MANCHESTER 1824

The University of Manchester

Black-Box Multigrid Preconditioning for Unsteady Incompressible Flows

David Silvester
University of Manchester
http://www.maths.manchester.ac.uk/~djs/

David Kay
Oxford University Computing Laboratory

- Joint work with
  - Phil Gresho (ex-LLNL)
  - David Griffiths (University of Dundee)

- Joint work with
  - Phil Gresho (ex-LLNL)
  - David Griffiths (University of Dundee)

For further details, see

- 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. MIMS Eprint 2008.61.

Outline

- Introduction: Gresho's "Smart Integrator" (SI)
- Part I: physical timescales:
  initial condition regularity: $\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0$
  wave speed conservation: $\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$
- Part II: Black-Box multigrid preconditioning:
  $\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} - \nu \nabla^2 \vec{u} + \nabla p = \vec{0}; \quad \nabla \cdot \vec{u} = 0$

- Introduction: Gresho's "Smart Integrator" (SI)

Time Integrator – I

For the simple ODE

$$\dot{u} = f(u)$$

we use the Trapezoidal Rule TR: $u_n \approx u(t_n)$ is computed so that for $n = 0, 1, \ldots$

$$u_{n+1} - u_n = \frac{1}{2}\Delta t_n \left(f_{n+1} + f_n\right)$$

Time Integrator – I

For the simple ODE

$$\dot{u} = f(u)$$

we use the Trapezoidal Rule TR: $u_n \approx u(t_n)$ is computed so that for $n = 0, 1, \ldots$

$$u_{n+1} - u_n = \tfrac{1}{2}\Delta t_n \left(f_{n+1} + f_n\right)$$

The Local Truncation Error LTE is

$$u_n - u(t_n) \equiv T_n = \tfrac{1}{12}\Delta t_n^3 \, \dddot{u}(\bar{t_n})$$

Time Integrator – II

For the simple ODE

$$\dot{u} = f(u)$$

we can estimate the LTE in TR using the explicit Adams-Bashforth method AB2: $u_n^* \approx u(t_n)$ is computed so that for $n = 1, 2, \ldots$

$$u_{n+1}^* - u_n^* = \Delta t_n \, f_n + \tfrac{1}{2}\Delta t_n^2 \left(\tfrac{f_n - f_{n-1}}{\Delta t_{n-1}}\right)$$

Time Integrator – II

For the simple ODE

$$\dot{u} = f(u)$$

we can estimate the LTE in TR using the explicit Adams-Bashforth method AB2: $u_n^* \approx u(t_n)$ is computed so that for $n = 1, 2, \ldots$

$$u_{n+1}^* - u_n^* = \Delta t_n \, f_n + \tfrac{1}{2}\Delta t_n^2 \left(\tfrac{f_n - f_{n-1}}{\Delta t_{n-1}}\right)$$

The Local Truncation Error LTE* is

$$u_n^* - u(t_n) \equiv T_n^* = -(2 + 3\tfrac{\Delta t_{n-1}}{\Delta t_n})\,\tfrac{1}{12}\Delta t_n^3 \,\dddot{u}(t_n^*)$$

Time Integrator – III

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

$$T_n = \frac{u_{n+1} - u^*_{n+1}}{3(1 + \frac{\Delta t_{n-1}}{\Delta t_n})}$$

Time Integrator – III

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

$$T_n = \frac{u_{n+1} - u^*_{n+1}}{3(1 + \frac{\Delta t_{n-1}}{\Delta t_n})}$$

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

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

Time Integrator – III

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

$$T_n = \frac{u_{n+1} - u^*_{n+1}}{3(1 + \frac{\Delta t_{n-1}}{\Delta t_n})}$$

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

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

- Implementation is “delicate”—it is very sensitive to round-off.

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:

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

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:

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

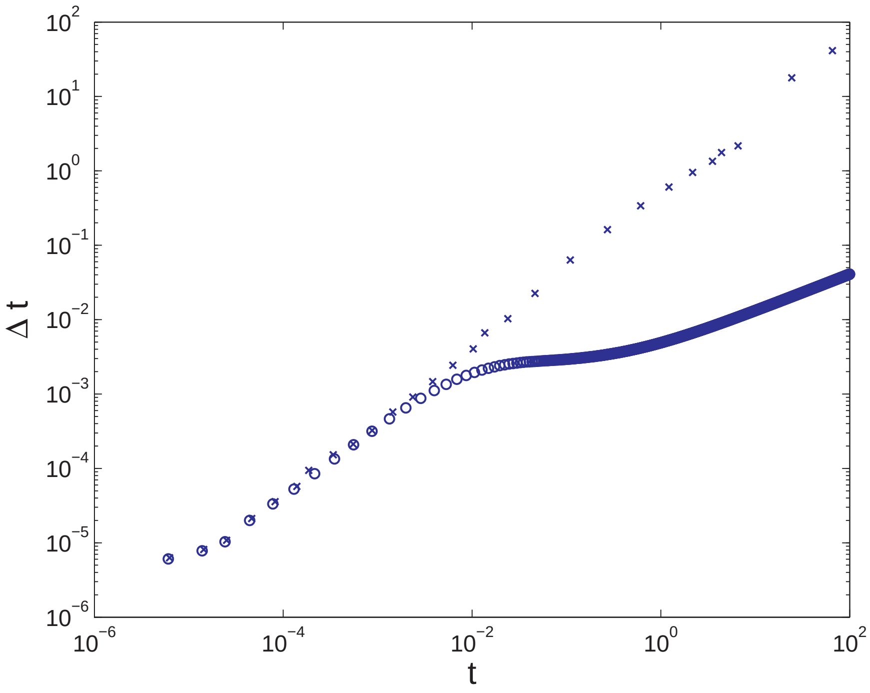
- We perform time-step averaging every n^* steps:

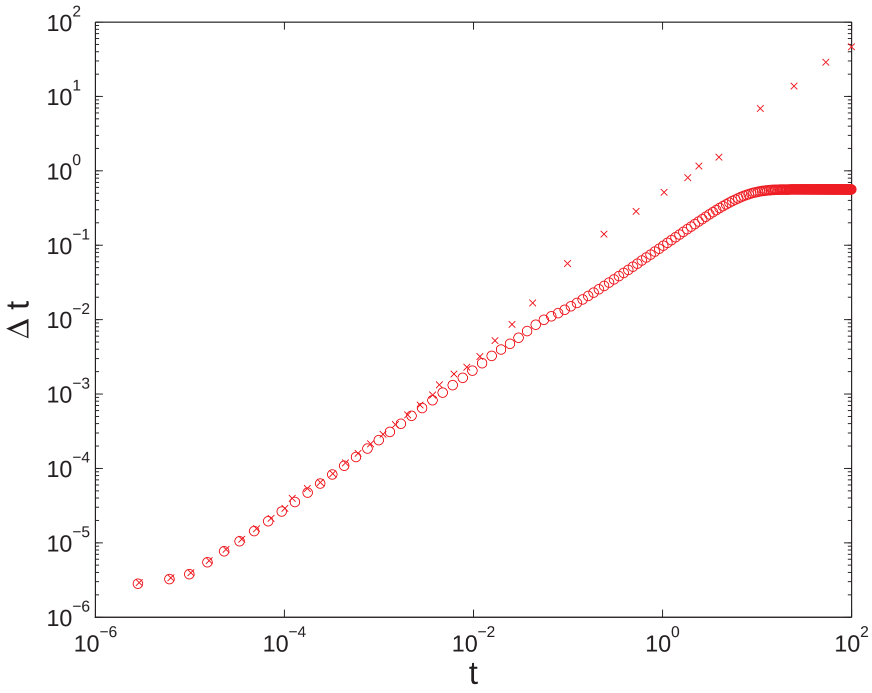
$$u_n := \tfrac{1}{2}(u_n + u_{n-1}); \quad u_{n+1} := u_n + \tfrac{1}{4}\Delta t_n v_n; \quad \dot{u}_{n+1} := \tfrac{1}{2} v_n.$$

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

$$\dot{u}_{n+1} = \frac{2}{\Delta t_n}\left(u_{n+1} - u_n\right) - \dot{u}_n$$

Stabilized AB2–TR





Advection-Diffusion of step profile on Shishkin grid.

$\texttt{tol} = 10^{-3}$ $\texttt{tol} = 10^{-4}$

- Introduction: Gresho's "Smart Integrator" (SI)
- Part I: physical timescales
  initial condition regularity: $\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0$

Heat Equation – I

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0, \qquad 0 < x < 1$$

$$u(0,t) = 1, \quad u(1,t) = 0 \qquad\qquad BC$$

$$u(x,0) = 1,\ 0 \le x < 1,\ u(1,0) = 0 \qquad IC$$

Solution.

$$u(x,t) = \begin{cases} \text{erf}\left(\frac{1-x}{\sqrt{4t}}\right) \\ (1-x)\ + \sum_{j=1}^{\infty} \frac{2}{j\pi} e^{-j^2\pi^2 t} \sin j\pi x \end{cases}$$

Heat Equation – II

Spatial Discretization

Using linear FEM gives the ODE system

$$M\dot{\mathbf{u}} + A\mathbf{u} = \mathbf{f}$$

with M and A both symmetric positive definite matrices.

Discrete solution.

$$\mathbf{u}(t) = (1 - x) \; + \sum_{k=1}^{n_u} a_k e^{-\lambda_k t} \mathbf{v}_k$$

where $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n_u}$ and $\{\lambda_k, \mathbf{v}_k\}$ satisfy

$$M\mathbf{v}_k = \lambda_k A\mathbf{v}_k.$$

Heat Equation – III

$$\mathbf{u}(t) = (1-x) \, + \sum_{k=1}^{n_u} a_k e^{-\lambda_k t} \mathbf{v}_k$$

... suggests two asymptotic extremes ...

- For $t < \frac{1}{\lambda_{n_u}} =: \tau_{\mathtt{mtb}}$ there is a fast transient:
  $\mathbf{u}(t) \sim a_{n_u} e^{-\lambda_{n_u} t} \mathbf{v}_{n_u} +$ slowly varying terms
- For $t \gg 1$ there is a slow transient:
  $\mathbf{u}(t) \sim (1-x) \, + a_1 e^{-\lambda_1 t} \mathbf{v}_1$

Heat Equation – III

$$\mathbf{u}(t) = (1-x)\ + \sum_{k=1}^{n_u} a_k e^{-\lambda_k t}\mathbf{v}_k$$

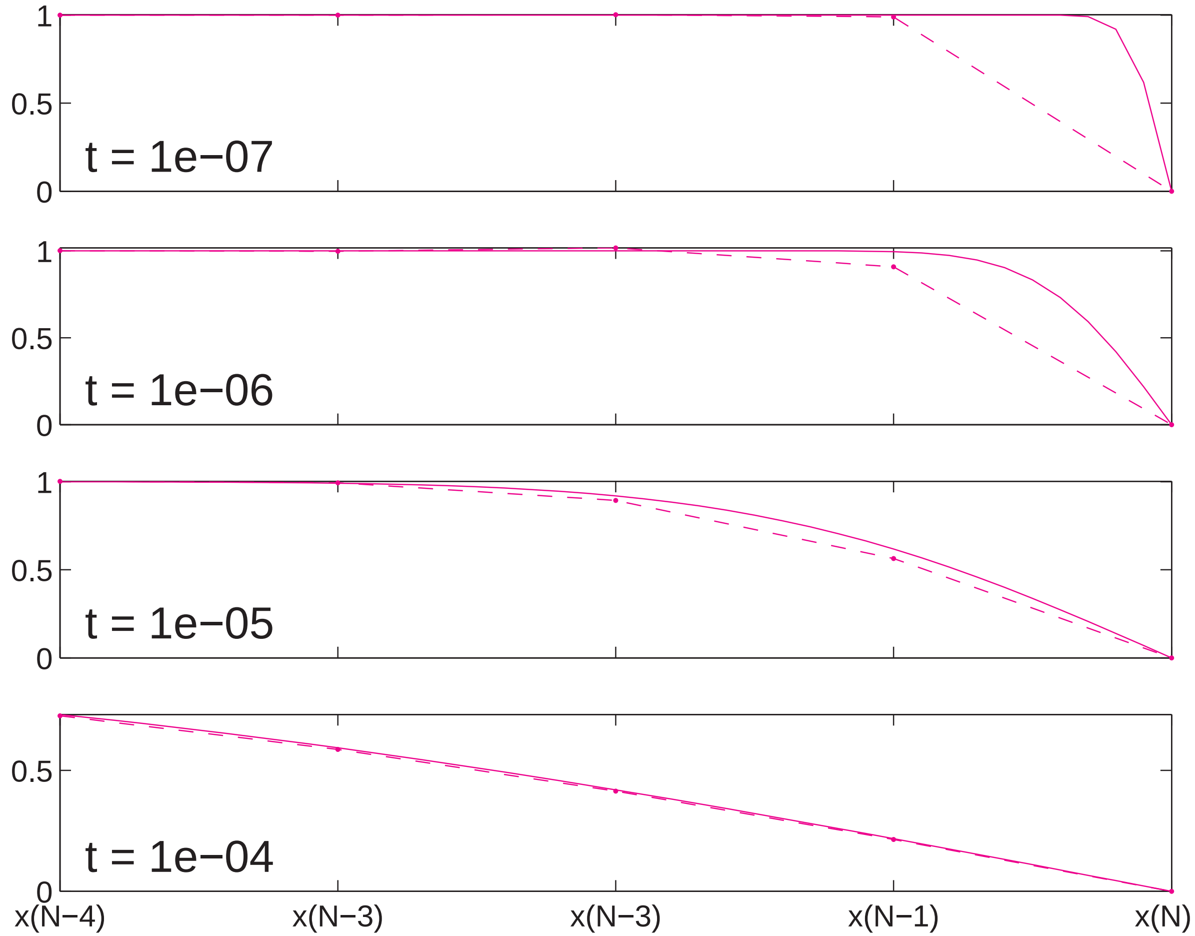
... suggests two asymptotic extremes ...

- For $t < \frac{1}{\lambda_{n_u}} =: \tau_{\texttt{mtb}}$ there is a fast transient:
  $\mathbf{u}(t) \sim a_{n_u} e^{-\lambda_{n_u} t}\mathbf{v}_{n_u} +$ slowly varying terms
- For $t \gg 1$ there is a slow transient:
  $\mathbf{u}(t) \sim (1-x)\ + a_1 e^{-\lambda_1 t}\mathbf{v}_1$

$\tau_{\texttt{mtb}} \approx \frac{h^2}{4}$ is the “Minimum Time of Believability” for spatially discretized convection-diffusion problems—it is the time for discontinuities in IC to grow to size h.

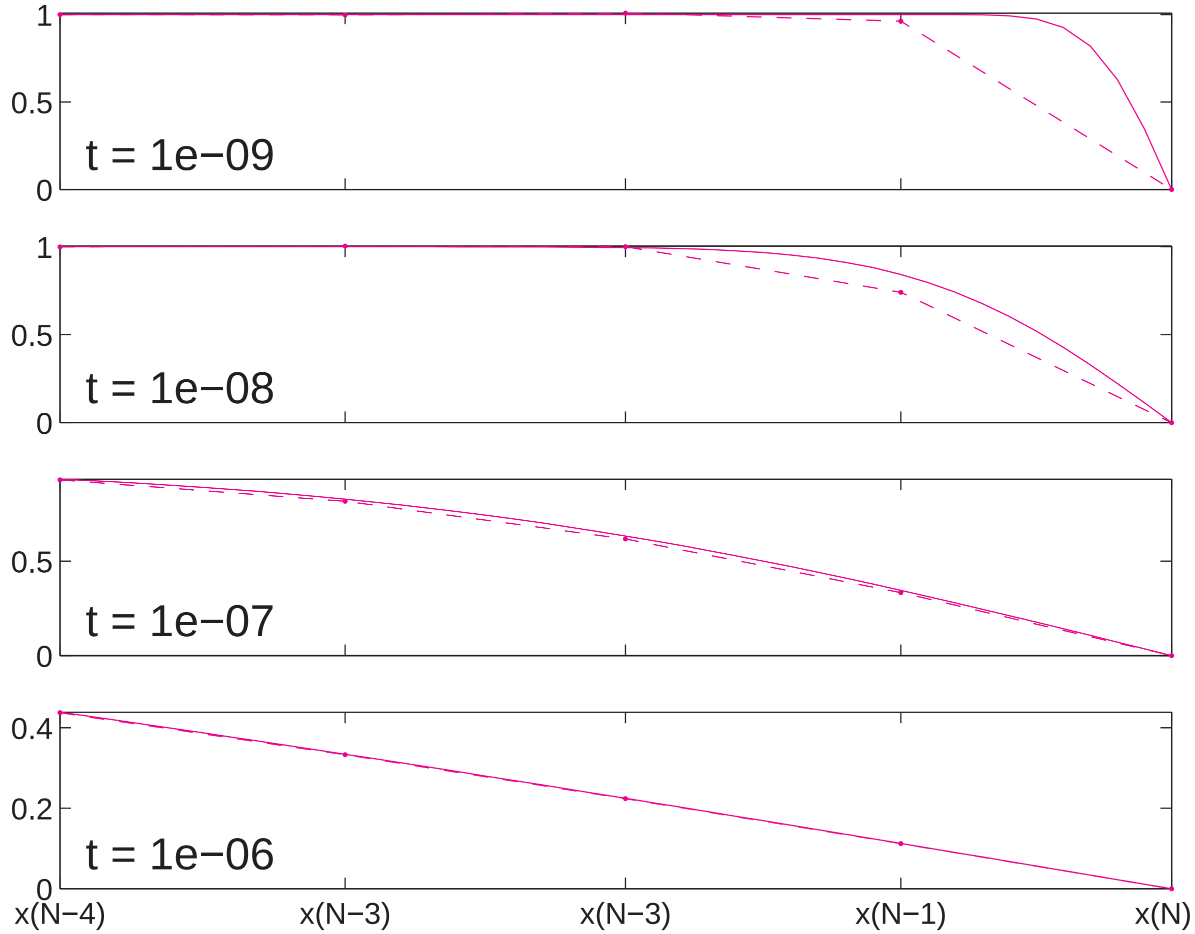
Spatial discretization I

- Uniform: $n_u = 255$, $h = 1/256$, $\tau_{\mathtt{mtb}} \sim 4 \times 10^{-6}$



Spatial discretization II

- Geometric: $h_{\min} = 2 \times 10^{-4}, n_u = 255, \tau_{\mathtt{mtb}} \sim 10^{-8}$



Heat Equation – IV

$$\mathbf{u}(t) = (1-x)\ + \sum_{k=1}^{n_u} a_k e^{-\lambda_k t}\mathbf{v}_k; \qquad \Delta t_n^3 = \frac{12\texttt{tol}}{\|\ddot{\mathbf{u}}\|}$$

- For $t < \tau_{\texttt{mtb}}$ there is a fast transient:
  $\mathbf{u}(t) \sim a_{n_u} e^{-\lambda_{n_u} t}\mathbf{v}_{n_u} +$ slowly varying terms
  $\Delta t_n \sim e^{\lambda_{n_u} t/3}$

- For $t \gg 1$ there is a slow transient:
  $\mathbf{u}(t) \sim (1-x)\ + a_1 e^{-\lambda_1 t}\mathbf{v}_1$
  $\Delta t_n \sim e^{\lambda_1 t/3}$

Heat Equation – IV

$$\mathbf{u}(t) = (1-x) \; + \sum_{k=1}^{n_u} a_k e^{-\lambda_k t}\mathbf{v}_k; \qquad \Delta t_n^3 = \frac{12\texttt{tol}}{\|\dddot{\mathbf{u}}\|}$$

- For $t < \tau_{\texttt{mtb}}$ there is a fast transient:
  $\mathbf{u}(t) \sim a_{n_u} e^{-\lambda_{n_u} t}\mathbf{v}_{n_u} +$ slowly varying terms
  $\Delta t_n \sim e^{\lambda_{n_u} t/3}$
- What happens in between?
- For $t \gg 1$ there is a slow transient:
  $\mathbf{u}(t) \sim (1-x) \; + a_1 e^{-\lambda_1 t}\mathbf{v}_1$
  $\Delta t_n \sim e^{\lambda_1 t/3}$

Heat Equation – V

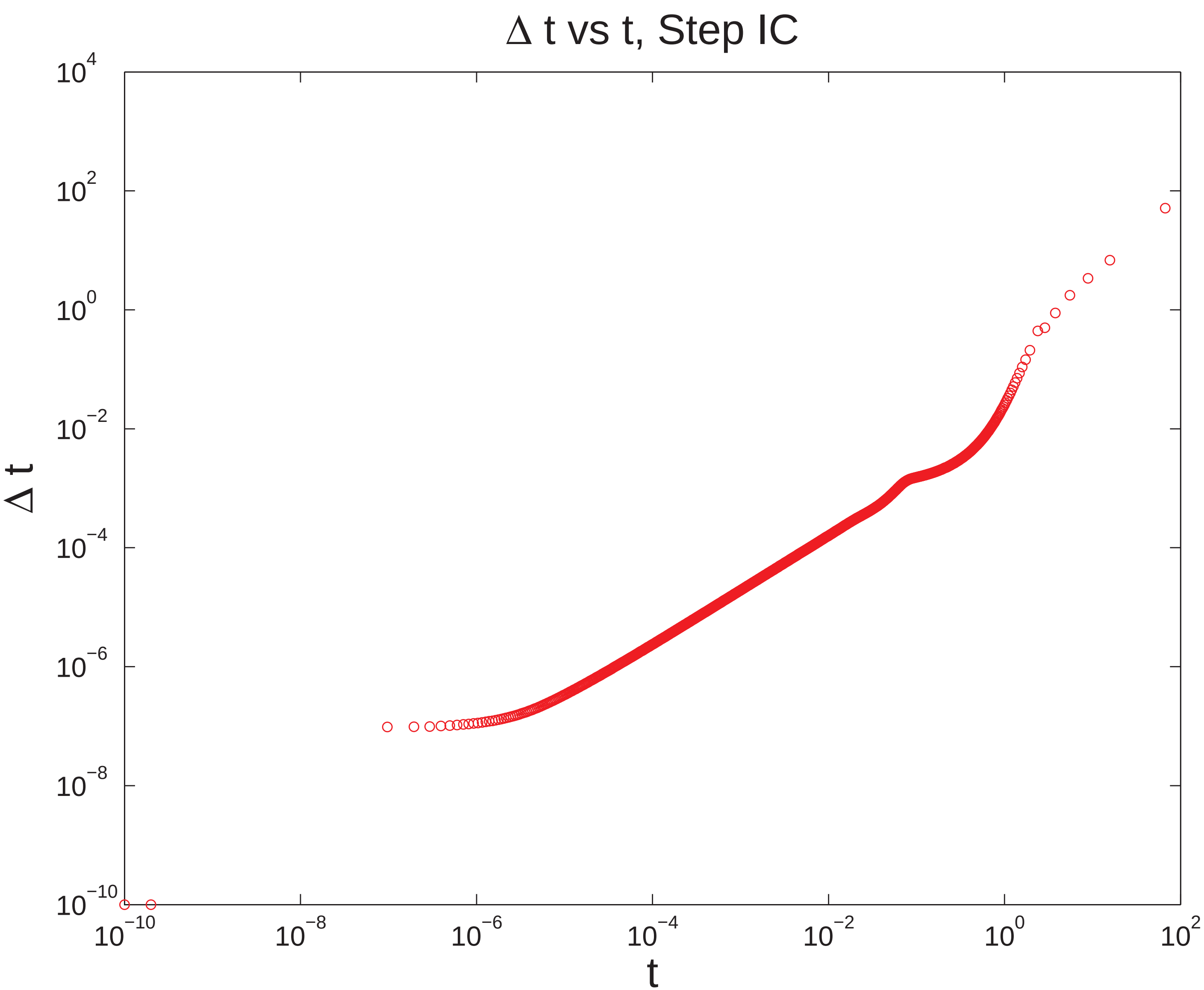
$$u(t) = (1-x) \; + \sum_{j=1}^{\infty} a_j e^{-j^2\pi^2 t} \sin j\pi x$$

Parabolic smoothing (Luskin & Rannacher)

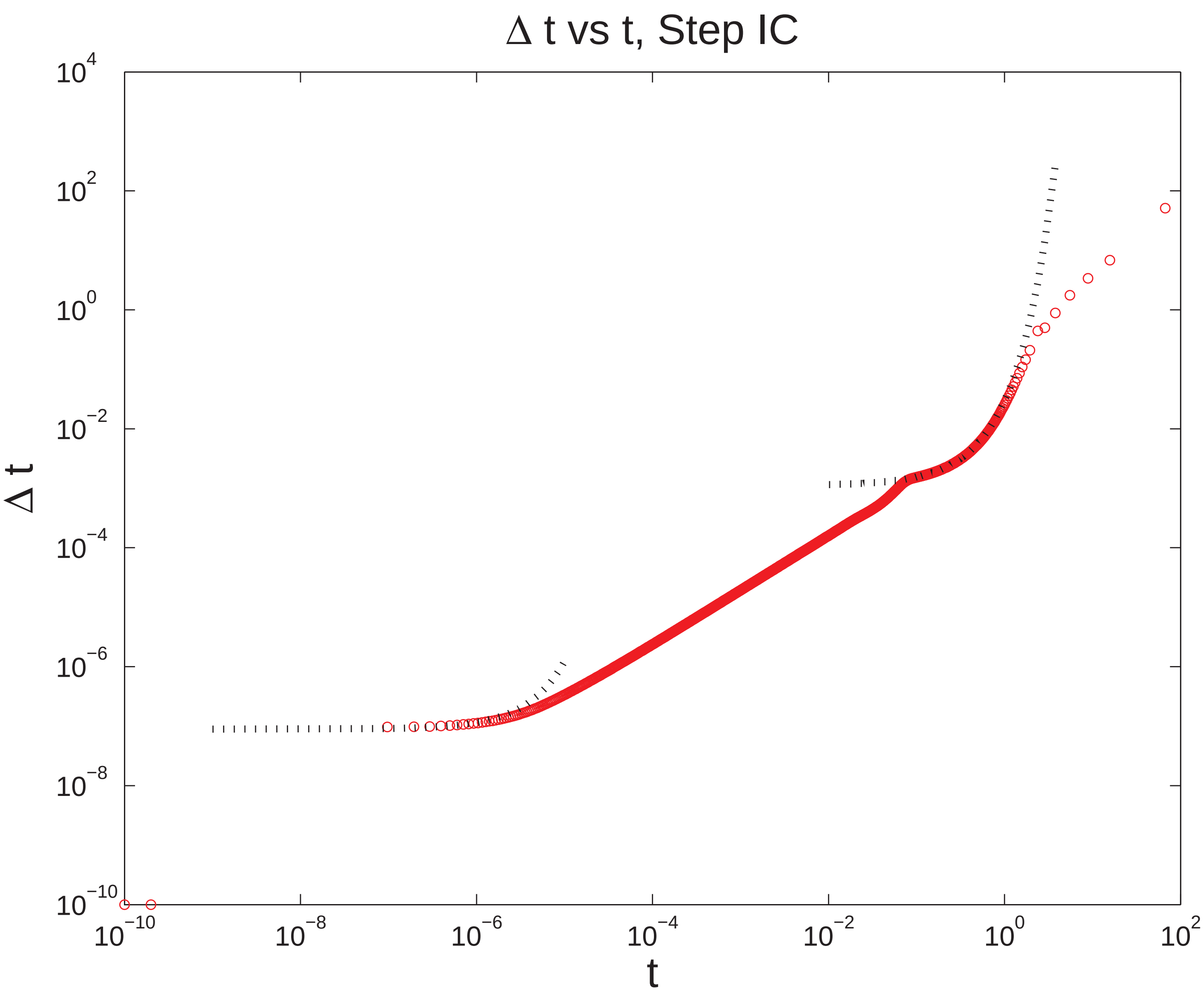
$$\begin{aligned}
\|\ddot{\mathbf{u}}\|^2 &\leq C \|\dddot{u}\|^2 \\
&= C \sum_{j=1}^{\infty} j^6 a_j^2 e^{-2j^2\pi^2 t} \\
&\leq C \max_j (j^{7+\epsilon} a_j^2 e^{-2j^2\pi^2 t}) \sum_{j=1}^{\infty} \frac{1}{j^{1+\epsilon}} \leq \frac{C}{t^{11/2}}
\end{aligned}$$

This gives the lower bound: $\Delta t_n \geq C t^{11/12}$

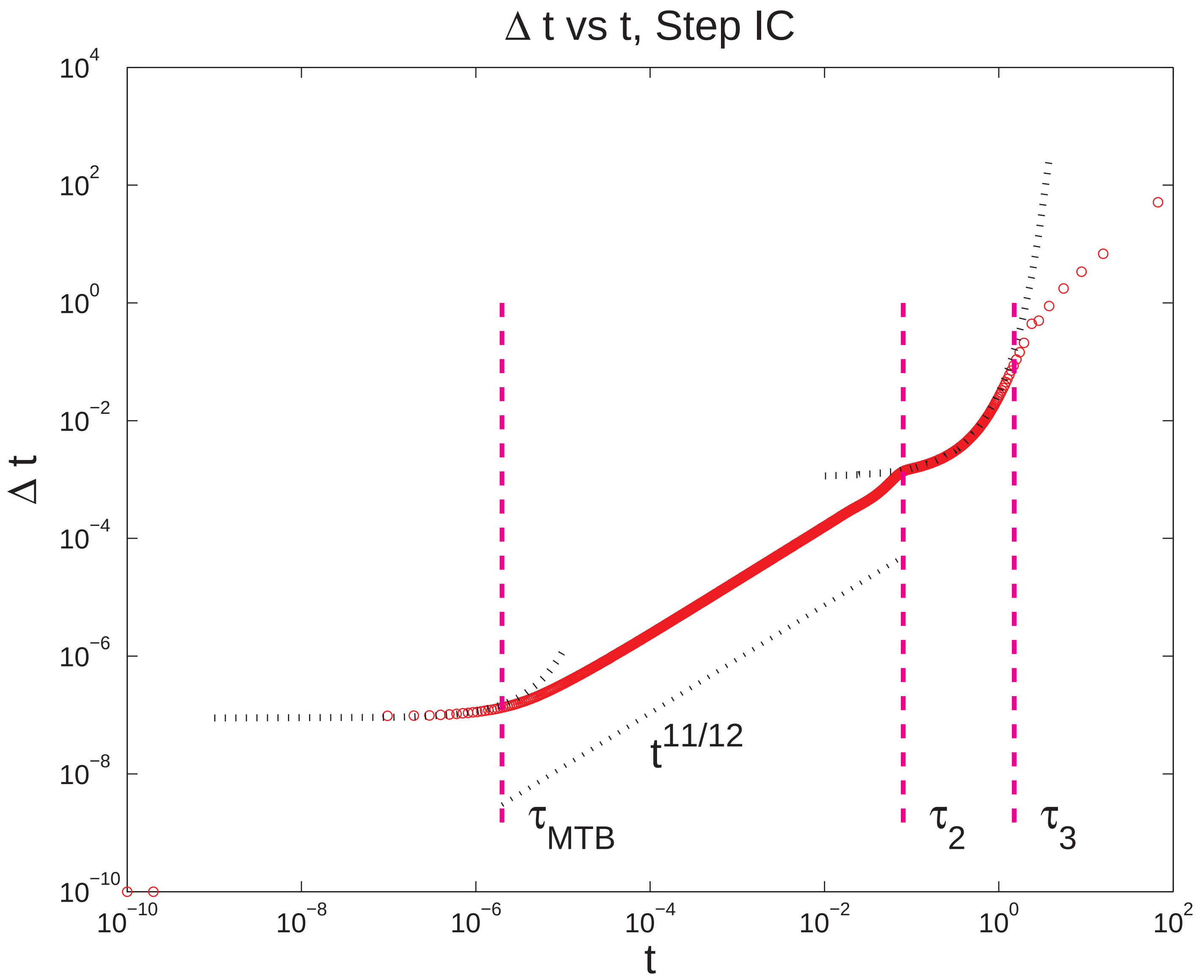
Uniform grid – Time steps



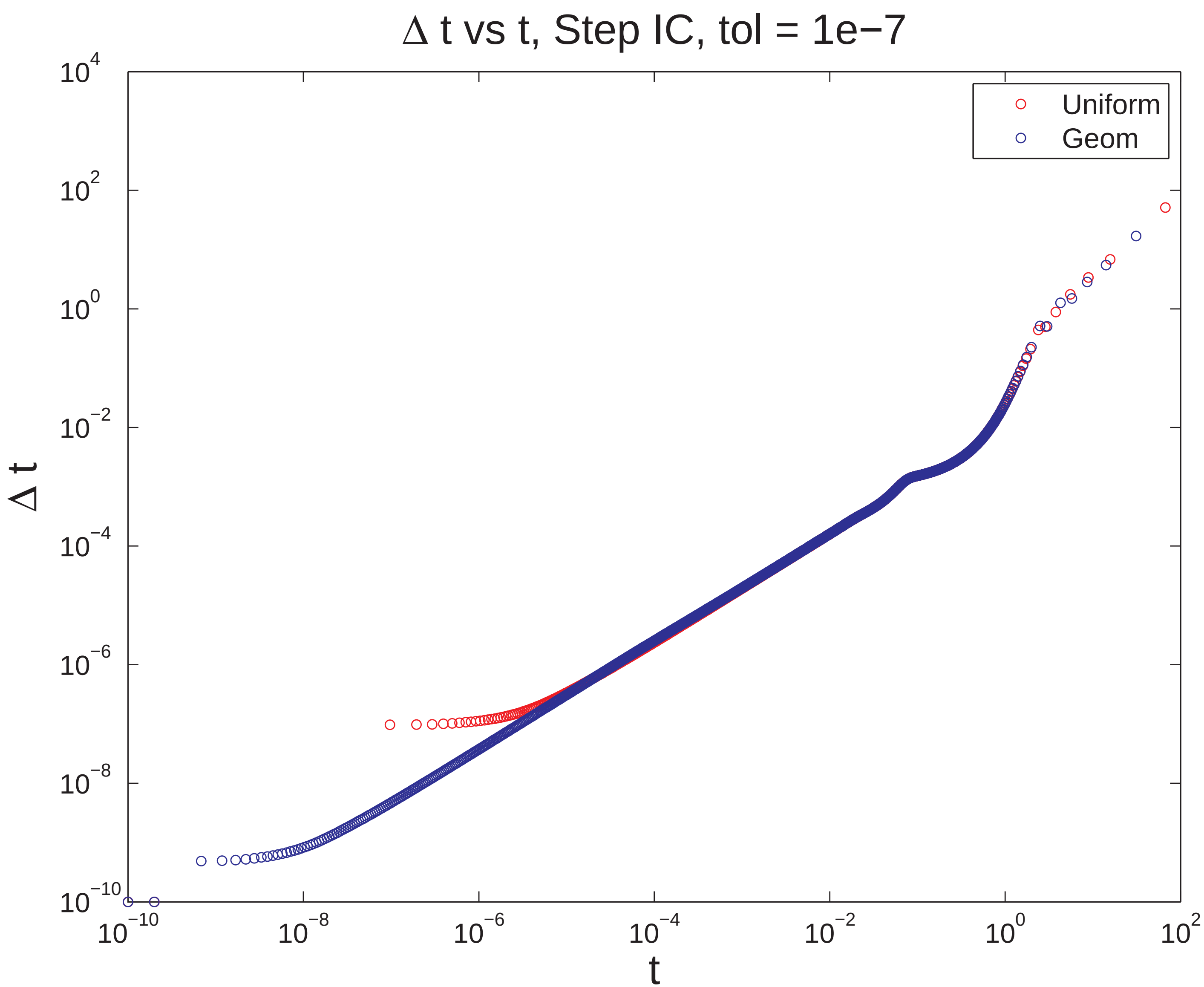
Uniform grid – Time steps



Uniform grid – Time steps



Uniform vs Geometric grid



- Introduction: Gresho's "Smart Integrator" (SI)
- Part I: physical timescales:
- Part II: Black-Box multigrid preconditioning:
  $\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} - \nu \nabla^2 \vec{u} + \nabla p = \vec{0}; \quad \nabla \cdot \vec{u} = 0$

Navier-Stokes Equations– I

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

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

Boundary and Initial conditions

$$\begin{aligned} \vec{u} &= \vec{g}_D \quad \text{on } \Gamma_D \times [0, T]; \\ \nu \nabla \vec{u} \cdot \vec{n} - p \vec{n} &= \vec{0} \quad \text{on } \Gamma_N \times [0, T]; \\ \vec{u}(\vec{x}, 0) &= \vec{u}_0(\vec{x}) \quad \text{in } \Omega. \end{aligned}$$

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

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

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

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

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

with linearization

$$\vec{f}^{n+1} = \frac{2}{k_{n+1}}\vec{u}^n + \frac{\partial\vec{u}^n}{\partial t} + (\vec{u}^n\cdot\nabla\vec{u}^n - \vec{w}^{n+1/2}\cdot\nabla\vec{u}^n)$$

$$\vec{w}^{n+1/2} = (1 + \frac{k_{n+1}}{k_n})\vec{u}^n - \frac{k_{n+1}}{k_n}\vec{u}^{n-1}$$

“Smart Integrator” (SI) definition

- Optimal time-stepping: time-steps automatically chosen to “follow the physics”.
- Black-box implementation: few parameters that have to be estimated a priori.

“Smart Integrator” (SI) definition

- Optimal time-stepping: time-steps automatically chosen to “follow the physics”.
- Black-box implementation: few parameters that have to be estimated a priori.
- Solver efficiency: the linear solver convergence rate is bounded independently of the discrete problem parameters.

Saddle-point system

The Oseen system (*) is:

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

- $F_v^{n+1} := \frac{2}{k_{n+1}} M_v + \nu A_v + N_v(\vec{w}_h^{n+1/2})$
- The timestep k_{n+1} is computed adaptively
- The vector f is constructed from the boundary data $\vec{g}_D^{n+1}$, 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

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

Preconditioned system

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

A perfect preconditioner is given by

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

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

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

$$S^{-1} = (B\mathcal{F}^{-1}B^T)^{-1}$$

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

$$S^{-1} = (B\mathcal{F}^{-1}B^T)^{-1}$$

See Chapter 8 of

- Howard Elman & David Silvester & Andrew Wathen
  Finite Elements and Fast Iterative Solvers: with applications in incompressible fluid dynamics
  Oxford University Press, 2005.

Two possible constructions ...

Schur complement approximation – I

Introducing the diagonal of the velocity mass matrix

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

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":

$$(B\mathcal{F}^{-1}B^T)^{-1} \approx \underbrace{(BM_*^{-1}B^T)}_{amg}{}^{-1}(BM_*^{-1}\mathcal{F}M_*^{-1}B^T)\underbrace{(BM_*^{-1}B^T)}_{amg}{}^{-1}$$

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”:

$$(B\mathcal{F}^{-1}B^T)^{-1} \approx Q^{-1} F_p \underbrace{A_p^{-1}}_{amg}$$

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 $\vec{u}(\vec{x}, t) = \vec{g}_D$, we model the impulse with a time-dependent boundary condition:

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

- 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 $[\boldsymbol{\alpha}^{x,n+1}, \boldsymbol{\alpha}^{y,n+1}]$ using GMRES(`maxit`, `itol`).

Adaptive Time-Stepping Algorithm II

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

Adaptive Time-Stepping Algorithm II

- Setup the Oseen System $(*)$ and compute $[\boldsymbol{\alpha}^{x,n+1}, \boldsymbol{\alpha}^{y,n+1}]$ 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

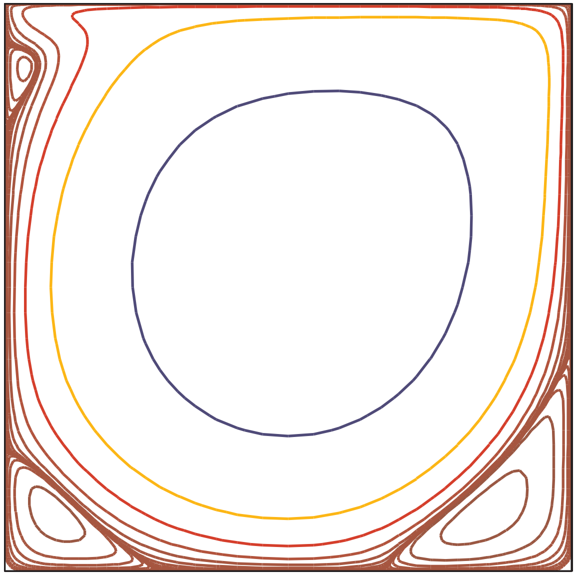
- Setup the Oseen System ($*$) and compute $[\boldsymbol{\alpha}^{x,n+1}, \boldsymbol{\alpha}^{y,n+1}]$ using GMRES(`maxit`, `itol`).

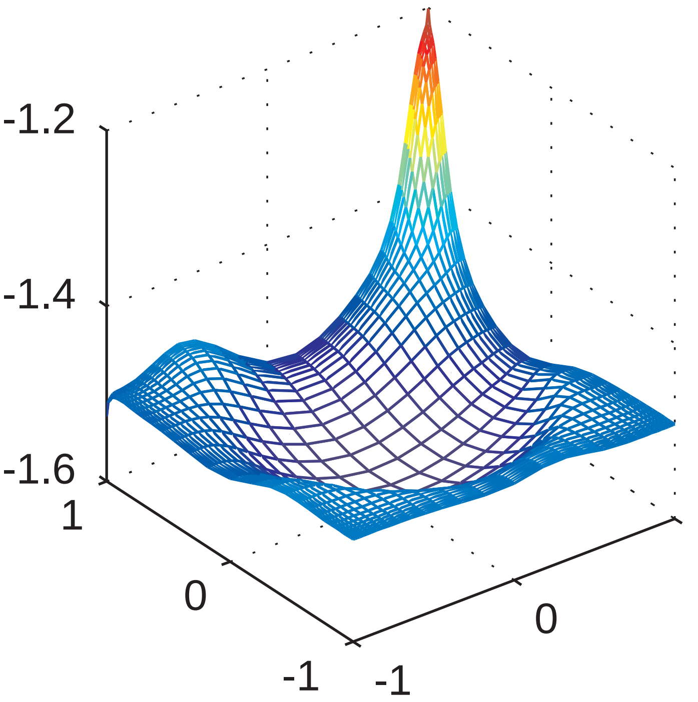
- Compute the LTE estimate $\mathbf{e}^{v,n+1}$

- If $\|\mathbf{e}^{v,n+1}\| > (1/0.7)^3$tol, we reject the current time step, and repeat the old time step with $k_{n+1} = k_{n+1}(\frac{\text{tol}}{\|\mathbf{e}^{v,n+1}\|})^{1/3}$.

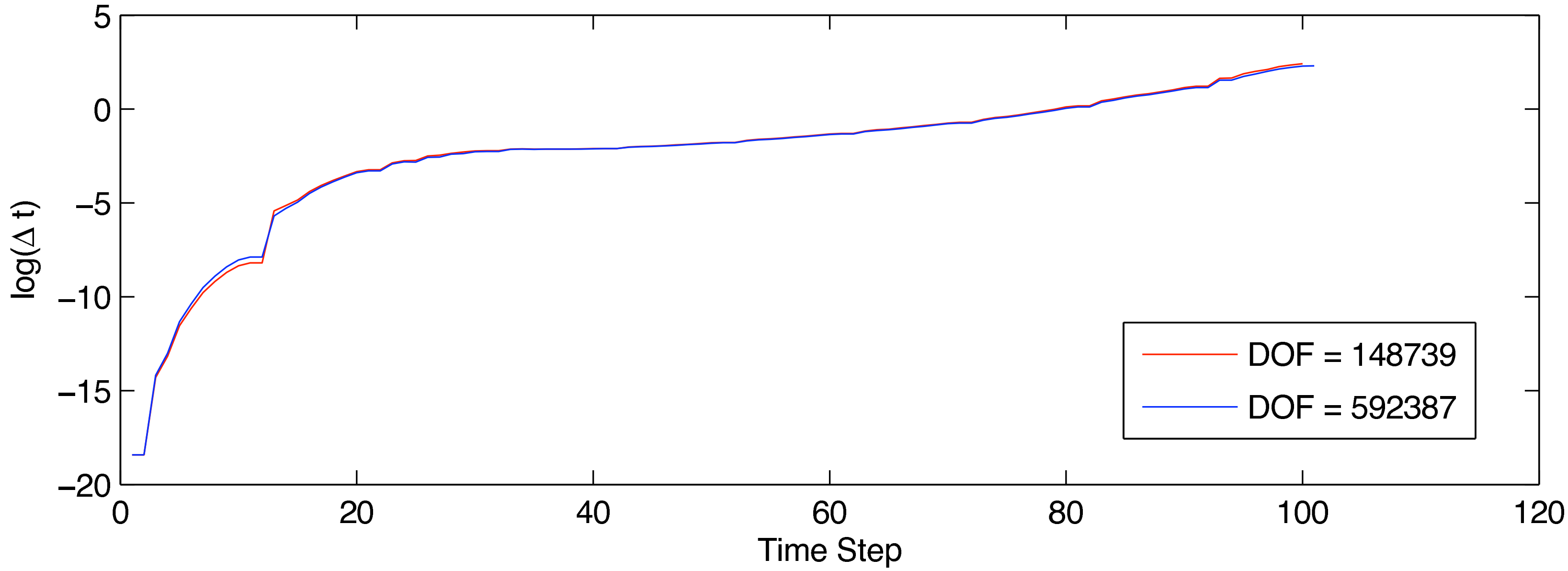
- Otherwise, accept the step and continue with $n = n + 1$ and k_{n+2} based on the LTE estimate and the accuracy tolerance tol.

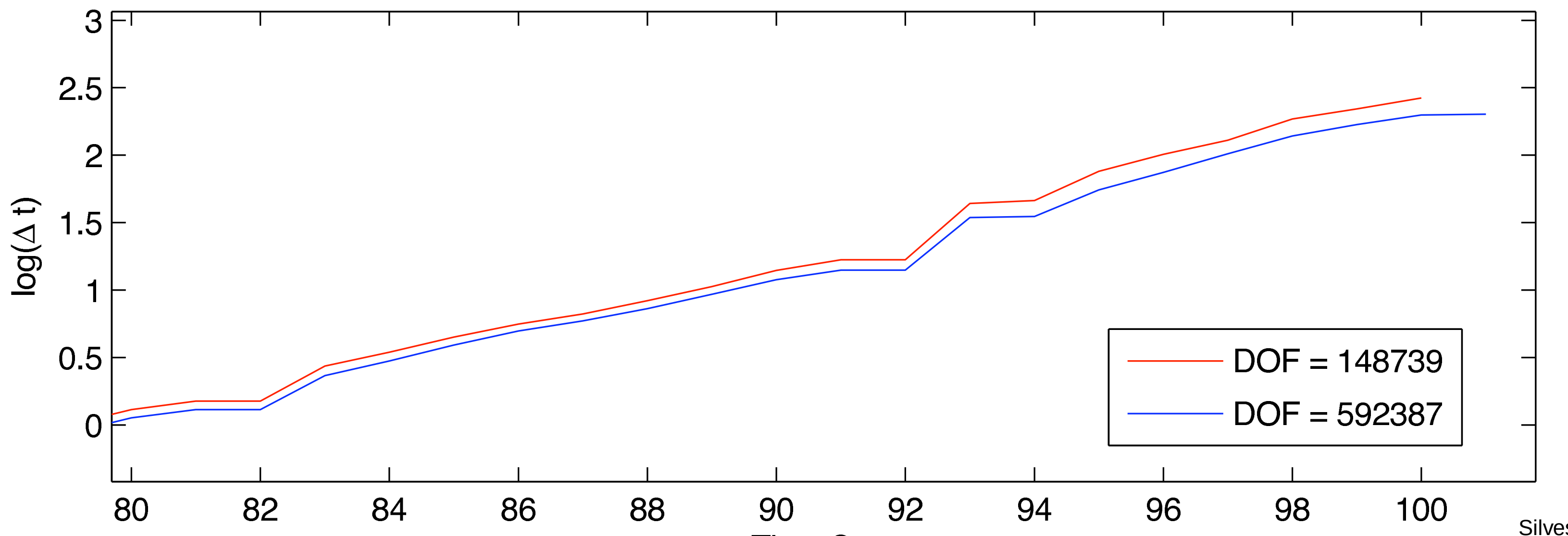
Example Flow Problem – I ($\nu = 1/1000$)



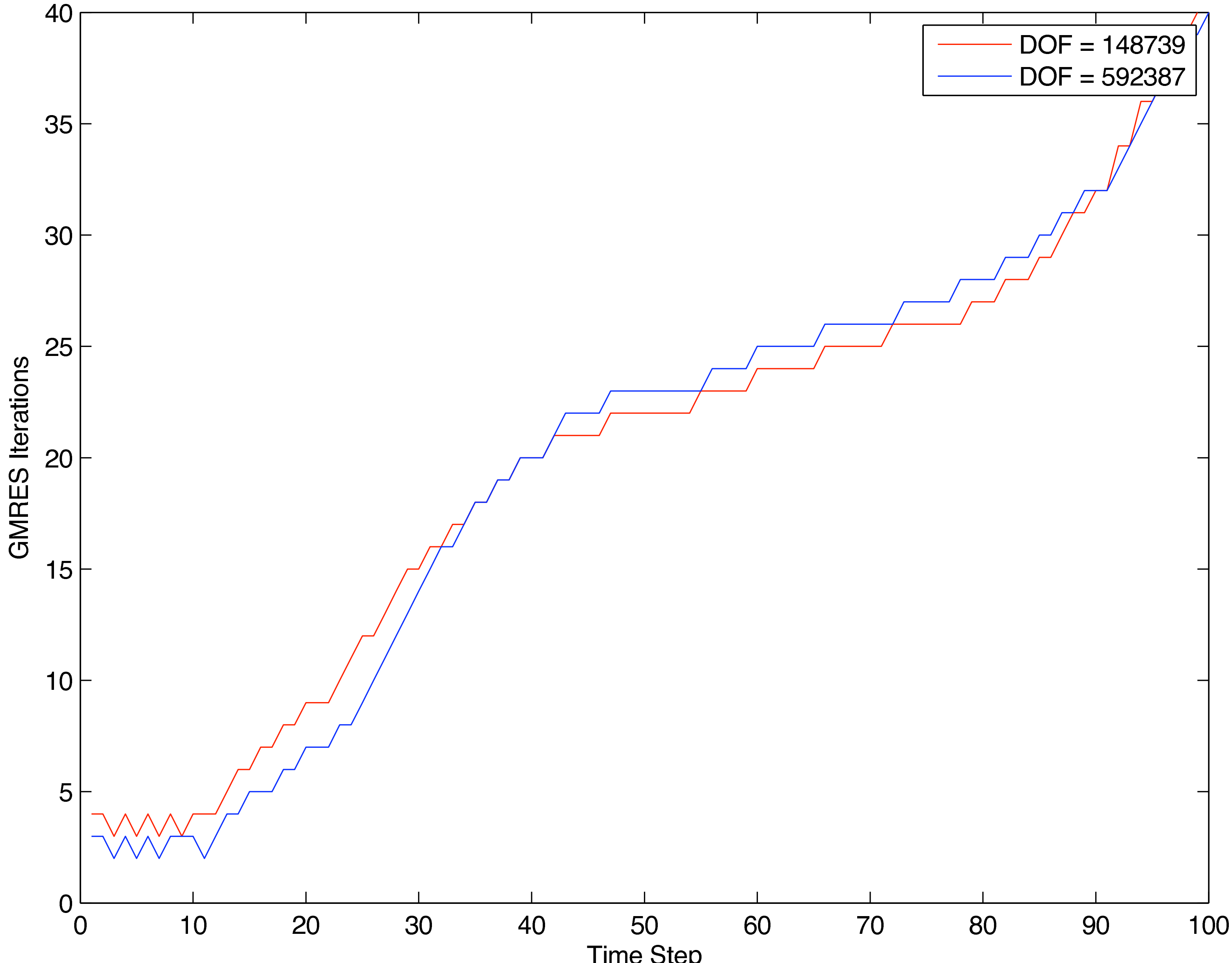


Time step evolution

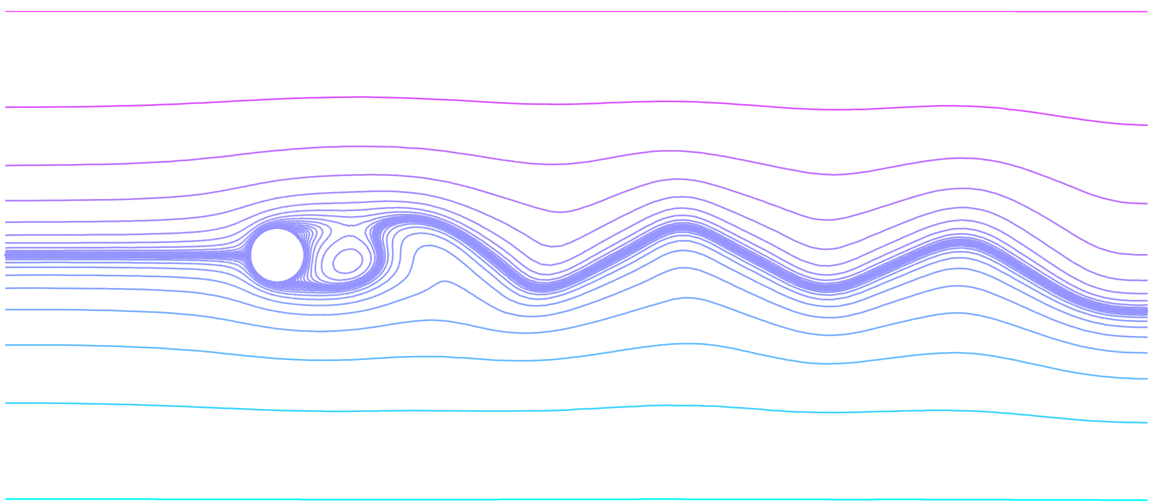


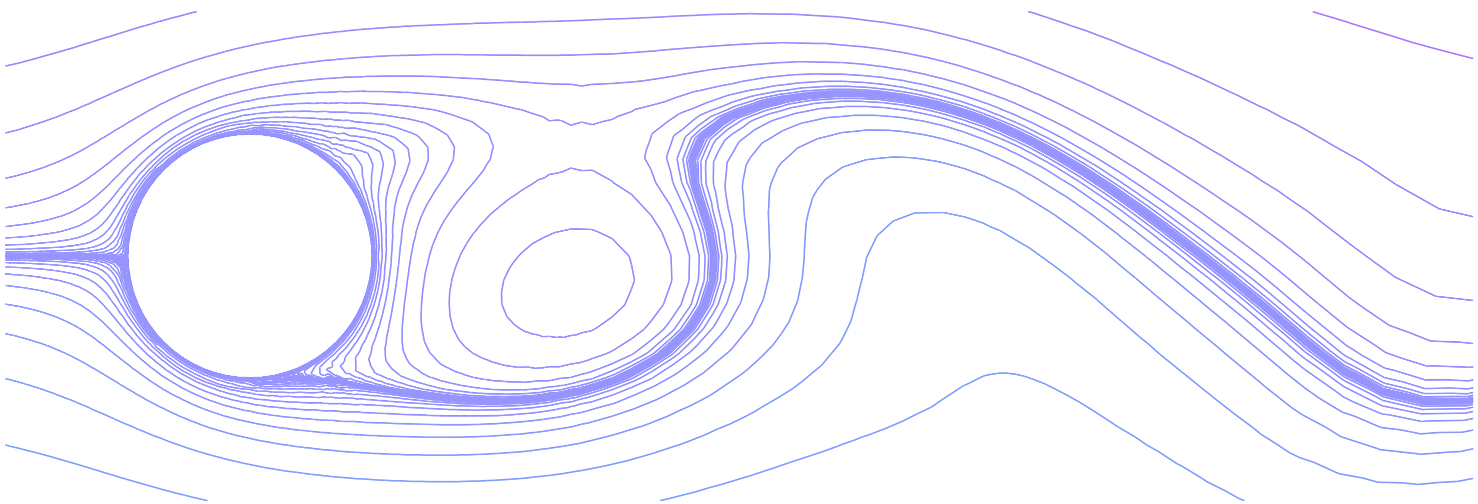


Linear solver performance

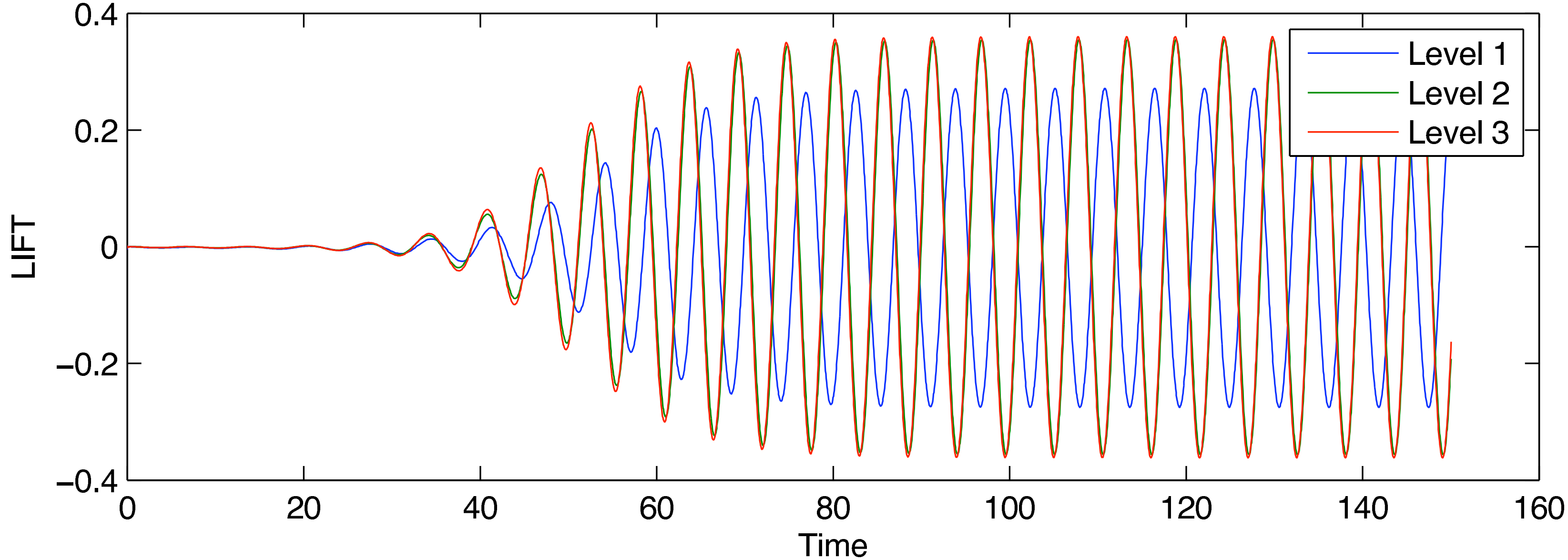


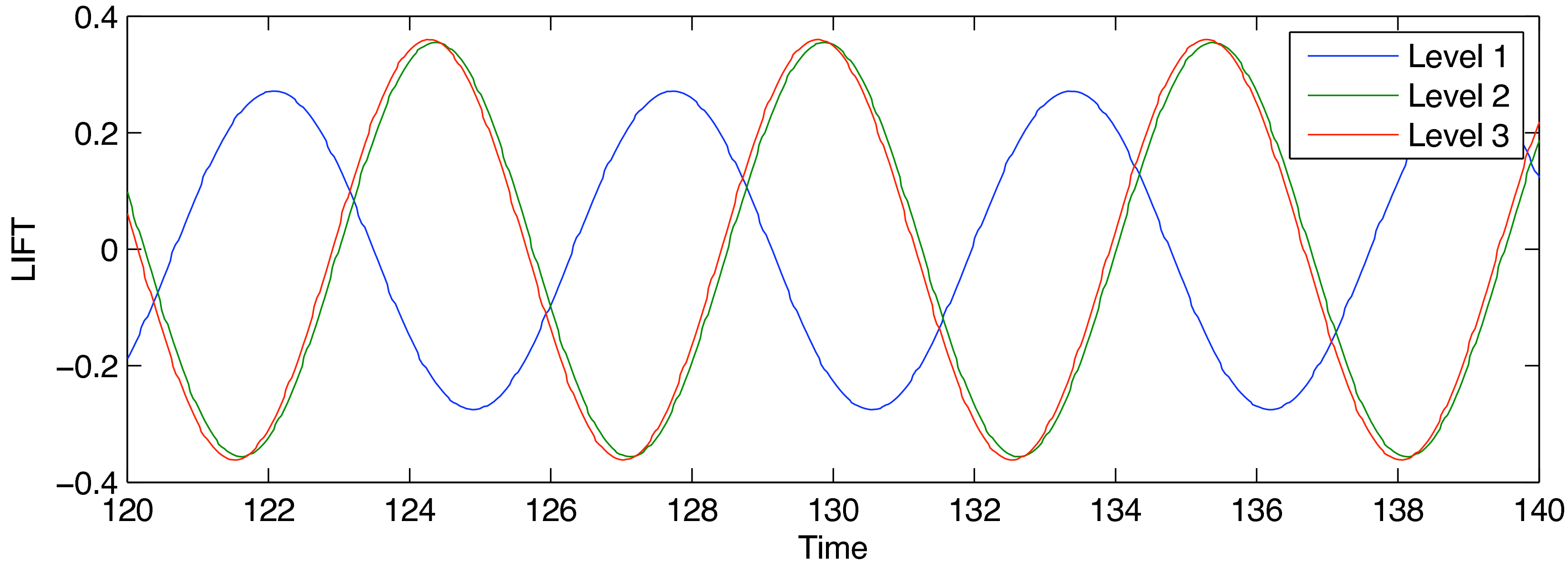
Example Flow Problem – II ($\nu = 1/100$)



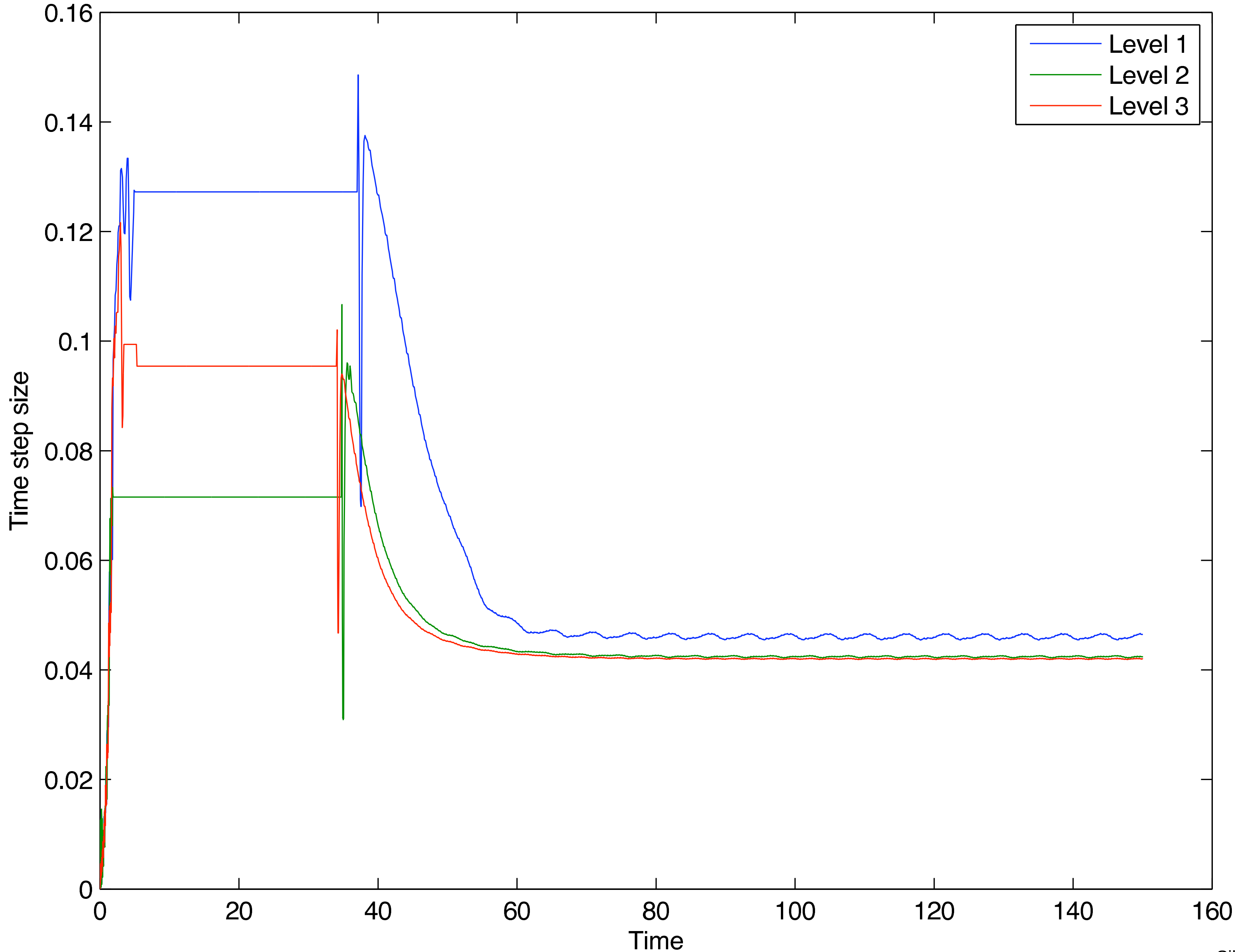


Lift Coefficient

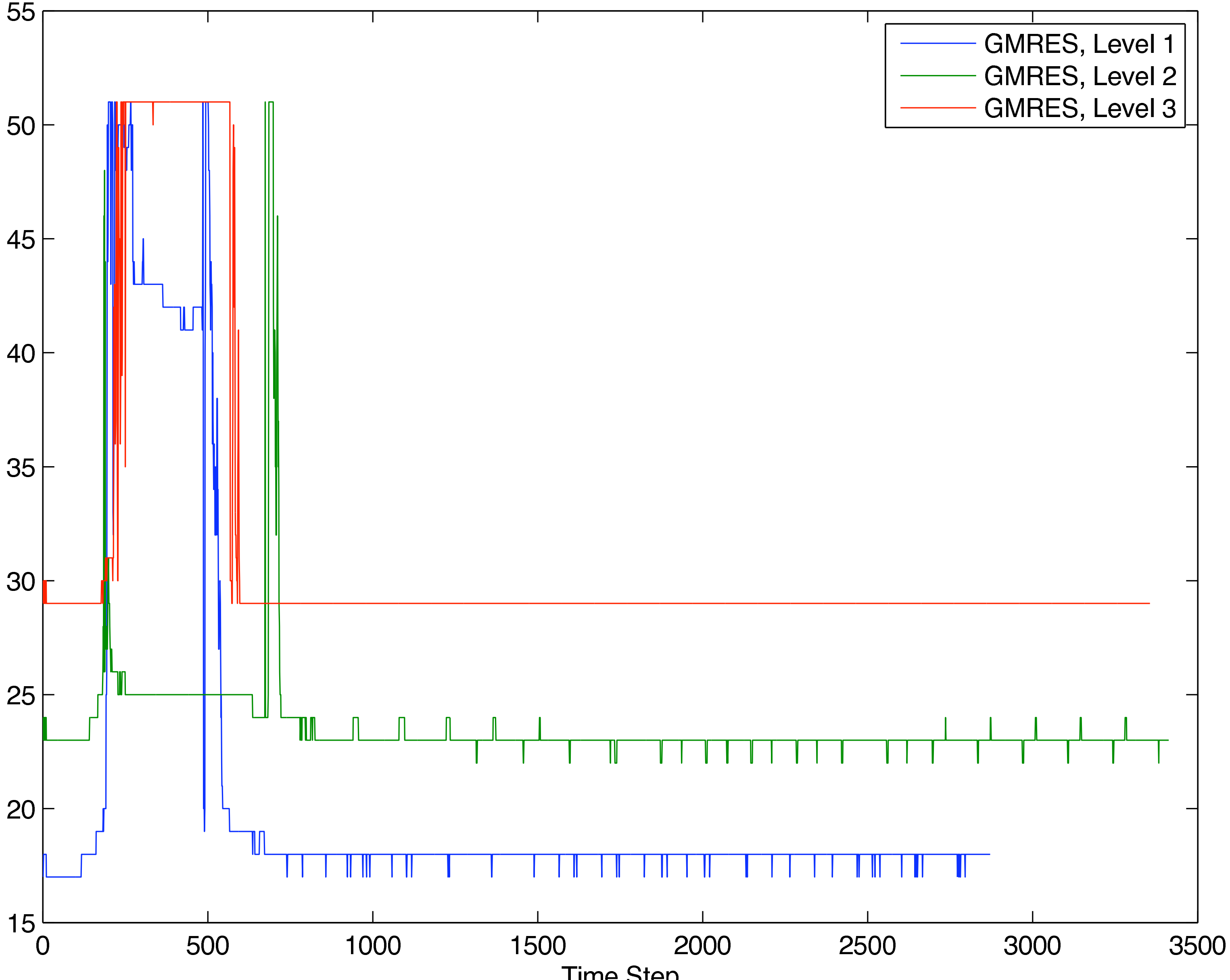




Time step evolution



Linear solver performance



Achievements

- Black-box implementation: no parameters that have to be estimated a priori.
- Optimal complexity: essentially $O(n)$ flops per iteration, where n is dimension of the discrete system.
- Optimal convergence: rate is bounded independently of h. Given an appropriate time accuracy tolerance convergence is also robust with respect to ν