Parallel multilevel incomplete factorization of saddle point matrices

Jonas Thies* and Fred W. Wubs†

* Centre for Interdisciplinary Mathematics
  Uppsala University, Sweden
  jonas@math.uu.se

† Computational Mechanics & Numerical Mathematics
  University of Groningen, the Netherlands
  f.w.wubs@rug.nl

Euro-TUG 2012
Lausanne June 4-6
Outline

1. Objectives
2. Numerical methods
3. Robust ILU for Navier-Stokes on structured grids
4. The HYbrid Multi-Level Solver HYMLS
5. Augmented (‘bordered’) systems
6. Outlook and conclusions
Objectives

Numerical methods
Robust ILU for Navier-Stokes on structured grids
The HYbrid Multi-Level Solver HYMLS
Augmented ('bordered') systems
Outlook and conclusions
Bifurcations and instabilities in fluid dynamics

- understand the physics of a flow
- time integration gives a glance at a point in parameter space
- we want to traverse parameter space and find interesting points
- our applications: transition to turbulence, climate change
Benchmark problems

1. 3D Lid Driven Cavity

2. Boussinesq on the Globe. Domain from 60 degrees N Lat. to 60 degrees S Lat. Continent modelled by one line going from the north pole to 50 degrees S Lat. Depth 4000m.

   Numerical ingredients: continuation of steady states and periodic solutions (LOCA), nonlinear equations (NOX), eigenvalue problems (Jacobi-Davidson).

   **Key challenge:** efficient solution of large sparse linear systems.
Numerical methods
Fully coupled fully implicit approach

Incompressible Navier-Stokes equations:

\[
\frac{\partial \bar{u}}{\partial t} + \mathbf{N}(\bar{u}, \bar{u}) + \mathbf{L}\bar{u} + \nabla \mathbf{p} = 0 \\
\nabla \cdot \bar{u} = 0
\]

- Discretize (here second order symmetry-preserving finite differences on C-grid)
- Linearize by Newton’s method
- Structure of resulting linear systems (Saddle-point matrix):

\[
\begin{pmatrix}
\mathbf{L} + \mathbf{N} & \mathbf{Grad} \\
\mathbf{Div} & 0
\end{pmatrix}
\begin{pmatrix}
\bar{u} \\
\mathbf{p}
\end{pmatrix} =
\begin{pmatrix}
\mathbf{f}\bar{u} \\
\mathbf{f}\mathbf{p}
\end{pmatrix}
\]

(1)
Numerical continuation methods

- Nonlinear system of equations: \( F(x, p) = 0 \)
  - \( F : \mathbb{R}^{n+p} \rightarrow \mathbb{R}^n \): nonlinear function,
  - \( x \in \mathbb{R}^n \) state vector,
  - \( p \in \mathbb{R}^d \) parameter vector.

- Pseudo-arclength method:
  - Arc-length parameter \( s \), choose parameter \( \eta = \eta(s) \in p \);
  - \( \Rightarrow \) branch of solutions \( x_k, \eta(s_k) \).
  - Need an additional equation: normalize tangent
    \[
    \dot{x}_k^T (x - x_k) + \dot{\eta}_k (\eta - \eta_k) - \Delta s_k^2 = 0.
    \]
  - Predictor-Corrector scheme using Tangent and Newton’s, resp.

- \( \Rightarrow \) Linear systems with the Jacobian
  \[ J = \begin{pmatrix}
  \Phi & F_{\eta} \\
  \dot{x}_k^T & \dot{\eta}_k
  \end{pmatrix}. \]
FVM: our new package for constructing $\Phi$ and $F(x, p)$

- read XML input file
- domain decomposition: create Epetra_Map
- one or two layers of overlap...
- $\Rightarrow$ can build $\Phi$ and $F$ on each subdomain
- Fortran API for doing this (application scientist has to fill a stencil array in Fortran, all MPI hidden)
- NOX/LOCA interface defined once for all our test cases
Direct vs. iterative linear solvers

<table>
<thead>
<tr>
<th>Sparse Direct</th>
<th>Preconditioned Iterative</th>
</tr>
</thead>
<tbody>
<tr>
<td>robust and easy to use</td>
<td>usually not robust, depend on many parameters</td>
</tr>
<tr>
<td>comput. complexity $O(N^2)$ in 3D (N: number of unknowns)</td>
<td>can have optimal complexity $O(N)$</td>
</tr>
<tr>
<td>substantial fill-in $O(N^{4/3})$</td>
<td>save memory + CPU time by avoiding fill-in</td>
</tr>
</tbody>
</table>

Can we combine the best of both?
→ ILU close to LU and preserve properties
popular methods for $Ax=b$

- sparse direct (robust, only feasible in 2D)
- Krylov methods - require robust preconditioning
- Plenty of methods for elliptic PDEs:
  - FFT (Poisson, structured grid)
  - Geometric Multigrid (structured grid)
  - Algebraic Multigrid
  - Fast Multipole for particle dynamics and Maxwell equations

There is no fast algorithm for (Navier-)Stokes in 3D!
‘Physics-based’ Schur-complement preconditioners

- use simplified \( \tilde{K} \approx K \) as preconditioner
- \( \tilde{K} \) typically involves Poisson- or convection-diffusion like systems that are solved using multigrid;
- for instance:

\[
\tilde{K} = \begin{bmatrix}
A & O \\
D & \hat{S}
\end{bmatrix}
\]

where \( A = -\frac{1}{Re} L + N \).

The Schur-complement \( S = -DAG \) is dense, so it has to be approximated somehow by \( \hat{S} \) in the preconditioner.
Drawbacks of block preconditioners

- System split into velocity and pressure globally
- Artificial pressure boundary conditions
- Choice of $\tilde{C}$ very hard for high Reynolds Numbers
- Nested iterations
- How to choose ‘inner’ convergence criteria?
- No notion of a ‘coarse grid’ as in multigrid for elliptic PDEs
- Adding e.g. heat transfer is typically not feasible (multi-block matrices)

$\implies$ Not a good option for transition to turbulence and multi-physics problems
Robust ILU for Navier-Stokes on structured grids
Ingredients for effective and robust incomplete factorization

- Eliminate velocity and pressure nodes together
- Fill reducing ordering
- Fourier-like transformation
  - improves diagonal dominance
  - to get rid of unwanted couplings
- Drop by retaining principal submatrices
  - these submatrices will be positive definite if the matrix is positive definite
- For incompressible Navier Stokes equation, do not drop in divergence and gradient part
  - There is no increase of fill in this part (not even in direct method) on C-grid
Trilinos usage

- NOX/LOCA for nonlinearity
- implements Ifpack_Preconditioner
- uses Ifpack_Container class (sparse and dense)
- own interface to KLU for subdomains
- Amesos on coarsest level
- heavy use of Epetra, EpetraExt and Teuchos
A cartoon of the new algorithm

Stokes on a structured C-grid
A cartoon of the new algorithm, step 1

Domain decomposition
A cartoon of the new algorithm, step 2

Identify separators
A cartoon of the new algorithm, step 3

Elimination yields ‘geometric’ Schur-complement
Flux representation (‘coarse grid’)
A saddle point matrix has the following structure:

\[ K = \begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix}. \] (2)

**Definition 1**
A gradient-type matrix has at most two nonzero entries per row and its row sum is zero.

**Definition 2**
A saddle point matrix (2) is called an $F$-matrix if $A$ is positive definite and $B$ is a gradient-type matrix.

The Jacobian of the Stokes equations ($Re \to 0$) on a C-grid is an $F$-matrix.
Computing an LU decomposition of an $\mathcal{F}$-matrix

\[
\begin{bmatrix}
  A & B \\
  B^T & 0 
\end{bmatrix}
\begin{bmatrix}
  x_v \\
  x_p 
\end{bmatrix}
= 
\begin{bmatrix}
  f_v \\
  f_p 
\end{bmatrix}
\]

\(V - \text{nodes}\)  \(P - \text{nodes}\)

**Algorithm: LU decomposition of an $\mathcal{F}$-matrix.**

- Compute a fill-reducing ordering for the graph $F(A) \cup F(BB^T)$,
- during Gaussian elimination, insert the P-nodes to form $2 \times 2$ pivots whenever a coupling between a V-node and a P-node is encountered.

**Theorem 1**

In every step of the above algorithm, the resulting Schur complement is an $\mathcal{F}$-matrix.
How is fill generated in the direct approach?

\[ \begin{bmatrix}
\alpha & \beta & a^T & b^T \\
\beta & 0 & \hat{b}^T & 0 \\
a & \hat{b} & \hat{A} & \hat{B} \\
b & 0 & \hat{B}^T & O
\end{bmatrix} \]

Elimination step:
- Multiple of $\hat{b}\hat{b}^T$ is added to $\hat{A}$;
- $\hat{b}$ becomes denser as P-nodes are eliminated;
- So dropping in $\hat{A}$ doesn’t make sense.
Subdomains and ‘separator groups’;
Retain one pressure per subdomain.

This ordering exposes parallelism in the matrix:

\[ K \Rightarrow \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix}, \]

where \( K_{11} \) is block-diagonal.
The Schur complement

- **LU-decomposition of the matrices on the subdomains**, $K_{11} = L_{11} U_{11}$;
- **Schur-complement**: $S = K_{22} - K_{21} K_{11}^{-1} K_{12}$;
- $S$ retains structural and numerical properties of $K$;
- $S$ has only a few rather dense `B' columns (with at most two entries per row);
- Solve the system with $S$ by a preconditioned Krylov subspace method.
How can we maintain sparsity?

- Still an $\mathcal{F}$-matrix;
- All V-nodes on a separator are now connected to the same 2 P-nodes;
- Use orthogonal transformation to disconnect them.
How can we maintain sparsity?

- Still an F-matrix;
- All V-nodes on a separator are now connected to the same 2 P-nodes;
- Use orthogonal transformation to disconnect them.

⇒ Only one V-node per separator remains connected to P-nodes ($V_\Sigma$-nodes)
Dropping

- Use simple drop-by-position:
  - Drop all couplings between separator groups
  - ... and all couplings between $V_\Sigma$ and regular $V$-nodes.
Dropping

- Use simple drop-by-position:
  - Drop all couplings between separator groups
  - ... and all couplings between $V_\Sigma$ and regular $V$-nodes.

$\rightarrow$ Block diagonal preconditioner with a ‘reduced matrix’ $S_2$ in the lower right.
why it works

- **Orthogonal transformations:**
  - Eliminate most V-P couplings to avoid fill;
  - ‘Transfer operators’ defining coarse problem $S_2$.

- **Coarse problem $S_2$:** solve for flux $V_\Sigma$ through each separator;
  - Still an $F$-matrix in case of the Stokes equations;

- **Constraint preconditioning:**
  - no approximations in ‘Grad’ or ‘Div’ part;
  - mass is conserved exactly throughout.

- **Drop-by-position**
  - original properties preserved (symmetry, positiveness);
  - singular subsystems cannot occur.

- **No segregation of variables:**
  - velocity and pressure kept together;
  - no nested iterations.
Stokes equations: relative fill

2D Stokes-C: fill-in

- iterative (subdomain size 8x8)
- iterative (subdomain size 16x16)

<table>
<thead>
<tr>
<th>Grid Size</th>
<th>16x16</th>
<th>32x32</th>
<th>64x64</th>
<th>128x128</th>
<th>256x256</th>
<th>512x512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill-in</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16x16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32x32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64x64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128x128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256x256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512x512</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

J. Thies & F. W. Wubs
Parallel multilevel incomplete factorization of saddle point matrices
Stokes equations: number of iterations

2D Stokes-C: number of GMRES iterations on Schur-complement

- 8x8
- 16x16
2D lid-driven cavity

- Incompressible Navier-Stokes;
- Stretched structured grid (ratio \( \approx 5 \));
- Newton’s method;
- First Hopf-bifurcation at \( Re \approx 8375 \) (Tiesinga & Wubs 2002).
Navier-Stokes: convergence behavior

2D Driven Cavity, first Newton step: number of GMRES iterations

Convergence criterion: $\frac{||r||}{||r_0||} < 10^{-6}$
Navier-Stokes: achieving high accuracy

- Driven Cavity, $512 \times 512$ grid;
- Subdomain size: $8 \times 8$;
- Convergence tolerance $10^{-10}$;
- Preconditioned GMRES;

$\Rightarrow$ Some modes not captured using this subdomain size.
Navier-Stokes: robust at high Reynolds numbers

- Can compute highly unstable steady states;
- Moderate increase in number of iterations;
- Conv. tol $10^{-8}$ here.
The HYbrid Multi-Level Solver HYMLS
Multi-Level ILU

- Reduced problem has same structure as original matrix;
- Recursive application leads to $N \log N$ comp. complexity;
- Cartesian partitioning can be used on coarser levels because nodes retain their GID
- Discretization looks less structured on coarser grids
- Orthogonal transforms act as transfer operators (cf. unsmoothed aggregation!)
- ‘Transfer operators’ (Householder) can be constructed as follows
  - Start with constant test vector on separators (for uniform grid)
  - Apply transform, pick $V_{\Sigma}$ nodes to form next test vector
Multi-Level
Multi-Level
3D Navier-Stokes: weak scaling of direct method and HYMLS
3D Navier-Stokes: strong scaling of HYMLS
Objectives
Numerical methods
Robust ILU for Navier-Stokes on structured grids
The HYbrid Multi-Level Solver HYMLS
Augmented (‘bordered’) systems
Outlook and conclusions

3D Navier-Stokes: more levels

256³ runs on 1024 cores (2h setup, 100s solve) but is too memory intensive right now (2nd setup fails with bad_alloc, future work...)
Augmented (‘bordered’) systems
They are everywhere

\[
\begin{bmatrix}
A & V \\
W^T & C
\end{bmatrix}
\begin{bmatrix}
x \\
s
\end{bmatrix}
=
\begin{bmatrix}
f_x \\
f_c
\end{bmatrix}
\]

where $A$ is a large sparse matrix and $V$ and $W$ contain a number of vectors. Occur in:

- Continuation (Jacobian $A$ singular near turning point)
- Eigenvalue computation in Jacobi-Davidson method
- DO method for stochastic PDEs using implicit methods

In latter two methods one has to compute a correction on a space perpendicular to the current space.
Standard solution

Standard approach: Make block LU factorization

\[
\begin{bmatrix}
A & 0 \\
W^T & I
\end{bmatrix}
\begin{bmatrix}
I \\
0
\end{bmatrix}
\begin{bmatrix}
A^{-1}V \\
C - W^T A^{-1} V
\end{bmatrix}
\]

What if \( A \) becomes singular.

Arpack: targets 0 and 0.1
Incorporation in multilevel approach

Multilevel ILU comes in very handy. Example in two-level case:

\[
\begin{bmatrix}
A_{11} & A_{12} & V_1 \\
A_{21} & A_{22} & V_2 \\
W_1^T & W_2^T & C
\end{bmatrix}
= \begin{bmatrix}
I & A_{11}^{-1} A_{12} & A_{11}^{-1} V_1 \\
0 & A_{22} - A_{21} A_{11}^{-1} A_{12} & V_2 - A_{21} A_{11}^{-1} V_1 \\
0 & W_2^T - W_1^T A_{11}^{-1} A_{12} & C - W_1^T A_{11}^{-1} V_1
\end{bmatrix}
\]

- Coarsest level: direct method with pivoting to preclude instability.
- Indefiniteness likely to occur for low frequency modes. Problem pushed to coarsest grid.
- Coarsest system indef. $\Rightarrow$ original problem indef., indicates bifurcation.
Outlook and conclusions
Generalizations

Different coordinate systems

- spherical coordinates common in geophysics

\[ S_2 = r^2 \cdot c \cdot 2d\phi \]
\[ S_0 = d\nu d\zeta \]
\[ S = d\phi d\zeta \]

Flux-formulation \( \Rightarrow \mathcal{F}\)-matrix

Different discretizations:

- rotate \( \vec{v} \) by 45° \( \Rightarrow \mathcal{F}\)-matrix

Different physics:

- can solve Poisson, Convection-Diffusion, Stokes with the same technique
- can handle multiple variables, so adding heat transfer is easy
Possible improvements

**Memory Usage** too much temporary memory allocations right now

**Scalability** aggressive coarsening leads to decrease of cores used on coarser grids

**Deflation** to avoid ‘plateaus’ in GMRES (exploits bordered solver)

**Adaptivity:**
- Any domain decomposition can be used;
- Inhom. problems: short separators in regions of weak coupling.

**Unstructured grids:**
- Structure-preserving direct method?
Bifurcation analysis requires fast and robust linear algebra

We developed a solver that features
- Ease of use: only one parameter;
- Robustness: factorization doesn’t break down;
- Can be used as approximate Jacobian
- Parallelism: exposed on every level
- Grid-independent convergence for ILU
- Extendable to multi-physics problems
- Communication/computation like DD methods

Next steps
- Improvements on accuracy and performance.
- Do some nice (multiphysics) CFD problems.
- Look for generalizations.
References


