Fast Solvers for Incompressible Flow Problems III

David Silvester
University of Manchester

Lecture III

- $$-\nabla^2 \vec{u} + \nabla p = \vec{0}; \quad \nabla \cdot \vec{u} = 0$$
- $$\vec{u} \cdot \nabla \vec{u} - \nu \nabla^2 \vec{u} + \nabla p = \vec{0}; \quad \nabla \cdot \vec{u} = 0$$
- $$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} - \nu \nabla^2 \vec{u} + \nabla p = 0; \quad \nabla \cdot \vec{u} = 0$$

Navier-Stokes Equations

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

$$\nabla \cdot \vec{u} = 0 \qquad \text{in } \mathcal{W}$$

Boundary and Initial conditions

$$\vec{u} = \vec{g} \quad \text{on } \Gamma_D \times [0, T];$$

$$\nu \nabla \vec{u} \cdot \vec{n} - p \vec{n} = \vec{0} \quad \text{on } \Gamma_N \times [0, T];$$

$$\vec{u}(\vec{x}, 0) = \vec{u}_0(\vec{x}) \quad \text{in } \Omega.$$

Finite element formulation

Introducing the basis sets

$$V_h = \text{span}\{\,\vec{\phi}_i\}_{i=1}^{n_u}, \quad \text{Velocity basis functions;}$$
$$Q_h = \text{span}\{\psi_j\}_{j=1}^{n_p}, \quad \text{Pressure basis functions.}$$

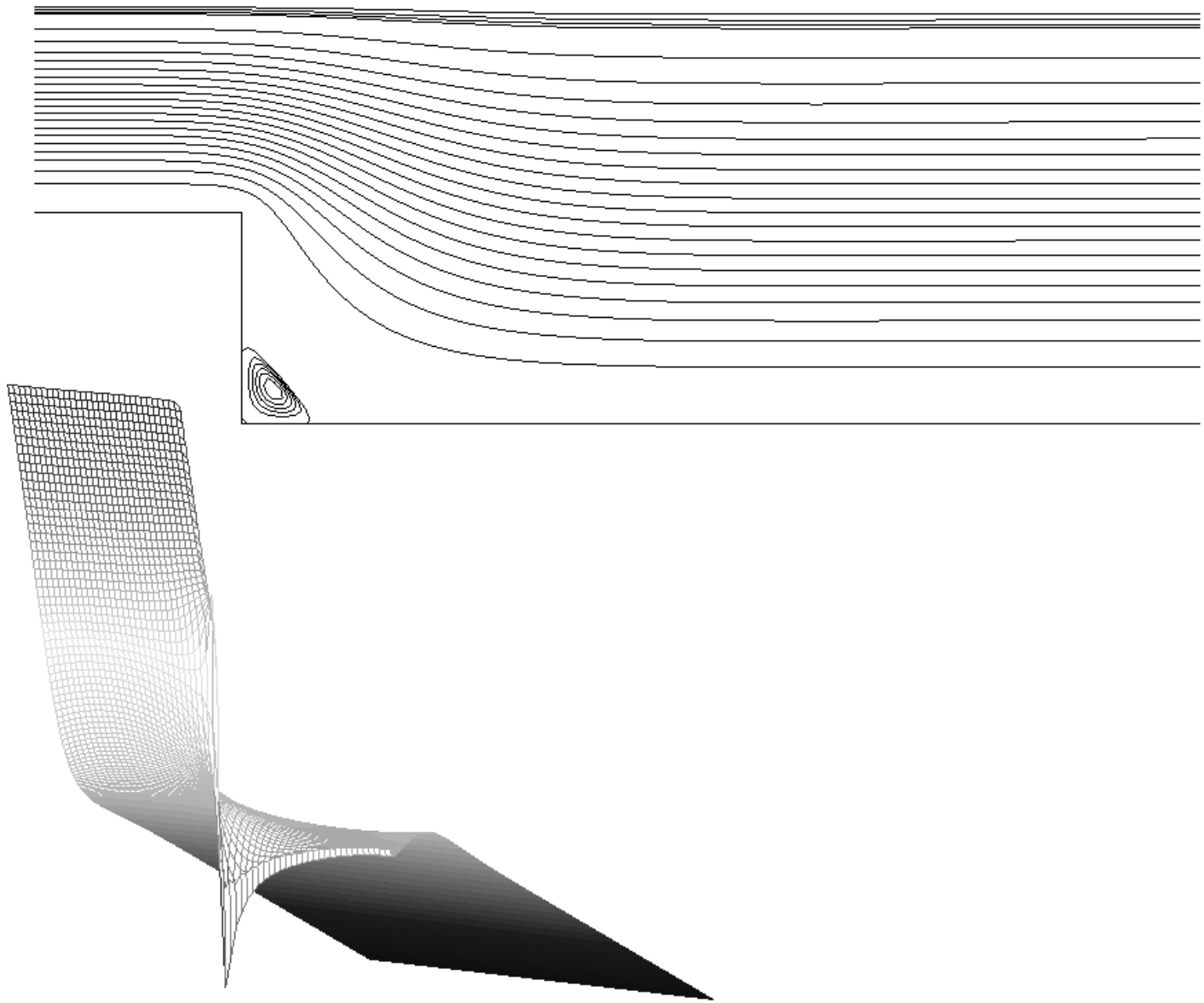
gives the method-of-lines discretized system:

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} N(\,\vec{u}) + \nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} \vec{f} \\ 0 \end{pmatrix}$$

with associated matrices

$$N_{ij} = (\,\vec{u} \cdot \nabla\, \vec{\phi}_i,\, \vec{\phi}_j), \quad \text{convection}$$
$$A_{ij} = (\nabla\, \vec{\phi}_i, \nabla\, \vec{\phi}_j), \quad \text{diffusion}$$
$$B_{ij} = -(\nabla \cdot\, \vec{\phi}_j, \psi_i), \quad \text{divergence .}$$

Example: Flow over a Step



Spatial Discretization

The method-of-lines discretized system is a semi-explicit system of DAEs:

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} N(\vec{u}) + \nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} \vec{f} \\ 0 \end{pmatrix}$$

- The DAEs have index equal to two
- The discrete problem is nonlinear $N(\vec{u}) + \nu A =: F(\vec{u})$

Spatial Discretization

The method-of-lines discretized system is a semi-explicit system of DAEs:

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} N(\vec{u}) + \nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} \vec{f} \\ 0 \end{pmatrix}$$

- The DAEs have index equal to two
- The discrete problem is nonlinear $N(\vec{u}) + \nu A =: F(\vec{u})$
- Implicit approximation in time does not look attractive!

Rest of the talk

- Time Stepping Algorithms

Rest of the talk

- Time Stepping Algorithms
  - Review : 1967 – 1999
  - Update : 2000 – 2010

Rest of the talk

- Time Stepping Algorithms
  - Review : 1967 – 1999
  - Update : 2000 – 2010
- Optimally preconditioned GMRES:

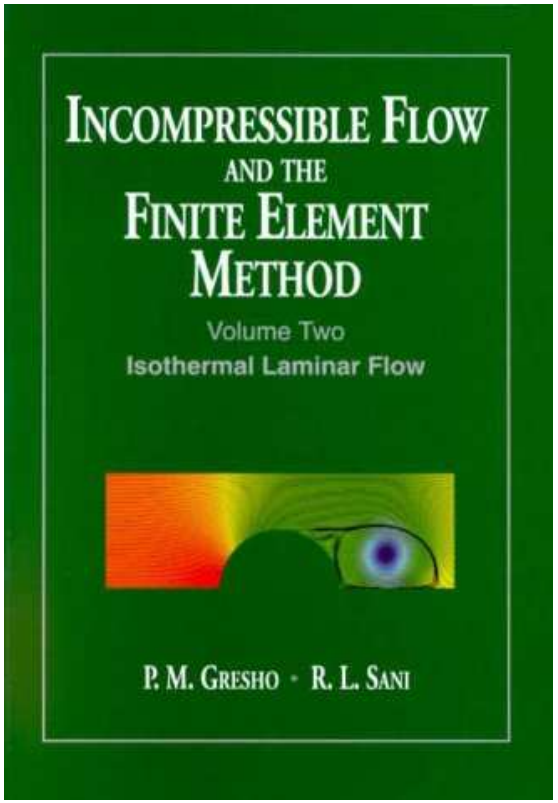
Rest of the talk

- Time Stepping Algorithms
  - Review : 1967 – 1999
  - Update : 2000 – 2010
- Optimally preconditioned GMRES:
  - Pressure Convection-Diffusion preconditioner
  - Least-squares commutator preconditioner

Rest of the talk

- Time Stepping Algorithms
  - Review : 1967 – 1999
  - Update : 2000 – 2010
- Optimally preconditioned GMRES:
  - Pressure Convection-Diffusion preconditioner
  - Least-squares commutator preconditioner
- A proof-of-concept implementation:
  - The IFISS 3.1 MATLAB Toolbox

- Review : 1967 – 1999
  - 1967
  - 1982
  - 1991
  - 1999



$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} N(\vec{u}) + \nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} \vec{f} \\ 0 \end{pmatrix}$$

- 1967

A simple splitting/projection approach — I

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} F(\vec{u}) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} \vec{f} \\ 0 \end{pmatrix}$$

Given a time step dT and a parameter $0 < \gamma \leq 2$

Algorithm: Chorin, 1967; Temam, 1969

for $k = 0, 1, \ldots$

solve $M \frac{\partial \vec{u}_*}{\partial t} + F(\vec{u}_k)\,\vec{u}_* = \vec{f} - B^T p_k$

solve $B M^{-1} B^T \phi = B\,\vec{u}_*$

compute $\vec{u}_{k+1} = \vec{u}_* - M^{-1} B^T \phi$

compute $p_{k+1} = p_k + (\gamma / dT)\,\phi$

end

A simple splitting/projection approach — II

Key Question: Is the projection/splitting consistent?

A simple splitting/projection approach — II

Key Question: Is the projection/splitting consistent?
Practictioners say yes ...

- Chorin (1967), Temam(1969), Kim & Moin (1985), Van Kan (1986), Bell, Colella & Glaz (1989), Gresho & Chan (1984, 1990), Perot (1993), Turek (1997).

A simple splitting/projection approach — II

Key Question: Is the projection/splitting consistent?
Practictioners say yes ...

- Chorin (1967), Temam(1969), Kim & Moin (1985), Van Kan (1986), Bell, Colella & Glaz (1989), Gresho & Chan (1984, 1990), Perot (1993), Turek (1997).

Mathematicians say sometimes ...

- Guermond (1984), Rannacher (1992), E & Liu (1995, 1996), Shen (1992, 1996), Prohl (1997, 2008).

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} N(\vec{u}) + \nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} \vec{f} \\ 0 \end{pmatrix}$$

- 1967
- 1982

A Stokes splitting/projection approach — I

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} N(\vec{u}) + \nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Given a time step dT, discretise the total derivative:

Algorithm: Pironneau, 1982; Hansbo, 1982

for $k = 0, 1, \ldots$

compute $X^m(\vec{x})$ the solution at $\tau_k = kdT$ of

$$\frac{dX}{d\tau} = \vec{u}_k(X, \tau), \quad X(\tau_{k+1}) = \vec{x};$$

interpolate $\vec{u}_k^* = \vec{u}_k(X^m(\vec{x}))$

solve
$$\begin{pmatrix} \frac{1}{dT} M + \nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u}_{k+1} \\ p_{k+1} \end{pmatrix} = \begin{pmatrix} \frac{1}{dT} M \, \vec{u}_k^* \\ 0 \end{pmatrix}$$

end

A simple splitting/projection approach — II

Key[1] Question: Is the total derivative approximation stable numerically ?

A simple splitting/projection approach — II

Key[1] Question: Is the total derivative approximation stable numerically ?
Mathematicians say yes, but care is needed ...

- Hansbo (1982), Pironneau (1982), Morton, Priestley & Suli (1988,1989, 1994), Bermejo & Staniforth (1991, 1992).

A simple splitting/projection approach — II

Key[1] Question: Is the total derivative approximation stable numerically ?
Mathematicians say yes, but care is needed ...

- Hansbo (1982), Pironneau (1982), Morton, Priestley & Suli (1988,1989, 1994), Bermejo & Staniforth (1991, 1992).

Key[2] Question: Adaptive time stepping ?

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} N(\vec{u}) + \nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} \vec{f} \\ 0 \end{pmatrix}$$

- 1967
- 1982
- 1991

A clever splitting/projection approach — I

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} N(\vec{u}) + \nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} \vec{f} \\ 0 \end{pmatrix}$$

Given dT and parameters $\theta = 1 - 1/\sqrt{2}$, $\alpha = \frac{1-2\theta}{1-\theta}$, $\beta = 1 - \alpha$.

Algorithm: Glowinski, 1986, 1991

for $k = 0, 1, \ldots$

solve $$\begin{pmatrix} \frac{1}{\theta dT} M + \alpha\nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u}_{k+\theta} \\ p_{k+\theta} \end{pmatrix} = \begin{pmatrix} f_k \\ 0 \end{pmatrix}$$

solve $$\left(\frac{1}{(1-2\theta)dT} M + N(\vec{u}_{k+1-\theta}) + \beta\nu A \right) \vec{u}_{k+1-\theta} = \vec{f}_{k+\theta}$$

solve $$\begin{pmatrix} \frac{1}{\theta dT} M + \alpha\nu A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u}_{k+1} \\ p_{k+1} \end{pmatrix} = \begin{pmatrix} f_{k+1-\theta} \\ 0 \end{pmatrix}$$

end

A clever splitting/projection approach — II

Key[1] Question: Is the algorithm over-complicated?

A clever splitting/projection approach — II

Key[1] Question: Is the algorithm over-complicated?
Mathematicians say no, ...

- Glowinski & Dean (1984, 1993) , Bristeau et al. (1985), Rannacher (1989, 1990), Turek (1996), Smith & Silvester (1997).

A clever splitting/projection approach — II

Key[1] Question: Is the algorithm over-complicated?
Mathematicians say no, ...

- Glowinski & Dean (1984, 1993) , Bristeau et al. (1985), Rannacher (1989, 1990), Turek (1996), Smith & Silvester (1997).

Key[2] Question: Adaptive time stepping ?

- Update : 2000 – 2011

- Philip Gresho & David Griffiths & David Silvester Adaptive time-stepping for incompressible flow; part I: scalar advection-diffusion, SIAM J. Scientific Computing, 30: 2018–2054, 2008.
- David Kay & Philip Gresho & David Griffiths & David Silvester Adaptive time-stepping for incompressible flow; part II: Navier-Stokes equations. SIAM J. Scientific Computing, 32: 111–128, 2010.

“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

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

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

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

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

with second-order linearization

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

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

Saddle-point system

In $\mathbb{R}^2$ the discretized Oseen system (*) is:

$$\begin{pmatrix} F^{n+1} & 0 & B_x^T \\ 0 & F^{n+1} & B_y^T \\ B_x & B_y & 0 \end{pmatrix} \begin{bmatrix} \alpha^{x,n+1} \\ \alpha^{y,n+1} \\ \alpha^{p,n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}^{x,n+1} \\ \mathbf{f}^{y,n+1} \\ \mathbf{f}^{p,n+1} \end{bmatrix}$$

- $F^{n+1} := \frac{2}{k_{n+1}} M + \nu A + N(\vec{w}_h^{n+1})$
- The vector $\mathbf{f}$ is constructed from the boundary data $\vec{g}^{n+1}$, the computed velocity $\vec{u}_h^n$ at the previous time level and the acceleration $\frac{\partial \vec{u}_h^n}{\partial t}$
- The system can be efficiently solved using "appropriately" preconditioned GMRES...

Preconditioned system

$$\begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} \mathcal{P}^{-1} \quad \mathcal{P} \begin{pmatrix} \alpha^u \\ \alpha^p \end{pmatrix} = \begin{pmatrix} \mathbf{f}^u \\ \mathbf{f}^p \end{pmatrix}$$

A perfect preconditioner is given by

$$\begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} \underbrace{\begin{pmatrix} F^{-1} & F^{-1}B^T S^{-1} \\ 0 & -S^{-1} \end{pmatrix}}_{\mathcal{P}^{-1}} = \begin{pmatrix} I & 0 \\ BF^{-1} & I \end{pmatrix}$$

with $F = \frac{2}{k_{n+1}} M + \nu A + N$ and $S = BF^{-1}B^T$.

For an efficient preconditioner we need to construct a sparse approximation to the “exact” Schur complement

$$S^{-1} = (BF^{-1}B^T)^{-1}$$

Two possible constructions ...

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":

$$(BF^{-1}B^T)^{-1} \approx (\underbrace{BM_*^{-1}B^T}_{\text{AMG}})^{-1}(BM_*^{-1}FM_*^{-1}B^T)(\underbrace{BM_*^{-1}B^T}_{\text{AMG}})^{-1}$$

Schur complement approximation – II

Introducing associated pressure matrices

$$
\begin{aligned}
M_p &\sim (\nabla\psi_i, \nabla\psi_j), && \text{mass} \\
A_p &\sim (\nabla\psi_i, \nabla\psi_j), && \text{diffusion} \\
N_p &\sim (\vec{w}_h \cdot \nabla\psi_i, \psi_j), && \text{convection} \\
F_p &= \frac{2}{k_{n+1}} M_p + \nu A_p + N_p, && \text{convection-diffusion}
\end{aligned}
$$

gives the “pressure convection-diffusion preconditioner”:

$$
(BF^{-1}B^T)^{-1} \approx M_p^{-1} F_p \underbrace{A_p^{-1}}_{\text{AMG}}
$$

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 `tol`, the new time step is selected to be the biggest possible such that $\|T_{n+1}\| \leq \texttt{tol} \times u_{\max}$. This criterion leads to

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

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 `tol`, the new time step is selected to be the biggest possible such that $\|T_{n+1}\| \leq \texttt{tol} \times u_{\max}$. This criterion leads to

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

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:

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

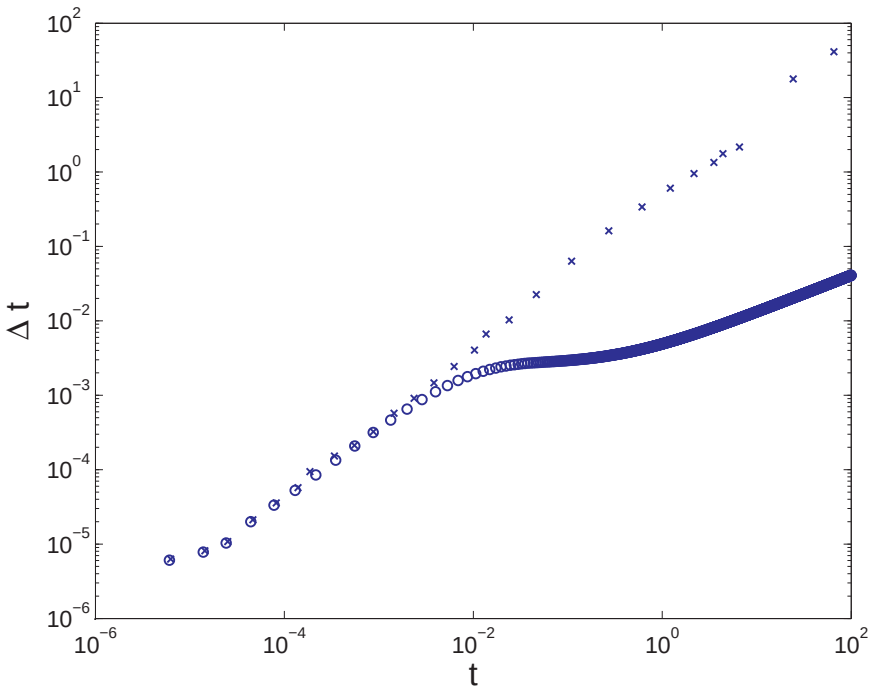
- We perform time-step averaging every n^* steps:

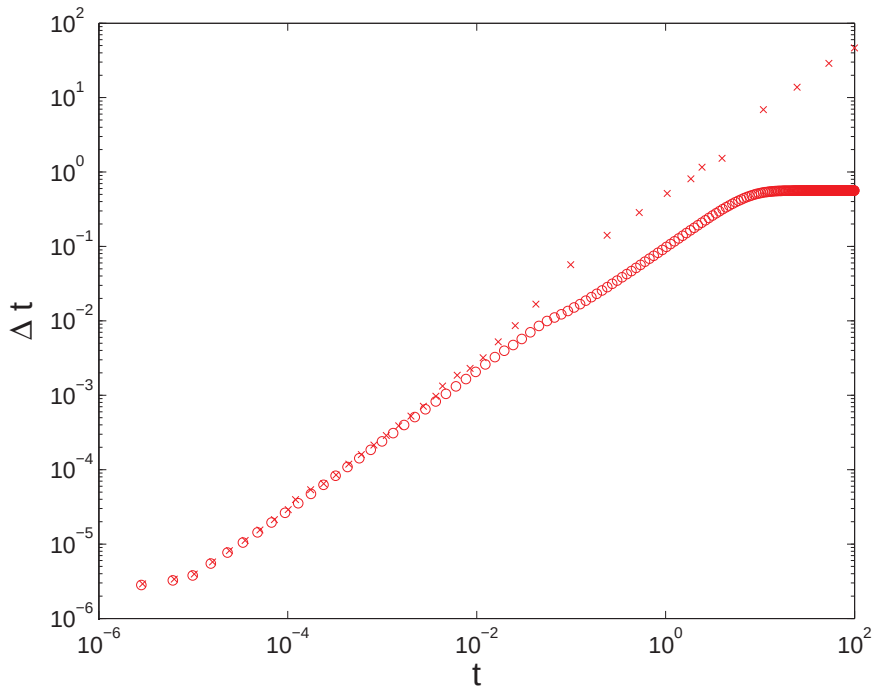
$$u_n := \tfrac{1}{2}(u_n + u_{n-1}); \quad u_{n+1} := u_n + \tfrac{1}{4} k_{n+1} v_n; \quad \dot{u}_{n+1} := \tfrac{1}{2} v_n.$$

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

$$\dot{u}_{n+1} = \frac{2}{k_{n+1}} (u_{n+1} - u_n) - \dot{u}_n = v_n - \dot{u}_n$$

Stabilized AB2–TR

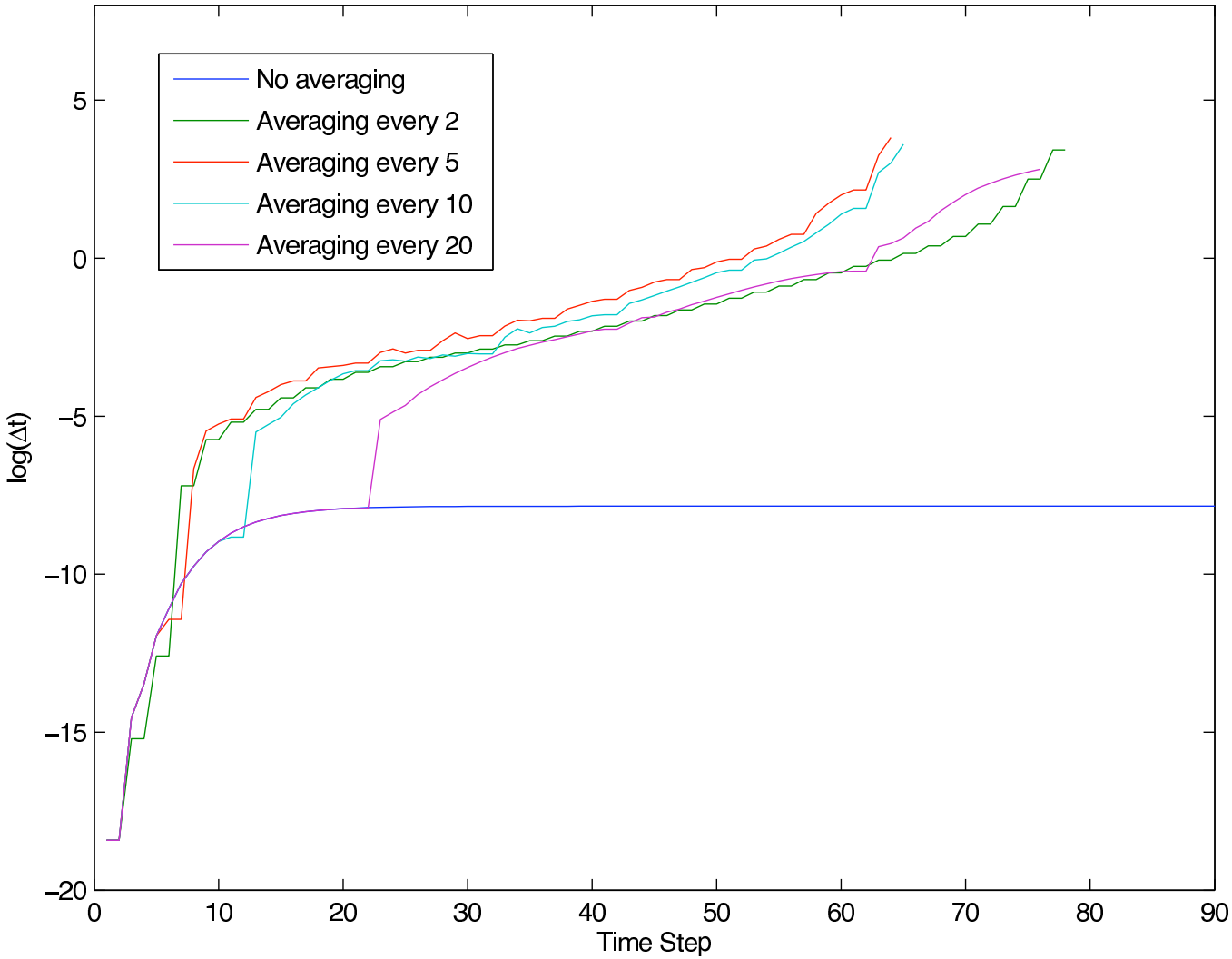




Advection-Diffusion of step profile on Shishkin grid.

`tol` $= 10^{-3}$ `tol` $= 10^{-4}$

Stabilized AB2–TR



"Spin up" driven cavity flow with $\nu = 1/100$.

Adaptive Time-Stepping Algorithm I

- The following parameters must be specified:

time accuracy tolerance	tol	(10^{-4})
GMRES tolerance	`itol`	(10^{-6})
GMRES iteration limit	`maxit`	(50)

Adaptive Time-Stepping Algorithm I

- The following parameters must be specified:

time accuracy tolerance	tol	(10^{-4})
GMRES tolerance	`itol`	(10^{-6})
GMRES iteration limit	`maxit`	(50)

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

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

Adaptive Time-Stepping Algorithm I

- The following parameters must be specified:

time accuracy tolerance	tol	(10^{-4})
GMRES tolerance	`itol`	(10^{-6})
GMRES iteration limit	`maxit`	(50)

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

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

- We specify the frequency of averaging, typically $n_* = 10$. We also choose a very small initial timestep, typically, $k_1 = 10^{-8}$.

Adaptive Time-Stepping Algorithm II

- Setup the Oseen System ($*$) and compute $[\alpha^{x,n+1}, \alpha^{y,n+1}]$ using GMRES(`maxit`, `itol`).

Adaptive Time-Stepping Algorithm II

- Setup the Oseen System $(*)$ and compute $[\alpha^{x,n+1}, \alpha^{y,n+1}]$ using GMRES(`maxit`, `itol`).
- Compute the LTE estimate $\mathbf{e}^{v,n+1}$

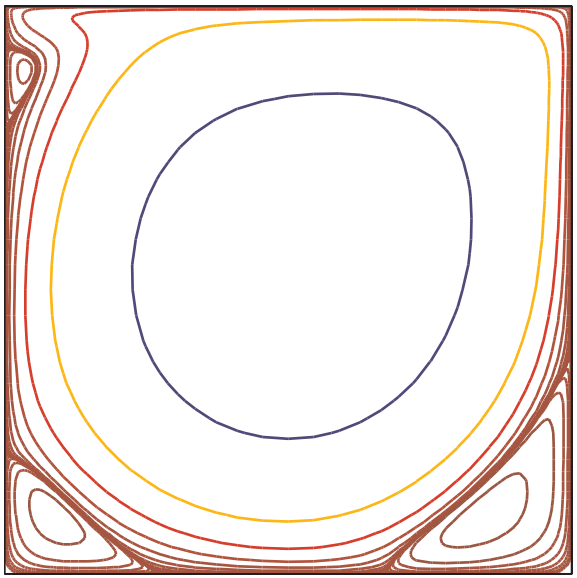
Adaptive Time-Stepping Algorithm II

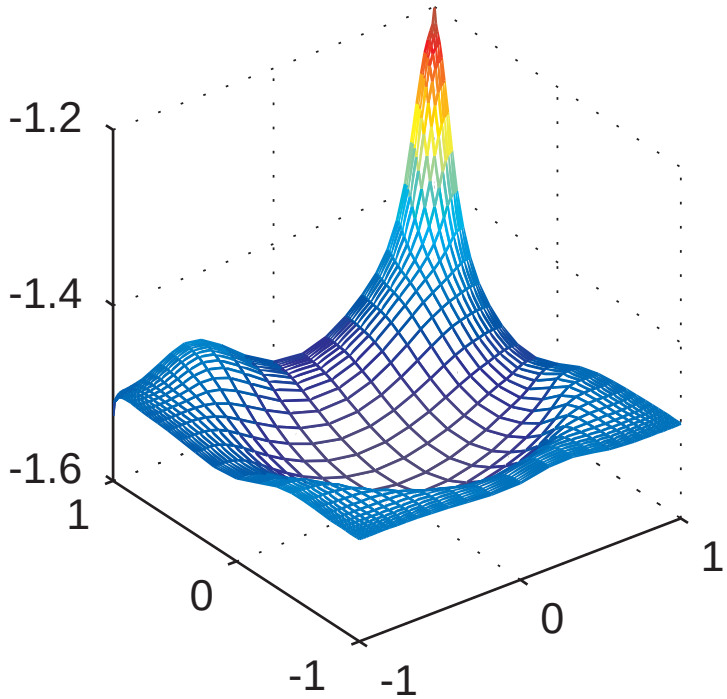
- Setup the Oseen System $(*)$ and compute $[\alpha^{x,n+1}, \alpha^{y,n+1}]$ using GMRES(`maxit`, `itol`).
- Compute the LTE estimate $\mathbf{e}^{v,n+1}$
- If $\|\mathbf{e}^{v,n+1}\| > (1/0.7)^3$tol, we reject the current time step, and repeat the old time step with $k_{n+1} = k_{n+1}(\frac{\text{tol}}{\|\mathbf{e}^{v,n+1}\|})^{1/3}$.

Adaptive Time-Stepping Algorithm II

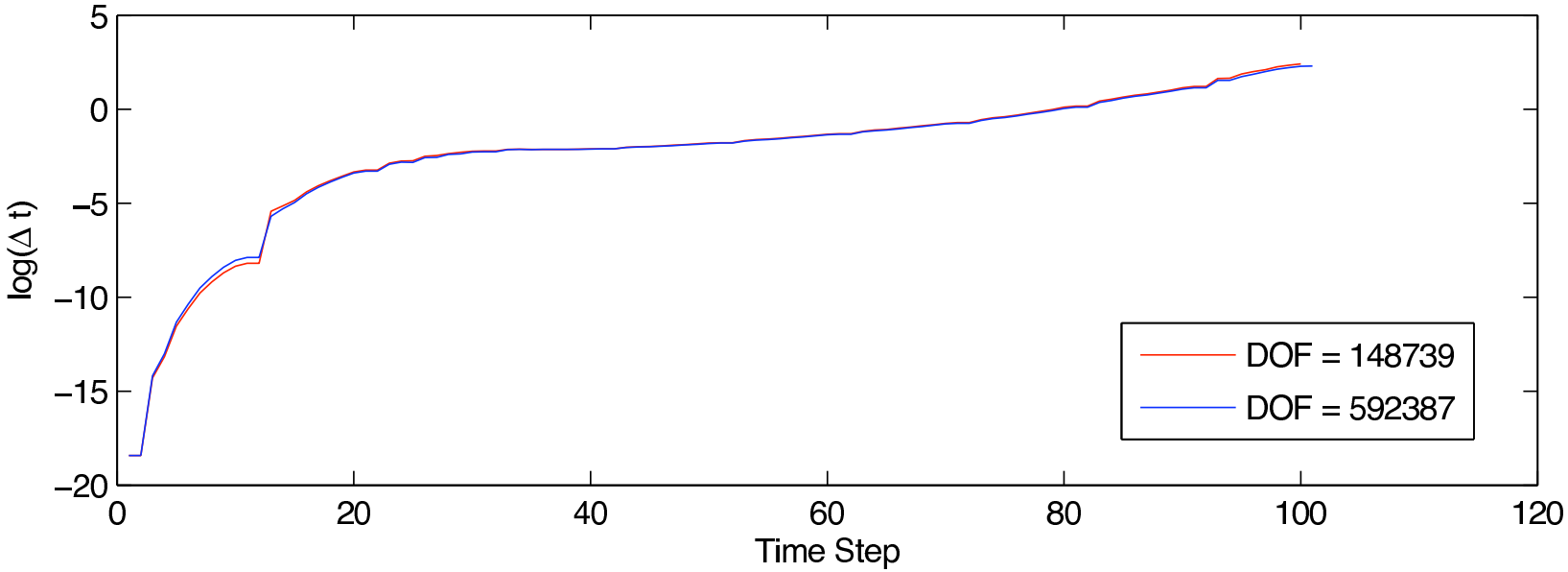
- Setup the Oseen System $(*)$ and compute $[\alpha^{x,n+1}, \alpha^{y,n+1}]$ using GMRES(`maxit`, `itol`).
- Compute the LTE estimate $\mathbf{e}^{v,n+1}$
- If $\|\mathbf{e}^{v,n+1}\| > (1/0.7)^3$tol, we reject the current time step, and repeat the old time step with $k_{n+1} = k_{n+1}(\frac{\text{tol}}{\|\mathbf{e}^{v,n+1}\|})^{1/3}$.
- Otherwise, accept the step and continue with $n = n + 1$ and k_{n+2} based on the LTE estimate and the accuracy tolerance tol.

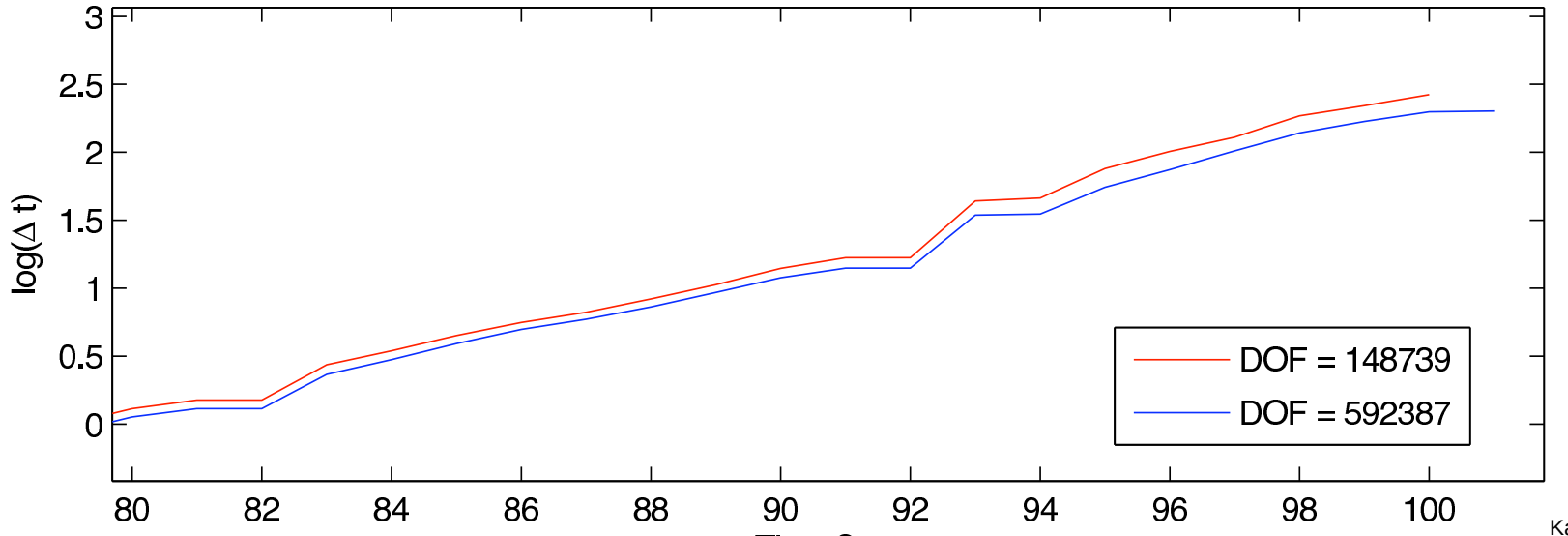
Example Flow Problem – I ($\nu = 1/1000$)



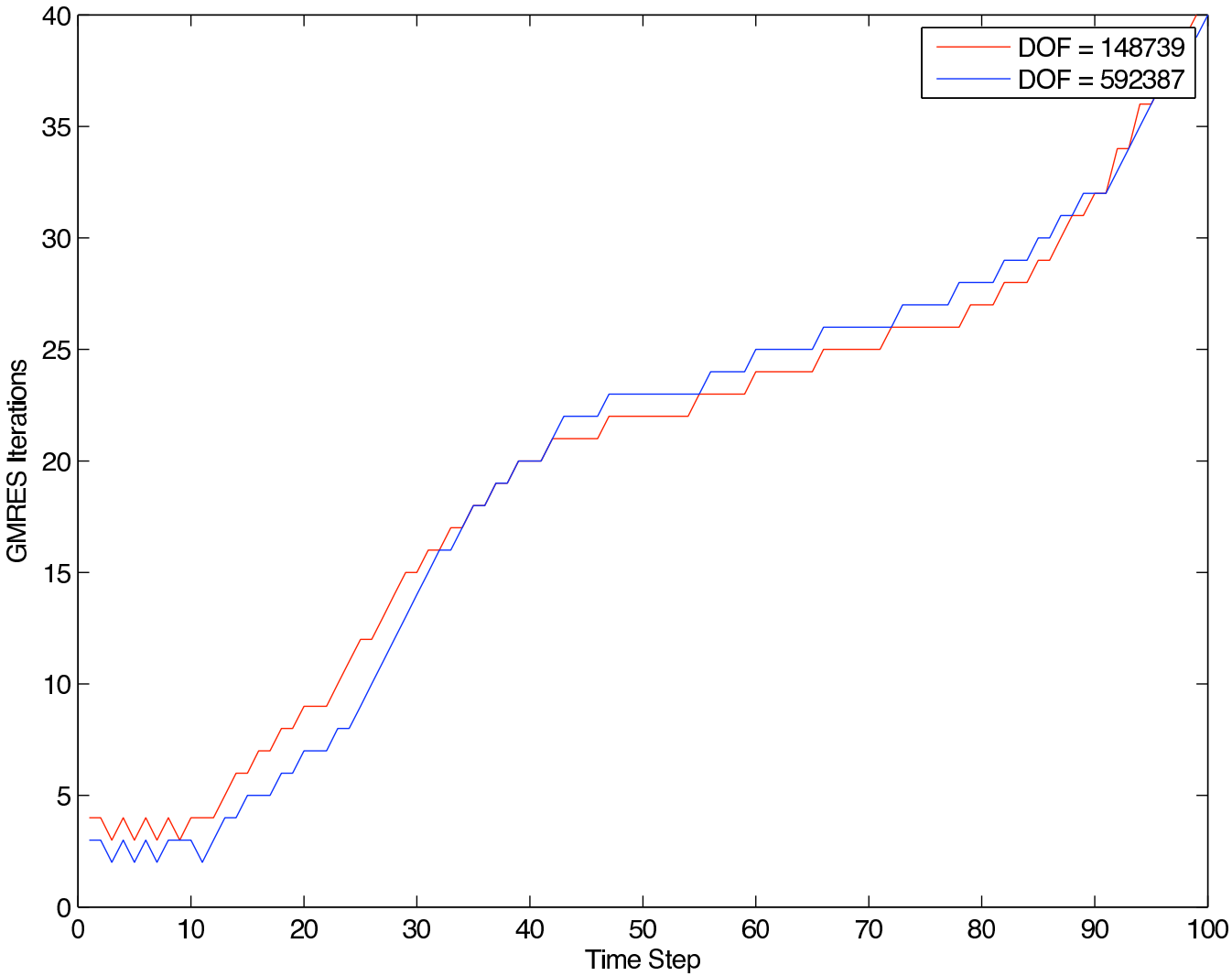


Time step evolution

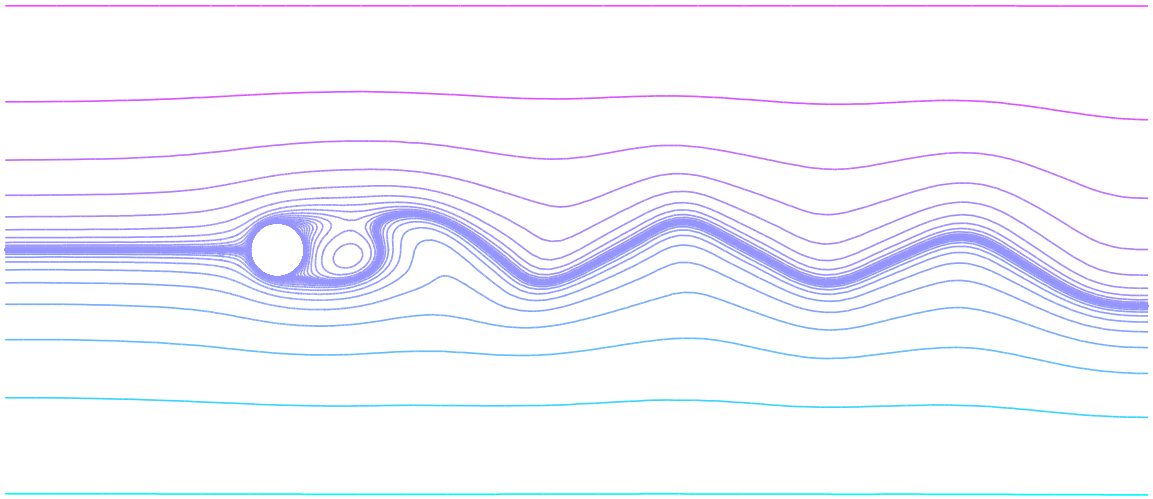


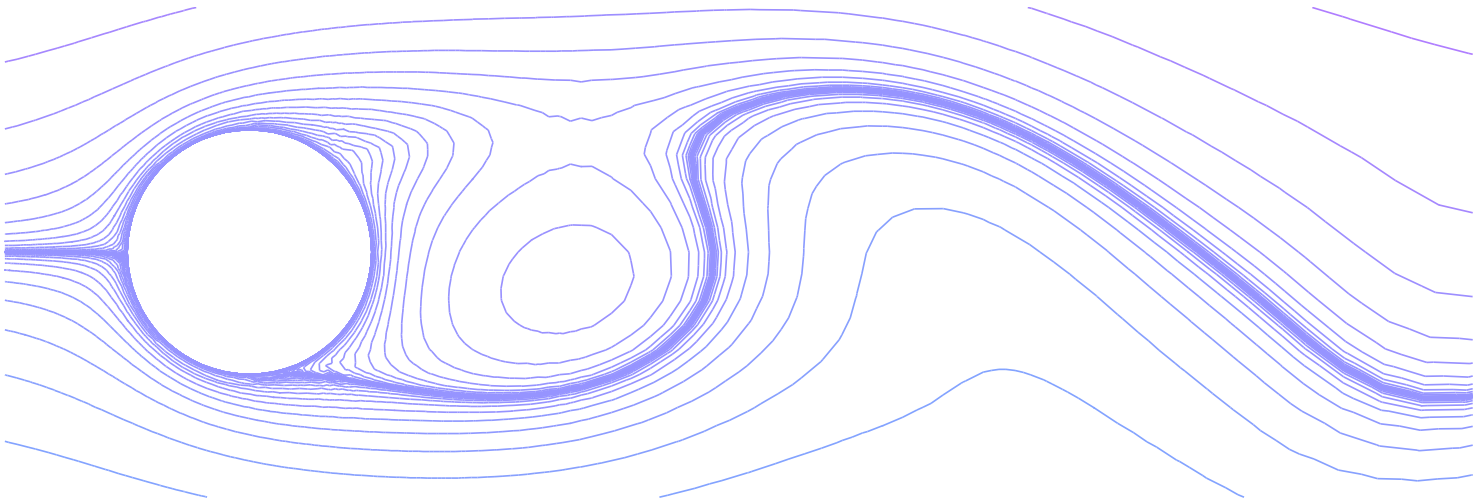


Linear solver performance

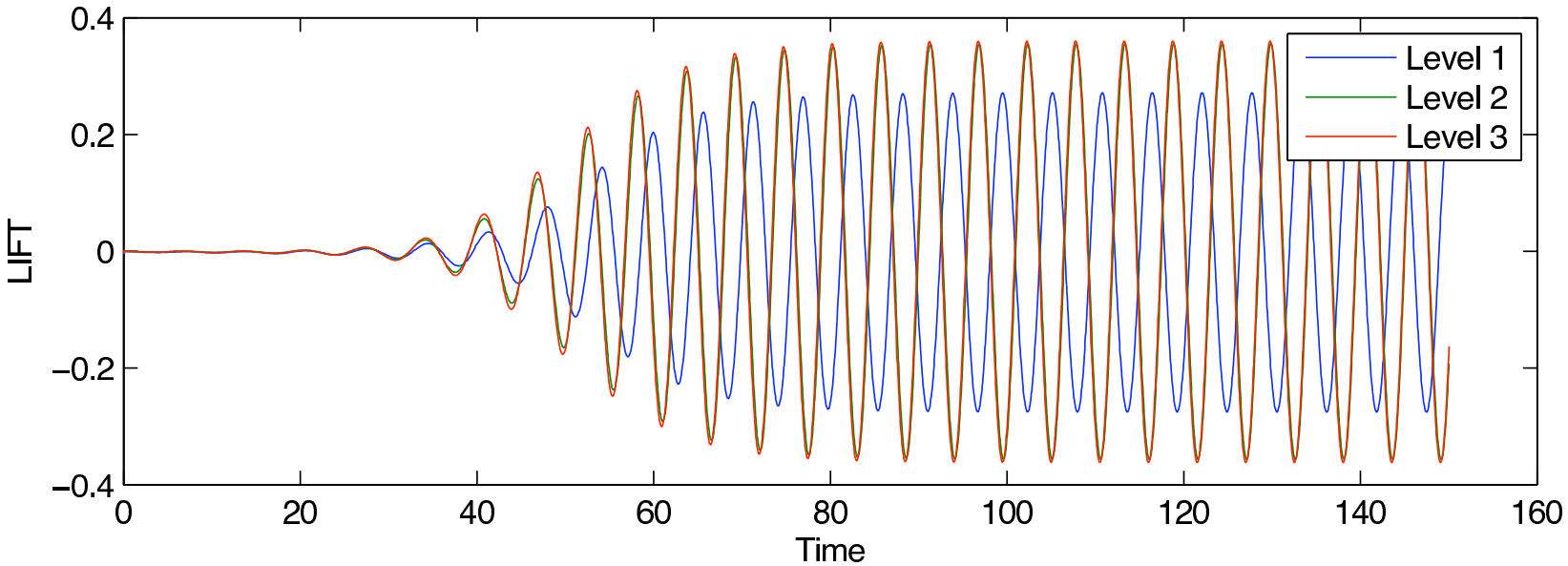


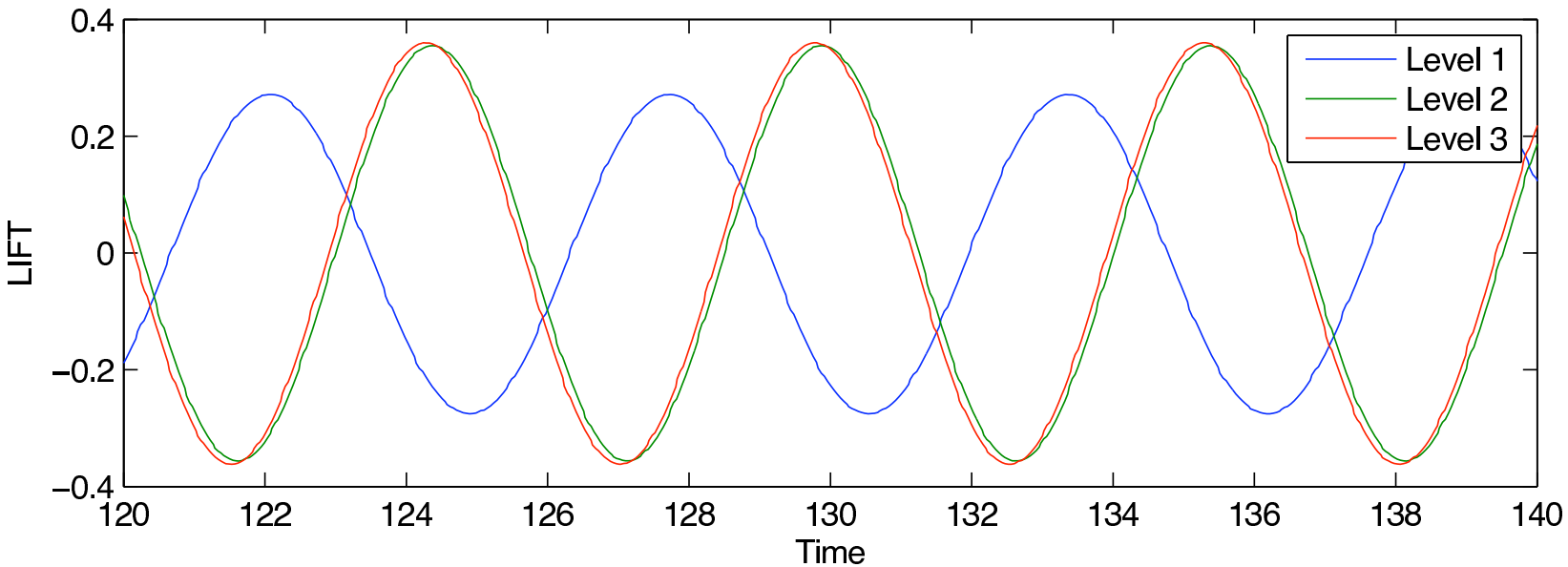
Example Flow Problem – II ($\nu = 1/100$)



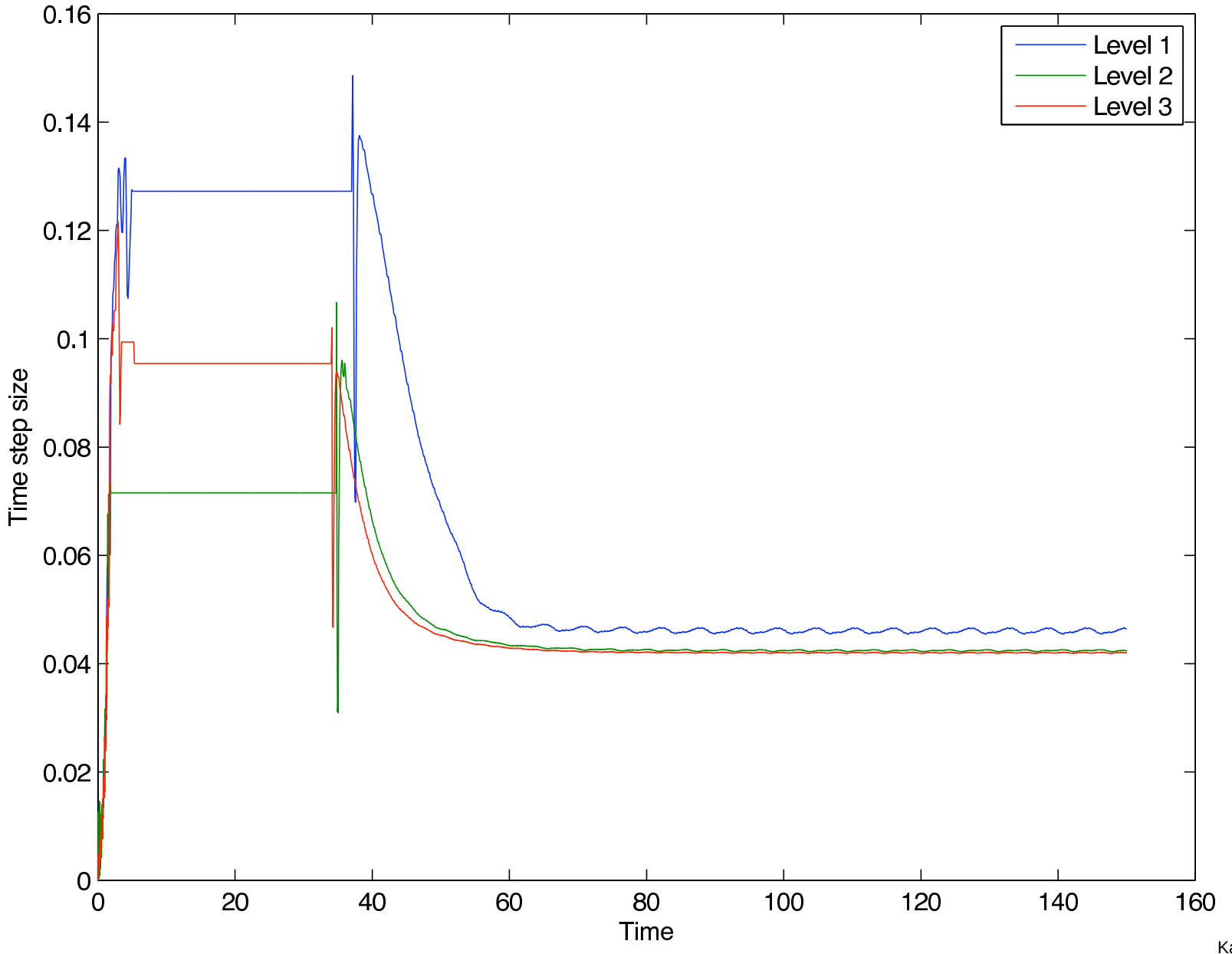


Lift Coefficient





Time step evolution



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 ν