Fast Iterative Solvers for Buoyancy Driven Flow Problems

David Silvester
School of Mathematics
University of Manchester
Background

- Philip Gresho & David Griffiths & David Silvester
  *Adaptive time-stepping for incompressible flow; part I: scalar advection-diffusion*

- David Kay & Philip Gresho & David Griffiths & David Silvester
  *Adaptive time-stepping for incompressible flow; part II: Navier-Stokes equations*
For full details see

- Howard Elman, Milan Mihajlović and David Silvester. 
  Fast iterative solvers for buoyancy driven flow problems. 
  MIMS Eprint 2010.75, Manchester Institute for Mathematical Sciences.
Buoyancy driven flow

\[
\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} - \nu \nabla^2 \vec{u} + \nabla p = \vec{j} T \quad \text{in } \mathcal{W} \equiv \Omega \times (0, T)
\]
\[
\nabla \cdot \vec{u} = 0 \quad \text{in } \mathcal{W}
\]
\[
\frac{\partial T}{\partial t} + \vec{u} \cdot \nabla T - \nu \nabla^2 T = 0 \quad \text{in } \mathcal{W}
\]

Boundary and Initial conditions

\[
\vec{u} = \vec{0} \quad \text{on } \Gamma \times [0, T]; \quad \vec{u}(\vec{x}, 0) = \vec{0} \quad \text{in } \Omega.
\]
\[
T = T_g \quad \text{on } \Gamma_D \times [0, T]; \quad \nu \nabla T \cdot \vec{n} = 0 \quad \text{on } \Gamma_N \times [0, T];
\]
\[
T(\vec{x}, 0) = T_0(\vec{x}) \quad \text{in } \Omega.
\]
“Smart Integrator” (SI)

- **Optimal time-stepping:** time-steps automatically chosen to “follow the physics”.
- **Black-box implementation:** few parameters that have to be estimated a priori.
- **Algorithm efficiency:** solve linear equations at every timestep.
“Smart Integrator” (SI)

- **Optimal time-stepping**: time-steps automatically chosen to “follow the physics”.
- **Black-box implementation**: few parameters that have to be estimated a priori.
- **Algorithm efficiency**: solve linear equations at every timestep.
- **Solver efficiency**: see later ...
Trapezoidal Rule (TR) time discretization

We subdivide $[0, T]$ into time levels $\{t_i\}_{i=1}^N$. Given $(\vec{u}^n, p^n)$ at time level $t_n$, $k_{n+1} := t_{n+1} - t_n$, compute $(\vec{u}^{n+1}, p^{n+1})$ via

$$\frac{2}{k_{n+1}} \vec{u}^{n+1} + \vec{w}^{n+1} \cdot \nabla \vec{u}^{n+1} - \nu \nabla^2 \vec{u}^{n+1} + \nabla p^{n+1} = f^{n+1}$$

$$-\nabla \cdot \vec{u}^{n+1} = 0 \quad \text{in } \Omega$$

$$\vec{u}^{n+1} = \vec{g}^{n+1} \quad \text{on } \Gamma_D$$

$$\nu \nabla \vec{u}^{n+1} \cdot \vec{n} - p^{n+1} \vec{n} = \vec{0} \quad \text{on } \Gamma_N$$

with second-order linearization

$$\vec{w}^{n+1} = (1 + \frac{k_{n+1}}{k_n}) \vec{u}^n - \frac{k_{n+1}}{k_n} \vec{u}^{n-1}$$

$$f^{n+1} = \frac{2}{k_{n+1}} \vec{u}^n + \nu \nabla^2 \vec{u}^n - \vec{u}^n \cdot \nabla \vec{u}^n - \nabla p^n$$
Adaptive time stepping components

- Starting from rest, $\vec{u}^0 = \vec{0}$, and given a steady-state temperature boundary condition $T(\vec{x}, t) = T_g$, we model the impulse with a time-dependent boundary condition:

$$T(\vec{x}, t) = T_g (1 - e^{-5t}) \quad \text{on } \Gamma_D \times [0, T].$$

We also choose a very small initial timestep, typically, $k_1 = 10^{-9}$. 
Adaptive time stepping components

Starting from rest, \( \vec{u}^0 = \vec{0} \), and given a steady-state temperature boundary condition \( T(\vec{x}, t) = T_g \), we model the impulse with a time-dependent boundary condition:

\[
T(\vec{x}, t) = T_g(1 - e^{-5t}) \quad \text{on } \Gamma_D \times [0, T].
\]

We also choose a very small initial timestep, typically, \( k_1 = 10^{-9} \).

The following parameters must be specified:

- time accuracy tolerance \( \varepsilon_t \) (10\(^{-5}\))
- GMRES tolerance \( \text{itol} \) (10\(^{-6}\))
- GMRES iteration limit \( \text{maxit} \) (50)
Rayleigh-Bernard convection

\[ T_c \]

\[ \bar{g} \]

\[ T_h \]
Problem I: Timestep & Kinetic Energy: $\varepsilon_t = 10^{-6}$
Reference Point Temperature: $\varepsilon_t = 10^{-6}$
“Smart Integrator” (SI) revisited

- Optimal time-stepping
- Black-box implementation
- Algorithm efficiency

- **Solver efficiency**: the linear solver convergence rate is robust with respect to the mesh size $h$ and the flow problem parameters.
Finite element matrix formulation

Introducing the basis sets

\[ X_h = \text{span}\{ \vec{\phi}_i \}_{i=1}^{n_u}, \quad \text{Velocity basis functions;} \]
\[ M_h = \text{span}\{ \psi_j \}_{j=1}^{n_p}, \quad \text{Pressure basis functions.} \]
\[ T_h = \text{span}\{ \phi_k \}_{k=1}^{n_T}, \quad \text{Temperature basis functions;} \]

gives the method-of-lines discretized system:

\[
\begin{pmatrix}
M & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & M
\end{pmatrix}
\begin{pmatrix}
\frac{\partial \vec{u}}{\partial t} \\
\frac{\partial p}{\partial t} \\
\frac{\partial T}{\partial t}
\end{pmatrix}
+ \begin{pmatrix}
F & B^T & - \frac{\circ}{M} \\
B & 0 & 0 \\
0 & 0 & F
\end{pmatrix}
\begin{pmatrix}
\vec{u} \\
p \\
T
\end{pmatrix}
= \begin{pmatrix}
\vec{0} \\
p \\
g
\end{pmatrix}
\]

with a (vertical–) mass matrix:

\[
\left( \frac{\circ}{M} \right)_{ij} = ([0, \phi_i], \phi_j)
\]
Preconditioning strategy

\[
\begin{pmatrix}
F & B^T & -\frac{\circ}{M} \\
B & 0 & 0 \\
0 & 0 & F
\end{pmatrix}
\mathcal{P}^{-1}
\begin{pmatrix}
\alpha^u \\
\alpha^p \\
\alpha^T
\end{pmatrix}
= \begin{pmatrix}
f^u \\
f^p \\
f^T
\end{pmatrix}
\]

Given \( S = BF^{-1}B^T \), a perfect preconditioner is given by

\[
\begin{pmatrix}
F & B^T & -\frac{\circ}{M} \\
B & 0 & 0 \\
0 & 0 & F
\end{pmatrix}
\mathcal{P}^{-1}
\begin{pmatrix}
F^{-1} & F^{-1}B^TS^{-1} & F^{-1}\frac{\circ}{M}F^{-1} \\
0 & -S^{-1} & 0 \\
0 & 0 & F^{-1}
\end{pmatrix}
\]

\[
= \begin{pmatrix}
I & 0 & 0 \\
BF^{-1} & I & BF^{-1}\frac{\circ}{M}F^{-1} \\
0 & 0 & I
\end{pmatrix}
\]
For an efficient preconditioner we need to construct a sparse approximation to the “exact” Schur complement

\[ S^{-1} = (BF^{-1}B^T)^{-1} \]

See Chapter 8 of

- Howard Elman & David Silvester & Andrew Wathen
  *Finite Elements and Fast Iterative Solvers: with applications in incompressible fluid dynamics*

For an efficient implementation we must also have an efficient AMG (convection-diffusion) solver ...
1 SUMMARY

Given an \( n \times n \) sparse matrix \( A \) and an \( n \)-vector \( z \), \texttt{HSL\_MI20} computes the vector \( x = Mz \), where \( M \) is an algebraic multigrid (AMG) \( v \)-cycle preconditioner for \( A \). A classical AMG method is used, as described in [1] (see also Section 5 below for a brief description of the algorithm). The matrix \( A \) must have positive diagonal entries and (most of) the off-diagonal entries must be negative (the diagonal should be large compared to the sum of the off-diagonals). During the multigrid coarsening process, positive off-diagonal entries are ignored and, when calculating the interpolation weights, positive off-diagonal entries are added to the diagonal.

Reference


ATTRIBUTES — Version: 1.1.0 Types: Real (single, double). Uses: \texttt{HSL\_MA48}, \texttt{HSL\_MC65}, \texttt{HSL\_ZD11}, and the LAPACK routines \texttt{GETRF} and \texttt{GETRS}. Date: September 2006. Origin: J. W. Boyle, University of Manchester and J. A. Scott, Rutherford Appleton Laboratory. Language: Fortran 95, plus allocatable dummy arguments and allocatable components of derived types. Remark: The development of \texttt{HSL\_MI20} was funded by EPSRC grants EP/C000528/1 and GR/S42170.
GMRES convergence close to steady state with $k_n \sim 4$.
Note that $\nu = 0.0218$ and $\nu = 0.00306$. 
Problem II: 1:4 cavity domain

Lateral heating: Hopf Bifurcation
Problem II: Kinetic Energy: $\varepsilon_t = 3 \times 10^{-5}$
Problem II: Time step history: $\varepsilon_t = 3 \times 10^{-5}$
Problem XXX: 8:1 cavity domain
Problem XXX: $31 \times 248$ stretched grid
Problem XXX: Snapshot Solution

Isotherms: $t=1200$

Streamlines: $t=1200$
Problem XXX: Time step history: $\varepsilon_t = 3 \times 10^{-5}$
GMRES convergence for snapshot solution with $k_n \sim 0.082$. Note that $\nu = 0.00145$ and $\nu = 0.00203$. 
What have we achieved?

- **Black-box implementation**: few parameters that have to be estimated a priori.

- **Optimal complexity**: essentially $O(n)$ flops per iteration, where $n$ is dimension of the discrete system.

- **Efficient linear algebra**: convergence rate is (essentially) independent of $h$. Given an appropriate time accuracy tolerance convergence is also robust with respect to $\nu$.
Adaptive Time Stepping AB2–TR

Consider the simple ODE $\dot{u} = f(u)$

Manipulating the truncation error terms for TR and AB2 gives the estimate

$$T_n = \frac{u_{n+1} - u_{n+1}^*}{3(1 + \frac{k_n}{k_{n+1}})}$$

Given some user-prescribed error tolerance $\text{tol}$, the new time step is selected to be the biggest possible such that $\|T_{n+1}\| \leq \text{tol} \times u_{\text{max}}$. This criterion leads to

$$k_{n+2} := k_{n+1} \left( \frac{\text{tol} \times u_{\text{max}}}{\|T_n\|} \right)^{1/3}$$

But look out for “ringing” ...
Stabilized AB2–TR

To address the instability issues:

- We rewrite the AB2–TR algorithm to compute updates $v_n$ and $w_n$ scaled by the time-step:

  $$u_{n+1} - u_n = \frac{1}{2} k_{n+1} v_n; \quad u^*_n - u^* = k_{n+1} w_n.$$  

- We perform time-step averaging every $n^* = 10$ steps:

  $$u_n := \frac{1}{2} (u_n + u_{n-1}); \quad u_{n+1} := u_n + \frac{1}{4} k_{n+1} v_n; \quad \dot{u}_{n+1} := \frac{1}{2} v_n.$$  

Contrast this with the standard acceleration obtained by “inverting” the TR formula:

$$\dot{u}_{n+1} = \frac{2}{k_{n+1}} (u_{n+1} - u_n) - \dot{u}_n = v_n - \dot{u}_n.$$
Schur complement approximation – I

Introducing the diagonal of the velocity mass matrix

\[ M_* \sim M_{ij} = (\vec{\phi}_i, \vec{\phi}_j), \]

gives the “least-squares commutator preconditioner”:

\[
(BF^{-1}B^T)^{-1} \approx \underbrace{\left( B M_*^{-1} B^T \right)^{-1}}_{amg} \left( B M_*^{-1} F M_*^{-1} B^T \right) \underbrace{\left( B M_*^{-1} B^T \right)^{-1}}_{amg}
\]
Schur complement approximation – II

Introducing associated pressure matrices

\[ M_p \sim (\nabla \psi_i, \nabla \psi_j), \quad \text{mass} \]
\[ A_p \sim (\nabla \psi_i, \nabla \psi_j), \quad \text{diffusion} \]
\[ N_p \sim (\vec{w}_h \cdot \nabla \psi_i, \psi_j), \quad \text{convection} \]
\[ F_p = \frac{2}{\kappa_{n+1}} M_p + \nu A_p + N_p, \quad \text{convection-diffusion} \]

gives the “pressure convection-diffusion preconditioner”:

\[ (BF^{-1}B^T)^{-1} \approx M_p^{-1}F_p A_p^{-1} \]

\[ \text{amg} \]