HYMLS: A parallel solver for steady coupled fluid-transport equations

Fred Wubs*, Weiyan Song* and Jonas Thies†

*Johann Bernoulli institute for mathematics and computing science
University of Groningen, the Netherlands
f.w.wubs@rug.nl

†Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Simulations- und Softwaretechnik, Germany
jonas.thies@dlr.de

PMAA14
Lugano, July 2, 2014

Outline

Problem Setting

Stability analysis

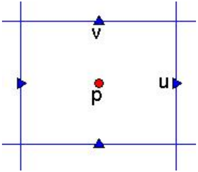
Objective

- 3D CFD problems, academic flow problems and geophysical applications (ocean models) $M\frac{du}{dt} = F(u, \mu)$ (possibly with noise)
- Compute branches of steady states: from $F(u, \mu) = 0$, compute $u(\mu)$ as function of μ.
- Compute stability of solution: solve eigenvalue problem $\lambda M v = F_u(u(\mu), \mu)v$
- Identify bifurcation points: compute μ for which $\lambda = 0$
- If $u(\mu) \equiv 0$ is a steady solution, we can find all bifurcation points from this solution at once from the eigenvalue problem $F_u(0, \mu)v = 0$

Key challenge: large sparse linear systems with the (shifted) Jacobian

Discretization

$$\frac{\partial \vec{u}}{\partial t} + \mathcal{N}(\vec{u}, \vec{u}) + \frac{1}{Re}\mathcal{L}\vec{u} + \nabla p = 0$$
$$\nabla \cdot \vec{u} = 0$$



- Note that $\mathcal{N}(\vec{u}, \hat{\vec{u}})$ is a bilinear form $\rightarrow$ in operator form $\mathcal{C}(\mathcal{A}\vec{u}. * \mathcal{B}\hat{\vec{u}})$.
- On closed domains it holds that $\int_\Omega \vec{u}\mathcal{N}(\vec{u}, \hat{\vec{u}})d\Omega = 0$ for any divergence free $\hat{\vec{u}}$. **To be preserved in discretization**
- Discretize (here second-order symmetry-preserving finite volumes on C-grid) $\rightarrow$ no artificial diffusion
- Structure of resulting linear systems (Saddle-point matrix):

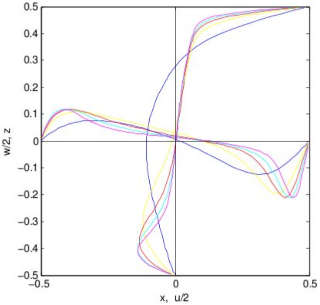
$$\begin{pmatrix} \frac{1}{Re}\mathbf{L} + \mathbf{N}(\vec{u}) & \mathbf{Grad} \\ \mathbf{Div} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} f_{\vec{u}} \\ f_p \end{pmatrix}$$

- Here $\mathbf{N}(\vec{u}) = \mathbf{C} * diag(\mathbf{A}\vec{u}) * \mathbf{B} + \mathbf{C} * diag(\mathbf{B}\vec{u}) * \mathbf{A}$ $\rightarrow$ store $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ during initialization together with linear parts

Requirements

Figure shows steady solutions at Reynolds numbers ranging from 100 to 2000 on 64^3 grid.

- Picard iteration (Oseen equation) stagnates for Re $>$ 200 $\rightarrow$ full Jacobian in linear system to be solved.
- Jacobian will eventually have positive eigenvalues.
- Big continuation steps should be possible (here 500).



Algorithmic overview

Parallel Data structure Trilinos: Epetra

Continuation program:

- Initialization
  - Partitioning + maps needed for parallelization
  - Set up templates for the matrix
  - Initialize solution
- Continuation using LOCA
  - Prediction
  - Correction using NOX.
    - Solve linear system using HYMLS
  - Eigenvalue computation using ANASAZI

Our workhorse: HYMLS

Some statements

- Dropping is safe for M-matrices only.

Some statements

- Dropping is safe for M-matrices only.
  No: For SPD matrix one can drop by retaining just principal submatrices
  Needs transformation to keep the relevant parts

Some statements

- Dropping is safe for M-matrices only.
  No: For SPD matrix one can drop by retaining just principal submatrices
  Needs transformation to keep the relevant parts
- Indefiniteness in (Navier)-Stokes matrix is always a big problem for the solution process.

Some statements

- Dropping is safe for M-matrices only.
  No: For SPD matrix one can drop by retaining just principal submatrices
  Needs transformation to keep the relevant parts
- Indefiniteness in (Navier)-Stokes matrix is always a big problem for the solution process.
  No: for certain types of discretizations one can iterate in the divergence free space.

Some statements

- Dropping is safe for M-matrices only.
  No: For SPD matrix one can drop by retaining just principal submatrices
  Needs transformation to keep the relevant parts
- Indefiniteness in (Navier)-Stokes matrix is always a big problem for the solution process.
  No: for certain types of discretizations one can iterate in the divergence free space.
- Grid independent convergence with an ILU factorization is not possible.

Some statements

- Dropping is safe for M-matrices only.
  No: For SPD matrix one can drop by retaining just principal submatrices
  Needs transformation to keep the relevant parts
- Indefiniteness in (Navier)-Stokes matrix is always a big problem for the solution process.
  No: for certain types of discretizations one can iterate in the divergence free space.
- Grid independent convergence with an ILU factorization is not possible.
  No: for instance NGICCG (vdPloeg,Botta,W, 1997) shows grid independent convergence. Number of iterations controlled by dropping parameter.

Some statements

- Dropping is safe for M-matrices only.
  No: For SPD matrix one can drop by retaining just principal submatrices
  Needs transformation to keep the relevant parts
- Indefiniteness in (Navier)-Stokes matrix is always a big problem for the solution process.
  No: for certain types of discretizations one can iterate in the divergence free space.
- Grid independent convergence with an ILU factorization is not possible.
  No: for instance NGICCG (vdPloeg,Botta,W, 1997) shows grid independent convergence. Number of iterations controlled by dropping parameter.
- A Navier-Stokes Jacobian can be approximated arbitrary close by an ILU factorization that will lead to an $O(N)$ algorithm.

Some statements

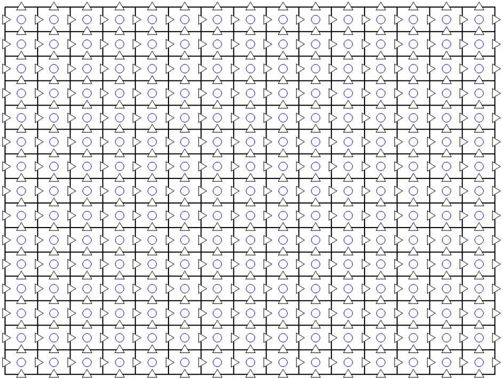
- Dropping is safe for M-matrices only.
  No: For SPD matrix one can drop by retaining just principal submatrices
  Needs transformation to keep the relevant parts
- Indefiniteness in (Navier)-Stokes matrix is always a big problem for the solution process.
  No: for certain types of discretizations one can iterate in the divergence free space.
- Grid independent convergence with an ILU factorization is not possible.
  No: for instance NGICCG (vdPloeg,Botta,W, 1997) shows grid independent convergence. Number of iterations controlled by dropping parameter.
- A Navier-Stokes Jacobian can be approximated arbitrary close by an ILU factorization that will lead to an $O(N)$ algorithm.
  Yes, regarding the above we expect this

Ingredients

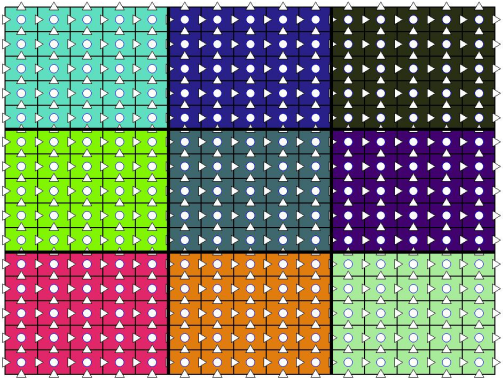
- Fill reducing ordering
- Local Fourier-like transformation
  - improves diagonal dominance
  - to get rid of unwanted couplings
- Drop by retaining principal submatrices
- For incompressible Navier Stokes equation, do not drop in divergence and gradient part
  - There is no increase of fill in this part (even not in direct method) on Arakawa A, B, and C-grids

A cartoon of the new algorithm

Stokes on a structured C-grid

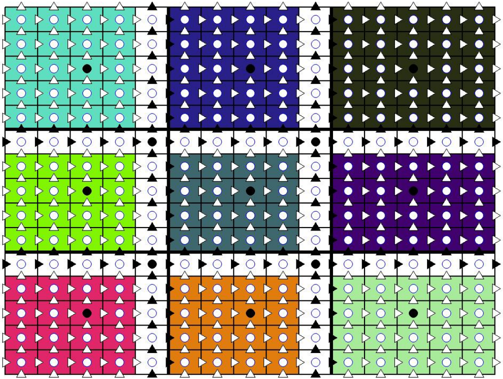


Domain decomposition



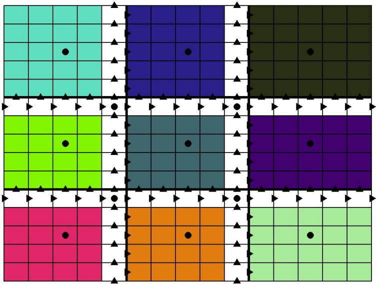
A cartoon of the new algorithm, step 2

Identify separators



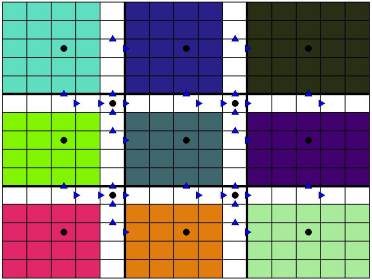
A cartoon of the new algorithm, step 3

Elimination yields 'geometric' Schur-complement



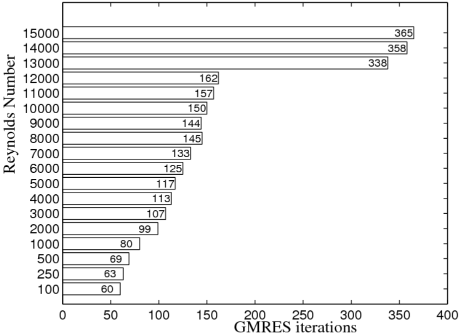
NB: All horizontal (vertical) velocities on a vertical (horizontal) separator are coupled to the pressure inside
What happens if we eliminate a velocity that is coupled to two pressure unknowns?

Flux representation ('coarse grid')



Transformation is such that amount of mass flowing through the separator remains the same.
Remaining V-nodes: separators are decoupled, V-nodes and coarse grid $(=V_\Sigma)$ nodes decoupled

Robust at high Reynolds numbers



- Can compute highly unstable steady states;
- Moderate increase in number of iterations;
- Conv. tol 10^{-8} here.

Numerical results

3D Lid Driven Cavity

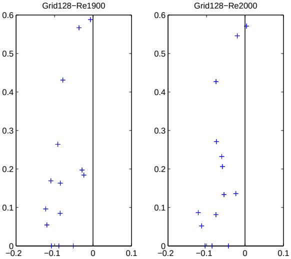
- All results obtained on Millipede cluster: 12 cores per node (opteron cores), 24GB per node
- One linear solve in process (8 digits gain)
- "Eff" indicates the deviation from optimal algorithmic ($O(N)$ behaviour) and parallel scalability.

nx	sx	nl	np	setup	eff	solve	eff	iters
32	4	3	8	16.4	1.00	5.8	1.00	76
48	4	3	32	11.5	0.83	4.6	0.94	78
64	4	3	32	37	1.13	12.5	1.07	81
81	3	4	27	75	0.95	34	1.22	102
128	4	4	64	171	1.30	100	2.15	115

- Number of levels influences computation time.
- The amount of parallel work for solution part is relative low.

Eigenvalue results

Eigenvalues at Re=1900 and 2000 on 128^3 grid.



Re	λ_{32}	λ_{64}	λ_{128}
1900	$-53.05 + 468i$	$-22.28 + 550i$	$-6.762 + 588i$
2000	$-42.35 + 463i$	$-10.73 + 545i$	$3.901 + 571i$
Re_c	2395	2093	1963

Eigenvalues in thousands on sequence of grids.

Our conclusion: Based on second order extrapolation Re_c is about 1930.

Matrix from stability study at Ra=2000 (singular at Ra=2600) 3D computation

nx	sx	nl	np	nnz S	setup	eff	solve	eff	iters
32	4	3	8	3361	11	1.00	6	1.00	85
64	4	4	32	1	64	2.91	31	2.58	109
128	4	4	64	10969	250	2.84	109	2.27	130

To study scalability we performed 2D computations

nx	sx	nl	np	nnz S	setup	eff	solve	eff	iters
64	4	3	16	646	0.318	1.00	0.20	1.00	65
128	4	3	32	3046	0.723	1.14	0.475	1.19	72
256	4	3	32	13126	2.85	1.12	2.58	1.61	76
512	4	3	32	54406	38.4	3.77	12.3	1.92	77
1024	4	3	128	221446	728	71.5	54.4	8.50	80
1024	4	4	128	13126	40	3.9	33	5.15	145
2048	4	4	128	54406	56	1.37	104	4.06	146

Poisson problem

3D Laplace, gain 10 digits

nx	sx	nl	np	nnz S	setup	eff	solve	eff	iters	Mem
128	4	4	16	631	11	1.00	16.1	1.00	57	
256	4	5	128	1	16	1.45	32	1.98	68	
512	4	5	512	631	52	2.36	177	5.50	87	0.62TB
1024	4	5	2048	7839	535	12.2	777	12.1	105	4.2 TB
					ML					
512			64		23	1.00	48.9	1.00	11	71GB
1024			128		586	6.37	211	1.08	10	291GB

Swapping occurs in 1024^3 case
There is room for improvement, 10 times slower than ML

Outlook and Conclusions

Concluding remarks

- Trilinos is a very pleasant framework to work with. Epetra will be extended to Tepetra. This allows also complex arithmetic. Handy for eigenvalue computations.

Concluding remarks

- Trilinos is a very pleasant framework to work with. Epetra will be extended to Tepetra. This allows also complex arithmetic. Handy for eigenvalue computations.
- ANASAZI works but dominates the computation time. Reuse of already computed basis and better targeting (using complex arithmetic). Preference for Jacobi-Davidson solver.

Concluding remarks

- Trilinos is a very pleasant framework to work with. Epetra will be extended to Tepetra. This allows also complex arithmetic. Handy for eigenvalue computations.
- ANASAZI works but dominates the computation time. Reuse of already computed basis and better targeting (using complex arithmetic). Preference for Jacobi-Davidson solver.
- HYMLS makes it possible to do real steady state computations.

Concluding remarks

- Trilinos is a very pleasant framework to work with. Epetra will be extended to Tepetra. This allows also complex arithmetic. Handy for eigenvalue computations.
- ANASAZI works but dominates the computation time. Reuse of already computed basis and better targeting (using complex arithmetic). Preference for Jacobi-Davidson solver.
- HYMLS makes it possible to do real steady state computations.
- Considerable effort to develop HYMLS, but is has high potential.
  - Easy extension with more physics, temperature, salt etc., avoids inner iterations.
  - Can be used as approximate Jacobian in Jac. Davidson for eigenvalue problems.
  - It benefits directly from improvements in Epetra
  - It gives control over the communication between processors

Concluding remarks

- Trilinos is a very pleasant framework to work with. Epetra will be extended to Tepetra. This allows also complex arithmetic. Handy for eigenvalue computations.
- ANASAZI works but dominates the computation time. Reuse of already computed basis and better targeting (using complex arithmetic). Preference for Jacobi-Davidson solver.
- HYMLS makes it possible to do real steady state computations.
- Considerable effort to develop HYMLS, but is has high potential.
  - Easy extension with more physics, temperature, salt etc., avoids inner iterations.
  - Can be used as approximate Jacobian in Jac. Davidson for eigenvalue problems.
  - It benefits directly from improvements in Epetra
  - It gives control over the communication between processors
- Grid independent convergence is possible with ILU factorization.

Concluding remarks

- Trilinos is a very pleasant framework to work with. Epetra will be extended to Tepetra. This allows also complex arithmetic. Handy for eigenvalue computations.
- ANASAZI works but dominates the computation time. Reuse of already computed basis and better targeting (using complex arithmetic). Preference for Jacobi-Davidson solver.
- HYMLS makes it possible to do real steady state computations.
- Considerable effort to develop HYMLS, but is has high potential.
  - Easy extension with more physics, temperature, salt etc., avoids inner iterations.
  - Can be used as approximate Jacobian in Jac. Davidson for eigenvalue problems.
  - It benefits directly from improvements in Epetra
  - It gives control over the communication between processors
- Grid independent convergence is possible with ILU factorization.
- Indefiniteness in (Navier)-Stokes matrix for standard A,B,C-grid can be treated exactly; method is provably robust for Stokes for every size of subdomain. Never had stagnation in Navier-Stokes.

References

- F.W.Wubs and J.Thies, ''A robust two-level incomplete factorization for (Navier-) Stokes saddle point matrices, SIAM J. Matrix Anal. Appl., 32:1475 - 1499, 2011.
- Jonas Thies and Fred Wubs, Design of a Parallel Hybrid Direct/Iterative Solver for CFD Problems. *Proceedings 2011 Seventh IEEE International Conference on eScience*, 5-8 December 2011, Stockholm, Sweden, pages 387 -394, 2011.
- G.L.G. Sleijpen and F.W. Wubs. Exploiting Multilevel Preconditioning Techniques in Eigenvalue Computations. SIAM Journal on Scientific Computing, 25(4):1249-1272, 2003.
- H.A. Dijkstra et al. Numerical Bifurcation Methods and their Application to Fluid Dynamics: Analysis beyond Simulation, *Communications in Computational Physics (CiCP)*, 2014. .