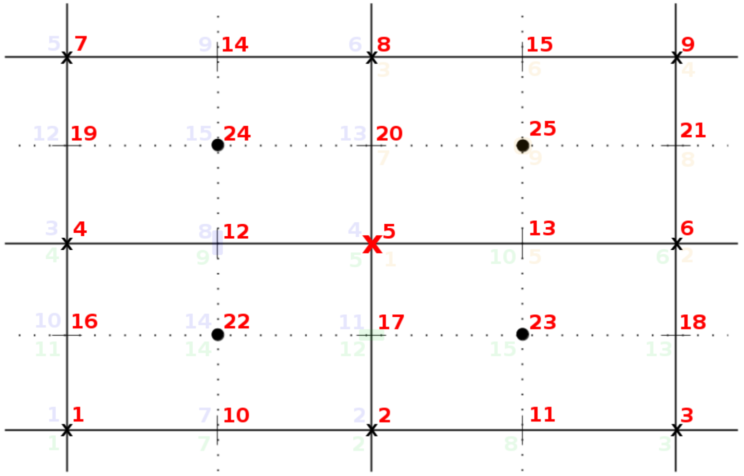




→ known to be inf-sup stable

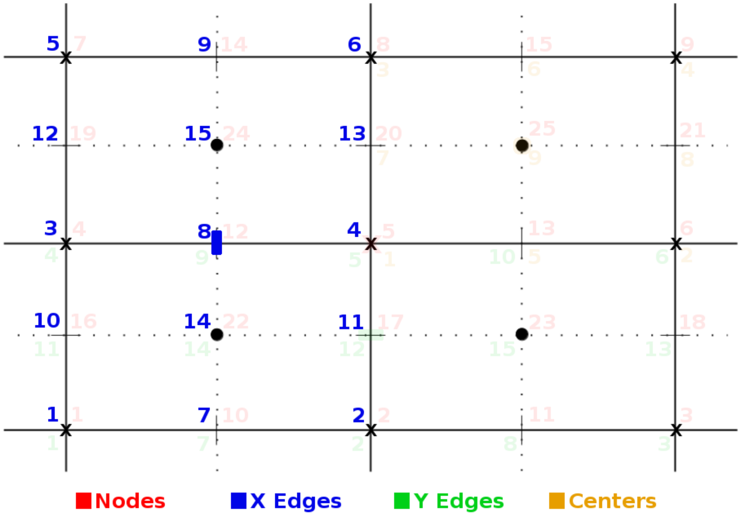
→ equally spaced, regular structure

Stencil structure

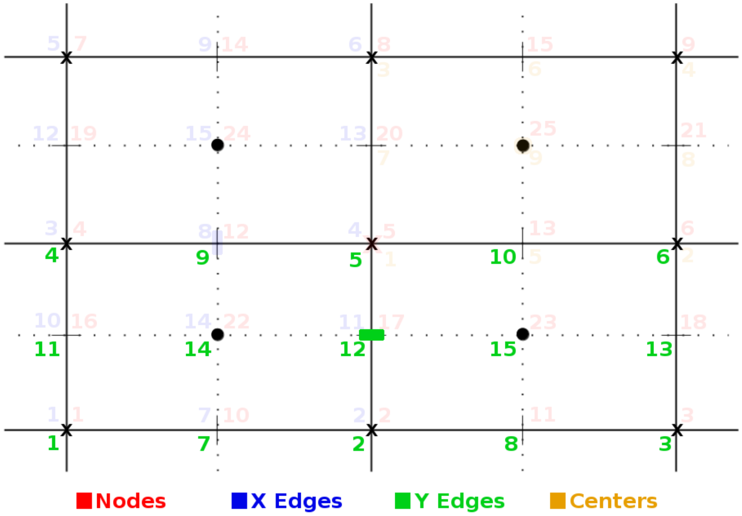


🟥 Nodes 🟦 X Edges 🟩 Y Edges 🟧 Centers

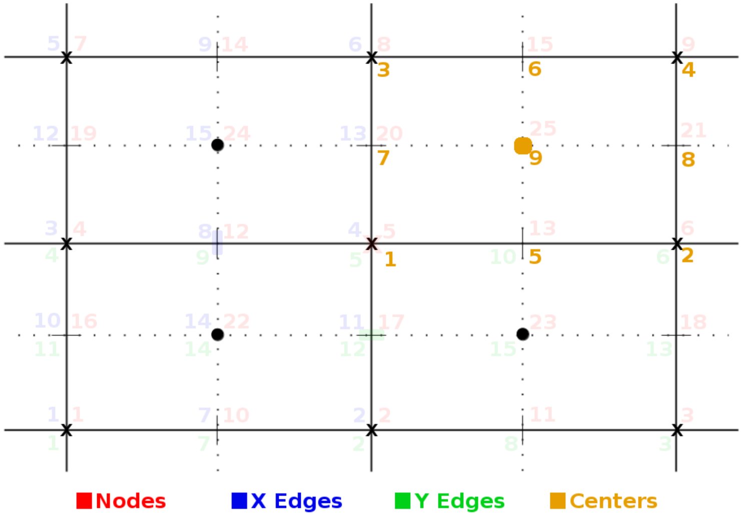
Stencil structure



Stencil structure



Stencil structure



Stencil extraction

Number of non-zeros per row (Laplacian):

	nodes	x-edges	y-edges	centers	**Total**
nodes	9	6	6	4	**25**
x-edges	6	3	4	2	**15**
y-edges	6	4	3	2	**15**
centers	4	2	2	1	**9**

$\rightarrow$ Natural consequence: localised stencil-based calculations

$\rightarrow$ Stencils independent from each other $\Rightarrow$ parallelisable

GMRES

Initial system

$$A\mathbf{x} = \mathbf{b}$$

GMRES minimises *2-norm* of residual

$$||\mathbf{r_m}||_2 = ||\mathbf{b} - A\mathbf{x_m}||_2$$

over all vectors in the *Krylov-subspace*

$$\mathbf{x_0} + \mathcal{K}_m(A, \mathbf{r_0}) = \mathbf{x_0} + span\{\mathbf{r_0}, A\mathbf{r_0}, ..., A^{m-1}\mathbf{r_0}\}.$$

(Right-)preconditioning:

$$AM^{-1}\mathbf{u} = \mathbf{b}$$
$$\mathbf{u} = M\mathbf{x}$$

$\rightarrow$ V-Cycle with rediscretisation on coarser grids

Braess-Sarazin

"Ideal" Braess-Sarazin update:

$$\begin{bmatrix}\mathbf{u}\\ p\end{bmatrix}^{new} = \begin{bmatrix}\mathbf{u}\\ p\end{bmatrix}^{old} + \omega_{BS}\begin{bmatrix}tD & B\\ B^T & 0\end{bmatrix}^{-1}\left(\begin{bmatrix}f\\ g\end{bmatrix} - A\begin{bmatrix}\mathbf{u}\\ p\end{bmatrix}^{old}\right)$$

$\rightarrow$ factorised system

$$\begin{bmatrix}tD & 0\\ B^T & S\end{bmatrix}\begin{bmatrix}I & \frac{1}{t}D^{-1}B\\ 0 & I\end{bmatrix}\begin{bmatrix}\delta\mathbf{u}\\ \delta p\end{bmatrix} = \begin{bmatrix}\mathbf{r_u}\\ r_p\end{bmatrix}$$

with Schur complement

$$S = -\frac{1}{t}B^T D^{-1} B$$

Braess-Sarazin

$\rightarrow$ compute S exactly

$\rightarrow$ approximately solve for solution using

$$S\delta p = r_p - \frac{1}{t} B^T D^{-1} \mathbf{r_u} \tag{3}$$

$$\delta \mathbf{u} = \frac{1}{t} D^{-1} (\mathbf{r_u} - B\delta p). \tag{4}$$

with one sweep of weighted Jacobi on (3), plugging solution into (4).

$\rightarrow$ parameter choice: $t = 1.1$, $\omega = 0.7$
(based on small experiments)

Fine-Scale Parallelism

GMRES:	matrix-vector and vector-vector calculations $\rightarrow$ easy parallelisation
Multigrid:	matrix-vector and vector-vector calculations $\rightarrow$ easy parallelisation
	interpolation/restriction $\rightarrow$ easy parallelisation
Braess-Sarazin:	matrix-vector and vector-vector calculations $\rightarrow$ easy parallelisation
Solve with S:	weighted Jacobi $\rightarrow$ easy parallelisation
Calculate $\delta\mathbf{u}$, δp:	matrix-vector and vector-vector calculations $\rightarrow$ easy parallelisation

GPU considerations

- fine-grain parallelism perfect for GPU's
- implemented using OpenCL
- additional cost of communication

  $\rightarrow$ moving initial data to GPU

  $\rightarrow$ moving final solution back to CPU
- Galerkin coarse-grid operators expensive
- direct solver on GPU?

Remarks on Numerical Experiments

- VCycle preconditioner

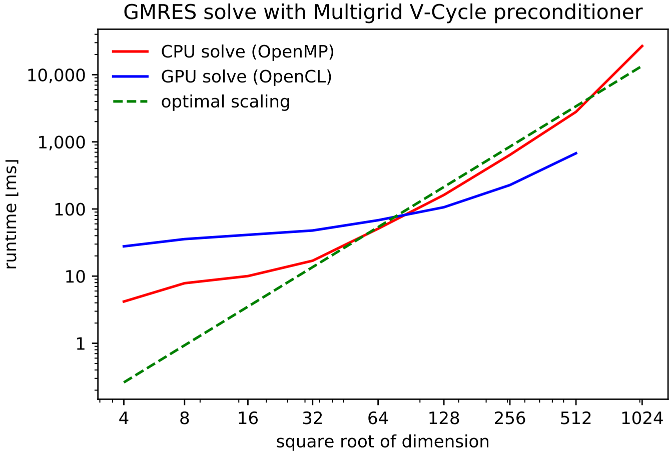
  possible solve on coarsest grid:

  1. approximate: Braess-Sarazin smoother
  2. exact: UMFPACK library on CPU

- GPU/CPU calculations identical
  (up to floating-point error)

- CPU: Intel Xeon E5; GPU: NVIDIA GeForce TITAN X

Speedup comparison



Summary

- structured grid
  $\rightarrow$ stencil formulation
- low memory cost
- many independent calculations
  $\rightarrow$ well parallelisable
- OpenCL speeds things up by a factor of up to 4 on larger meshes
- on small meshes CPU performs better
  $\rightarrow$ overhead of moving to GPU dominant

Future Work

- CUDA version
- Vanka relaxation

  Slow in serial (Braess-Sarazin better choice), potential benefits in parallel
- efficient Galerkin coarsening on GPU
- direct solve on GPU?

Thank you!