Fast Iterative Solvers for Buoyancy Driven Flow Problems

David Silvester
School of Mathematics
University of Manchester

Outline

- PDEs
  - $$\left.\begin{aligned} \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} - \nu \nabla^2 \vec{u} + \nabla p = 0 \\ \nabla \cdot \vec{u} = 0 \end{aligned}\right\} \quad \text{Navier–Stokes}$$

Outline

- PDEs
  - $$\left.\begin{aligned}\frac{\partial \vec{u}}{\partial t} + \vec{u}\cdot\nabla\vec{u} - \nu\nabla^2\vec{u} + \nabla p = 0\\ \nabla\cdot\vec{u} = 0\end{aligned}\right\}\quad \text{Navier–Stokes}$$
  - $$\left.\begin{aligned}\frac{\partial \vec{u}}{\partial t} + \vec{u}\cdot\nabla\vec{u} - \nu\nabla^2\vec{u} + \nabla p = \vec{j}T\\ \nabla\cdot\vec{u} = 0\\ \frac{\partial T}{\partial t} + \vec{u}\cdot\nabla T - \nu\nabla^2 T = 0\end{aligned}\right\}\quad \text{Boussinesq}$$

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

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

Spatial Discretization— I

Introducing the basis sets

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

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

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

Spatial Discretization— II

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 $F := \nu A + N(\vec{u})$

Spatial Discretization— II

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 $F := \nu A + N(\vec{u})$

To reduce the index we differentiate the constraint and substitute into the momentum equation ...

Spatial Discretization— III

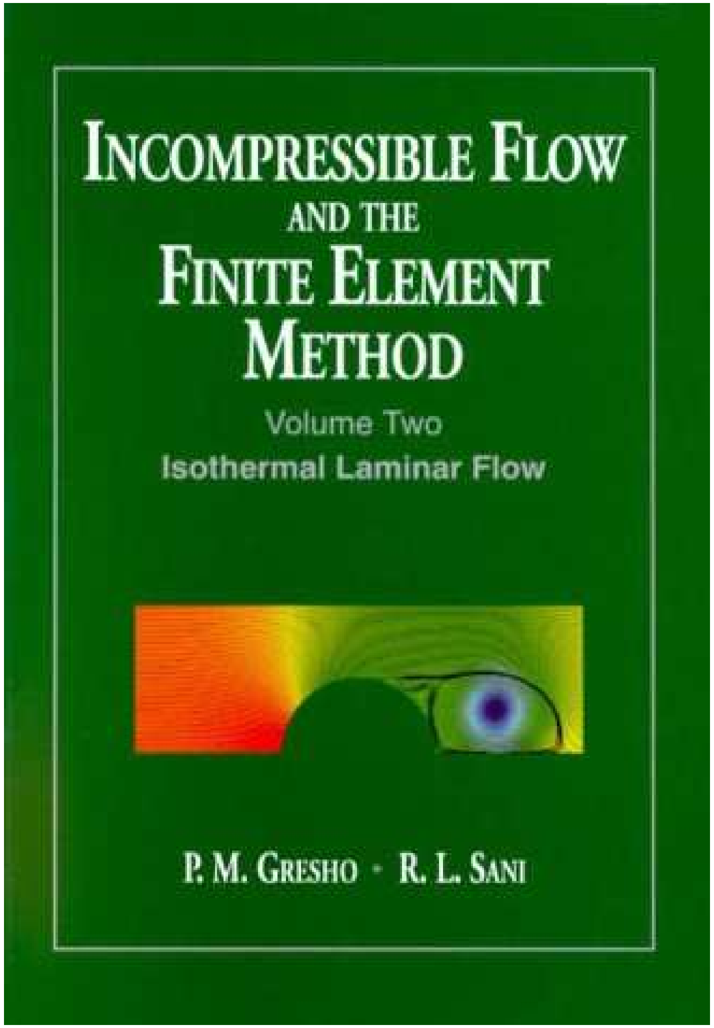
... to give an index one DAE system:

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial \vec{u}}{\partial t} \\ \frac{\partial p}{\partial t} \end{pmatrix} + \begin{pmatrix} F & B^T \\ BM^{-1}F & BM^{-1}B^T \end{pmatrix} \begin{pmatrix} \vec{u} \\ p \end{pmatrix} = \begin{pmatrix} \vec{f} \\ g \end{pmatrix}$$

The matrix $A_p = BM^{-1}B^T$ is the (consistent) Pressure Poisson matrix.

- Explicit approximation in time gives a decoupled formulation.
- Diagonally implicit approximation in time gives a segregated (SIMPLE-like) formulation.
- Implicit approximation in time does not look feasible.

Background


INCOMPRESSIBLE FLOW
AND THE
FINITE ELEMENT
METHOD
Volume Two
Isothermal Laminar Flow
P. M. GRESHO · R. L. SANI

- 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) 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 robust with respect to the mesh size h and the Reynolds number $1/\nu$.

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

$$
\begin{aligned}
\frac{2}{k_{n+1}}\,\vec{u}^{n+1} + \vec{w}^{n+1}\cdot\nabla\,\vec{u}^{n+1} - \nu\nabla^2\,\vec{u}^{n+1} + \nabla p^{n+1} &= \vec{f}^{n+1} \\
-\nabla\cdot\,\vec{u}^{n+1} &= 0 \qquad \text{in } \Omega \\
\vec{u}^{n+1} &= \vec{g}^{n+1} \quad \text{on } \Gamma_D \\
\nu\nabla\,\vec{u}^{n+1}\cdot\,\vec{n} - p^{n+1}\,\vec{n} &= \vec{0} \qquad \text{on } \Gamma_N
\end{aligned}
$$

with second-order linearization

$$
\begin{aligned}
\vec{w}^{n+1} &= \left(1 + \frac{k_{n+1}}{k_n}\right)\vec{u}^n - \frac{k_{n+1}}{k_n}\,\vec{u}^{n-1} \\
\vec{f}^{n+1} &= \frac{2}{k_{n+1}}\,\vec{u}^n + \nu\nabla^2\,\vec{u}^n - \vec{u}^n\cdot\nabla\,\vec{u}^n - \nabla p^n
\end{aligned}
$$

Saddle-point system

The discretized (Oseen–) system (*) is:

$$\begin{pmatrix} F^{n+1} & B^T \\ B & 0 \end{pmatrix} \begin{bmatrix} \alpha^{u,n+1} \\ \alpha^{p,n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}^{n+1} \\ \mathbf{g}^{n+1} \end{bmatrix}$$

- $F^{n+1} := \frac{2}{k_{n+1}} M + \nu A + N(\vec{w}_h^{n+1})$
- 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

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

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.

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

HSL HSL_MI20

PACKAGE SPECIFICATION HSL 2007

1 SUMMARY

Given an $n \times n$ sparse matrix **A** and an $n-$vector **z**, `HSL_MI20` computes the vector $\mathbf{x} = \mathbf{Mz}$, 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

[1] K. Stüben. *An Introduction to Algebraic Multigrid*. In U. Trottenberg, C. Oosterlee, A. Schüller, eds, 'Multigrid', Academic Press, 2001, pp 413-532.

ATTRIBUTES — Version: 1.1.0 **Types:** Real (single, double). **Uses:** `HSL_MA48`, `HSL_MC65`, `HSL_ZD11`, and the `LAPACK` routines `_GETRF` and `_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 `HSL_MI20` was funded by EPSRC grants EP/C000528/1 and GR/S42170.

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 $\|T_{n+1}\| \le \mathtt{tol} \times u_{\max}$. This criterion leads to

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

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 $\|T_{n+1}\| \leq \mathtt{tol} \times u_{\max}$. This criterion leads to

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

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:

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

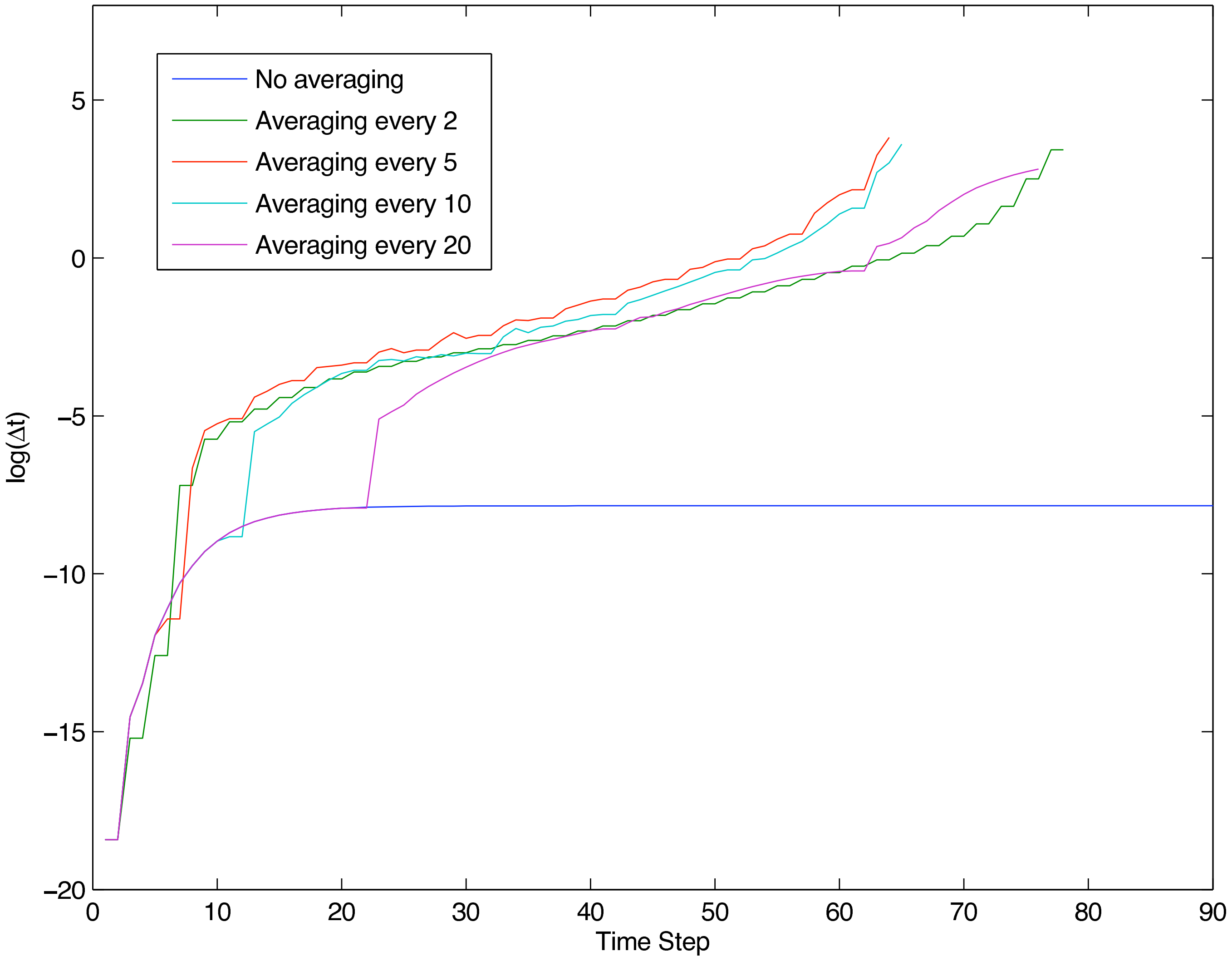
- We perform time-step averaging every n^* steps:

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

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

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

Stabilized AB2–TR



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

Adaptive Time-Stepping Algorithm

- The following parameters must be specified:

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

Adaptive Time-Stepping Algorithm

- The following parameters must be specified:

time accuracy tolerance	ε_t	(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

- The following parameters must be specified:

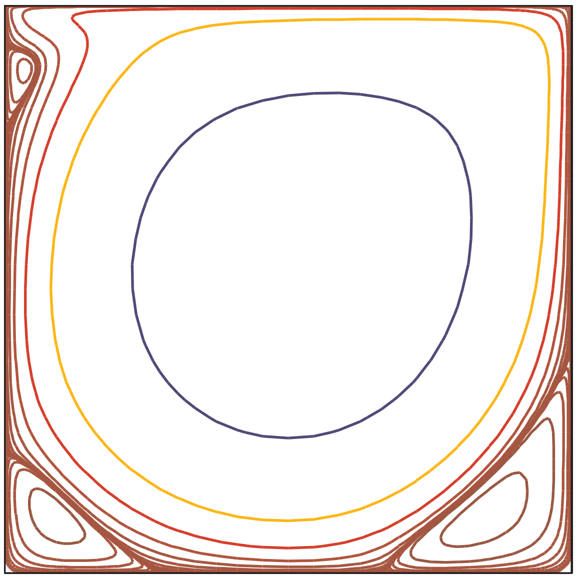
time accuracy tolerance	ε_t	(10^{-4})
GMRES tolerance	`itol`	(10^{-6})
GMRES iteration limit	`maxit`	(50)

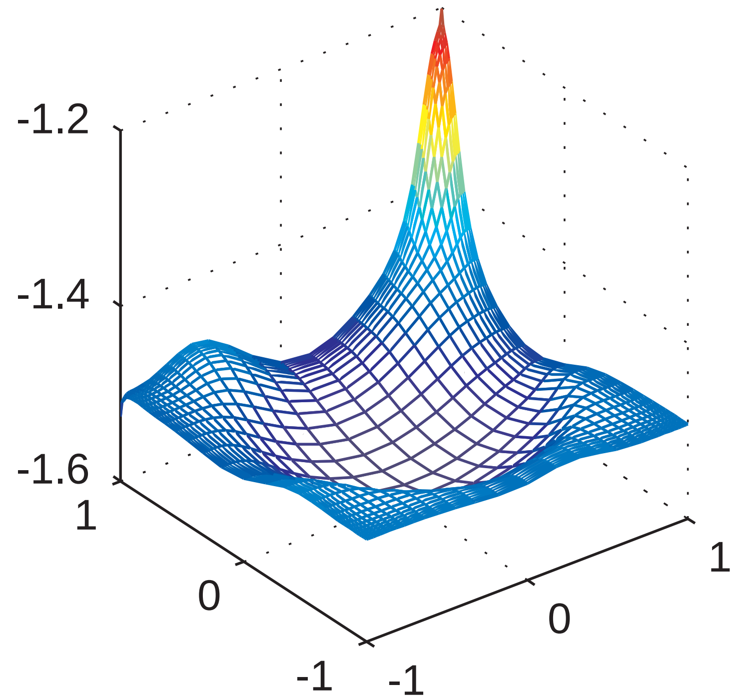
- Starting from rest, $\vec{u}^0 = \vec{0}$, and given a steady state boundary condition $\vec{u}(\vec{x}, t) = \vec{g}$, we model the impulse with a time-dependent boundary condition:

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

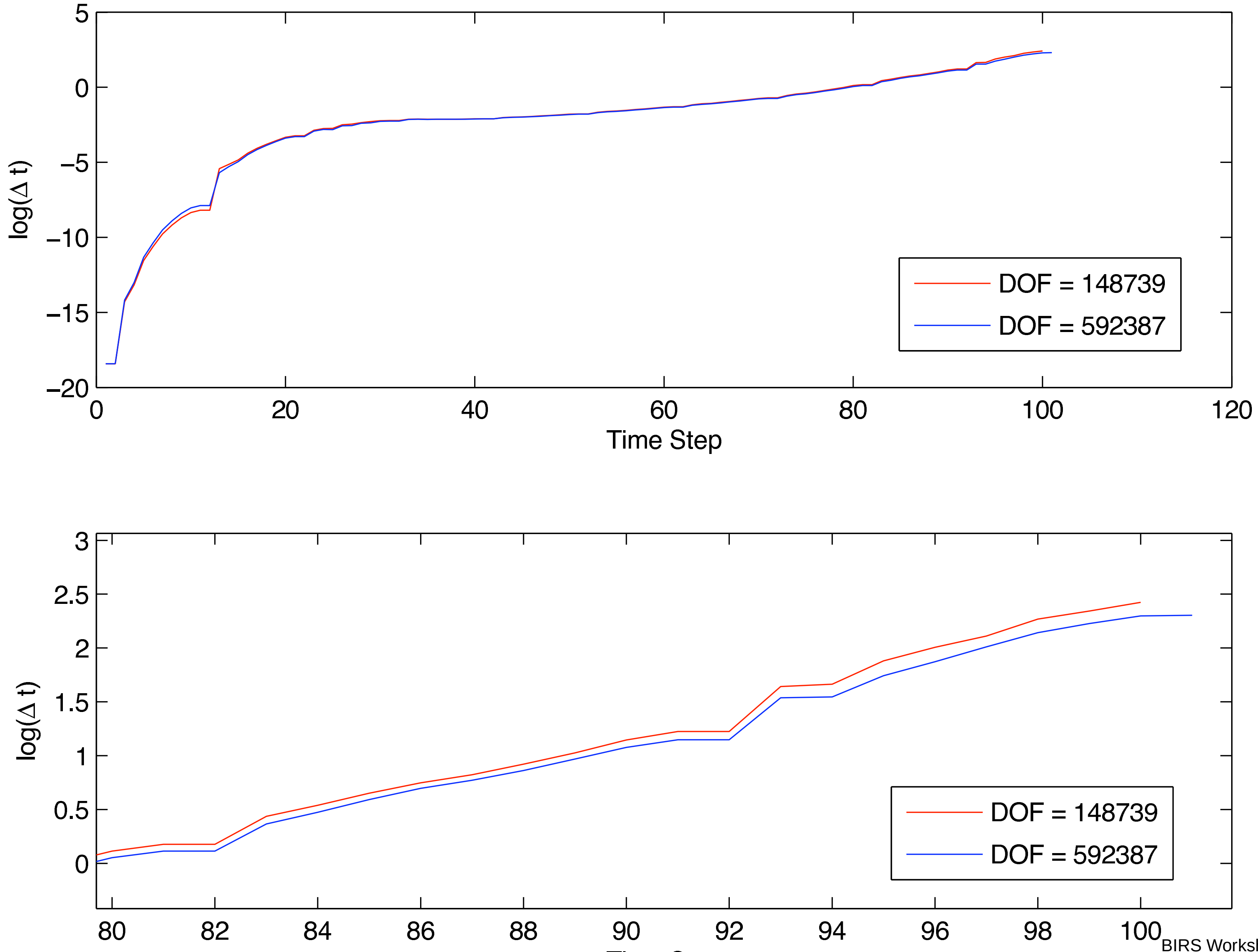
- We specify the frequency of averaging, typically $n_* = 10$. We also choose a very small initial timestep, typically, $k_1 = 10^{-8}$.

Example: Driven Cavity Flow ($\nu = 1/1000$)

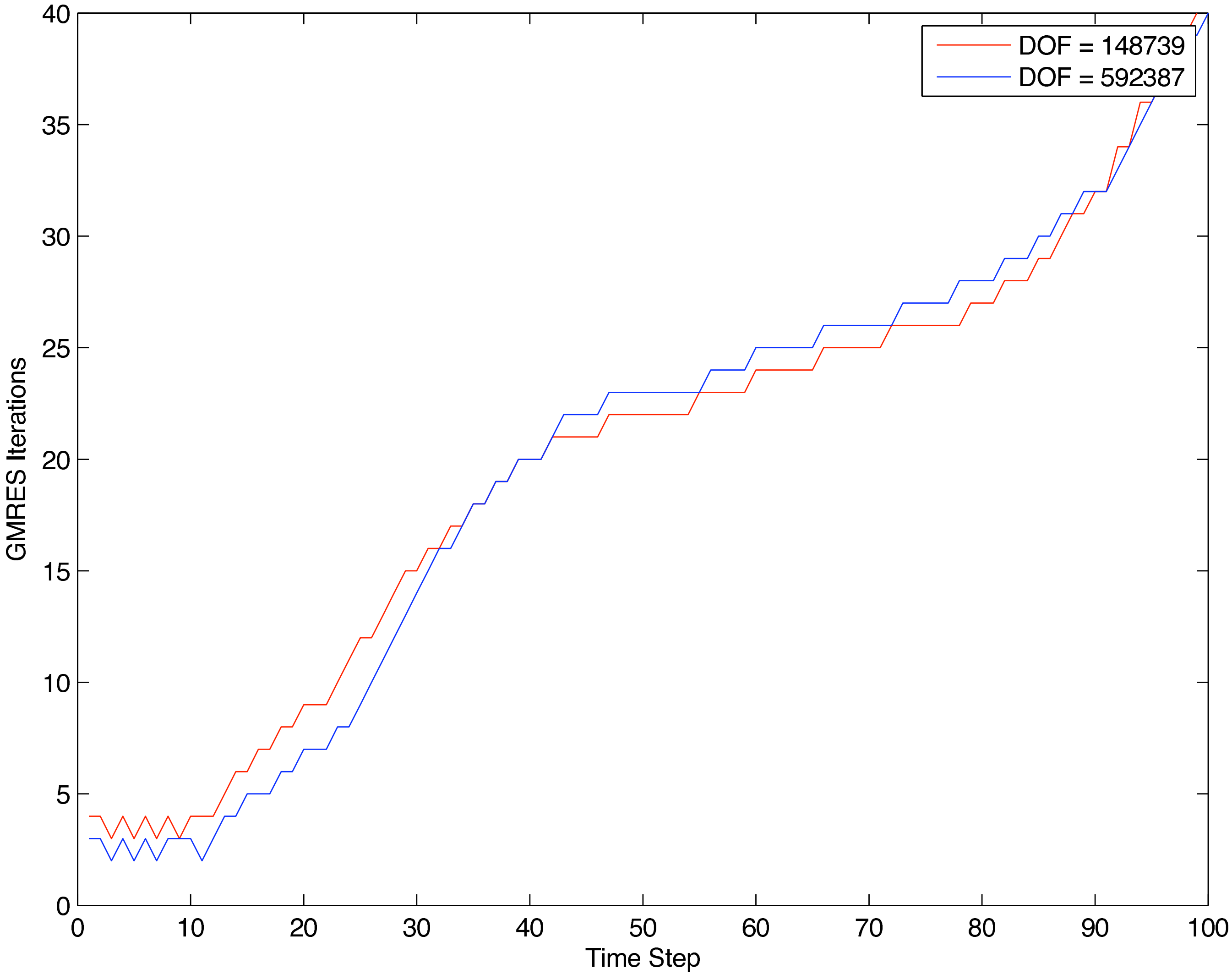




Time step evolution



Linear solver performance



Buoyancy driven flow

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

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

$$\frac{\partial T}{\partial t} + \vec{u}\cdot\nabla T - \nu\nabla^2 T = 0 \qquad \text{in } \mathcal{W}$$

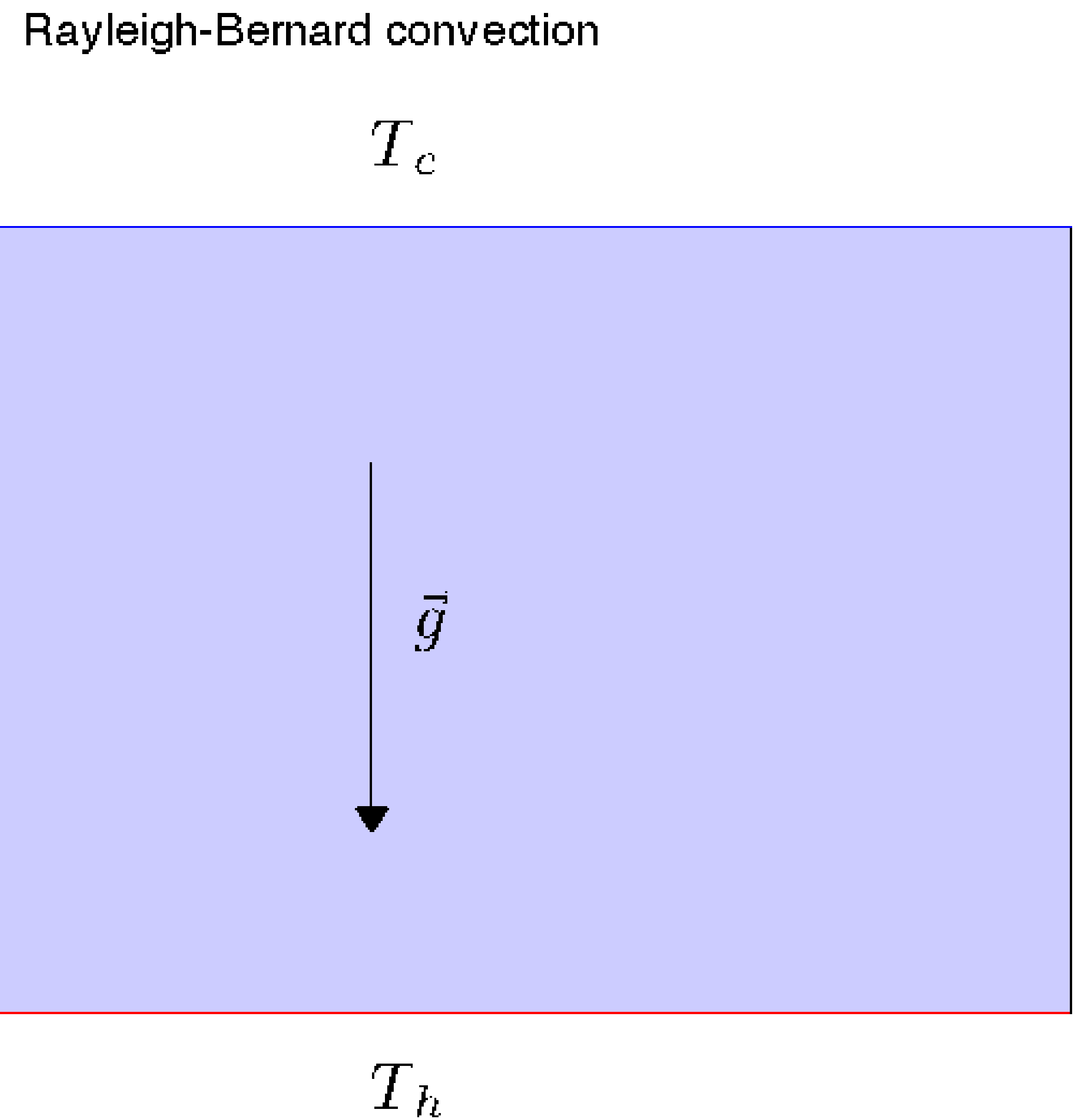
Boundary and Initial conditions

$$\vec{u} = \vec{0} \quad \text{on } \Gamma\times[0,T]; \qquad \vec{u}(\,\vec{x},0) = \vec{0} \quad \text{in } \Omega.$$

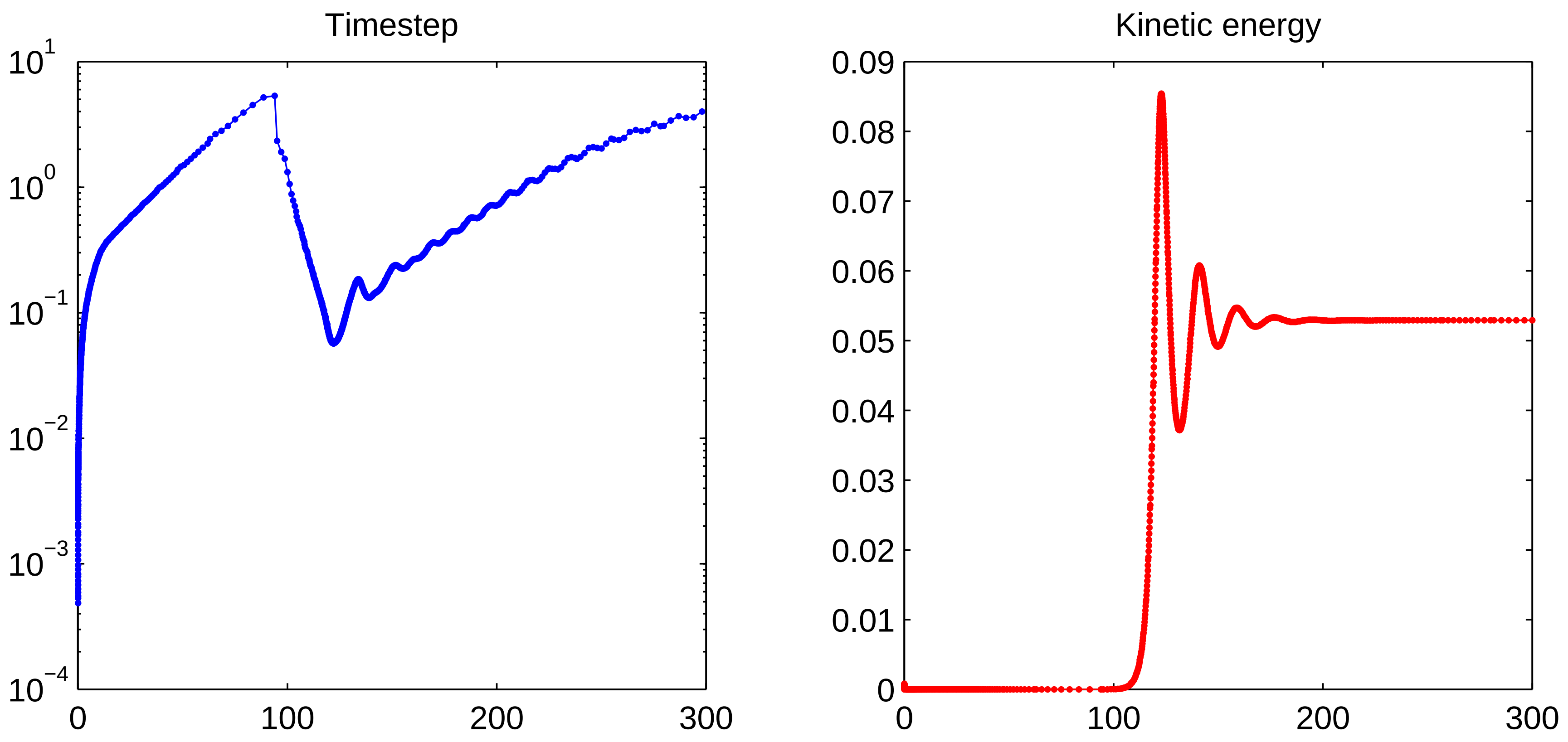
$$T = T_g \quad \text{on } \Gamma_D\times[0,T]; \qquad \nu\nabla T\cdot\,\vec{n} = 0 \quad \text{on } \Gamma_N\times[0,T];$$

$$T(\,\vec{x},0) = T_0(\,\vec{x}) \quad \text{in } \Omega.$$

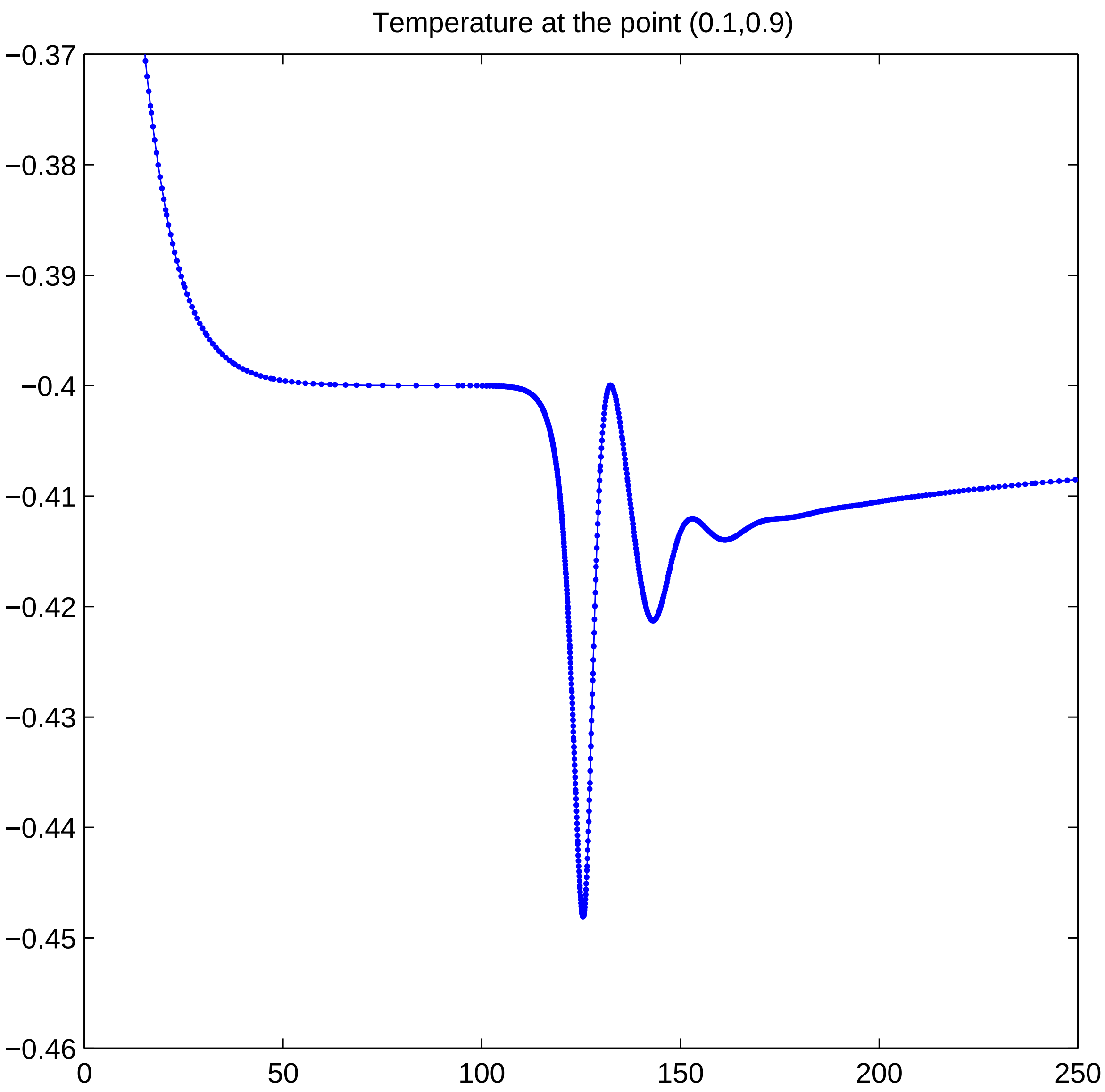
avi



Problem I: Timestep & Kinetic Energy : $\varepsilon_t = 10^{-6}$



Reference Point Temperature : $\varepsilon_t = 10^{-6}$



Finite element matrix formulation

Introducing the basis sets

$$\mathbf{X}_h = \text{span}\{\vec{\phi}_i\}_{i=1}^{n_u}, \quad \text{Velocity basis functions;}$$

$$M_h = \text{span}\{\psi_j\}_{j=1}^{n_p}, \quad \text{Pressure basis functions.}$$

$$T_h = \text{span}\{\phi_k\}_{k=1}^{n_T}, \quad \text{Temperature basis functions;}$$

gives the method-of-lines discretized system:

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

with a (vertical–) mass matrix:

$$(\overset{\circ}{\overline{M}})_{ij} = ([0, \phi_i], \phi_j)$$

Preconditioning strategy

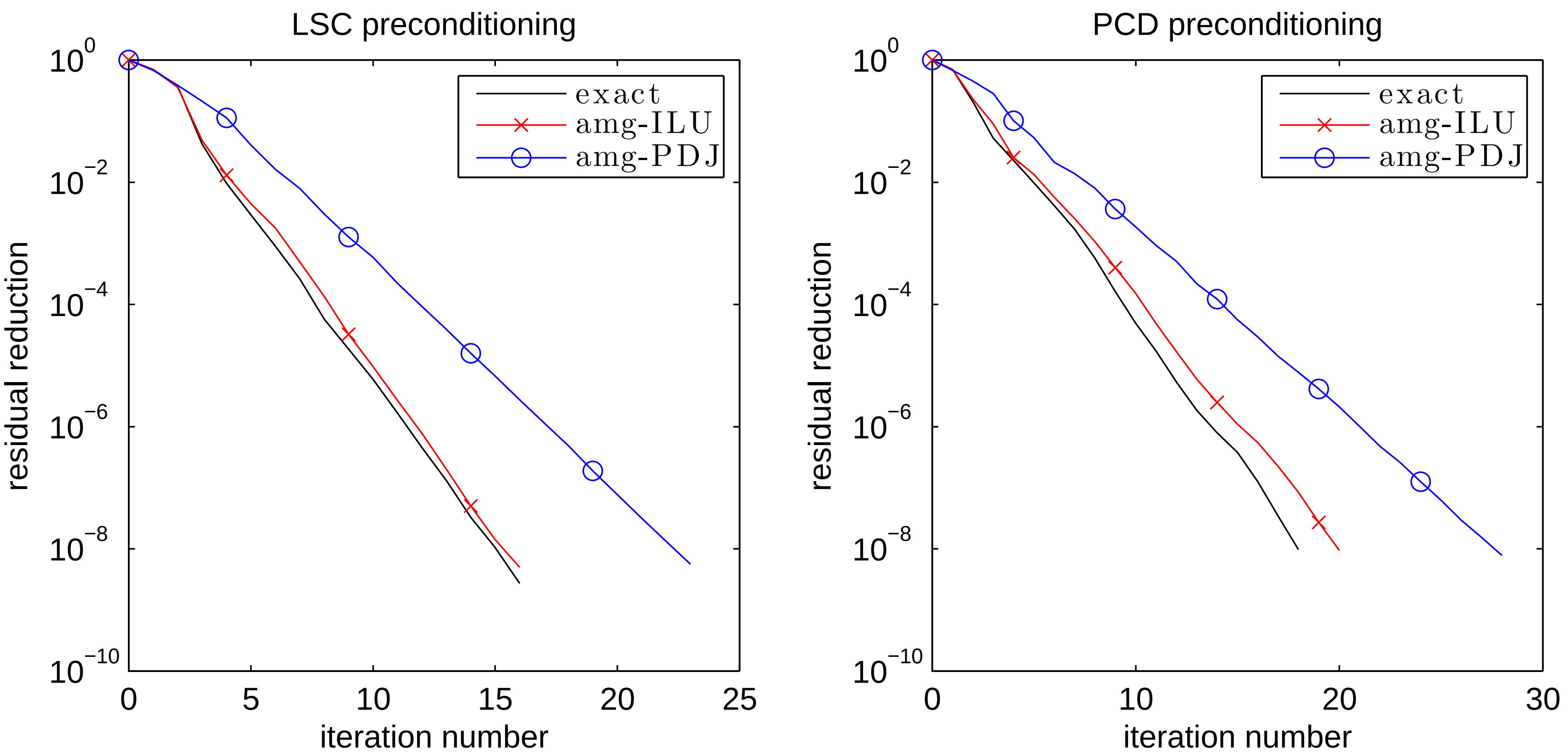
$$\begin{pmatrix} F & B^T & -\frac{\circ}{M} \\ B & 0 & 0 \\ 0 & 0 & F \end{pmatrix} \mathcal{P}^{-1} \quad \mathcal{P} \begin{pmatrix} \alpha^u \\ \alpha^p \\ \alpha^T \end{pmatrix} = \begin{pmatrix} \mathbf{f}^u \\ \mathbf{f}^p \\ \mathbf{f}^T \end{pmatrix}$$

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

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

$$= \begin{pmatrix} I & 0 & 0 \\ BF^{-1} & I & BF^{-1}\frac{\circ}{M}F^{-1} \\ 0 & 0 & I \end{pmatrix}$$

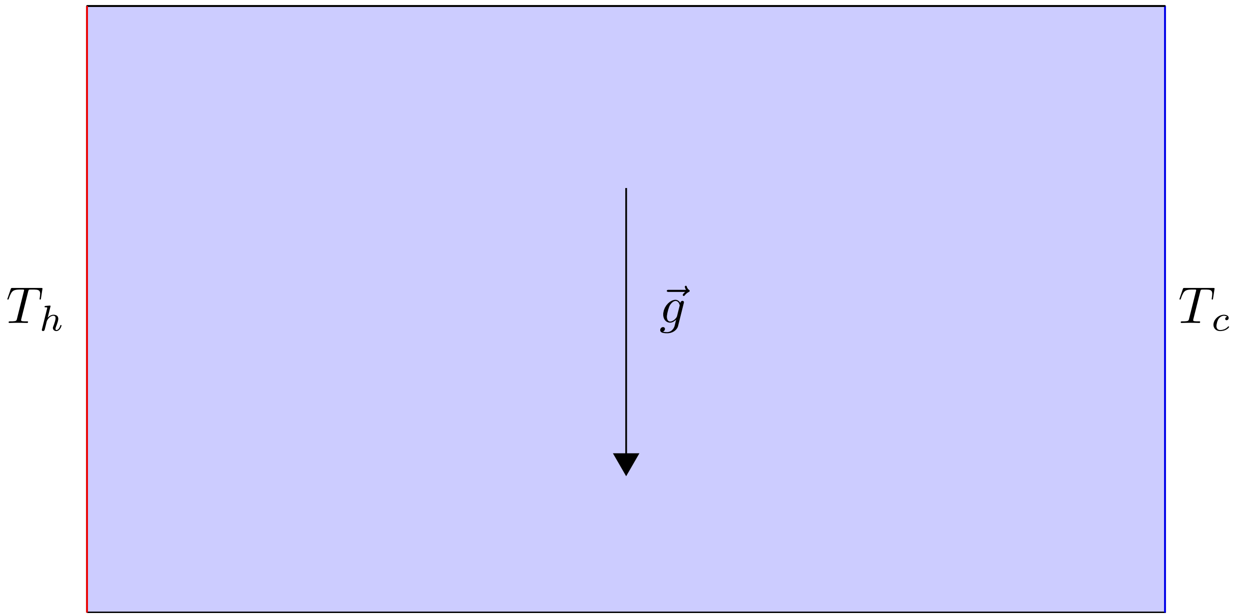
Solver performance



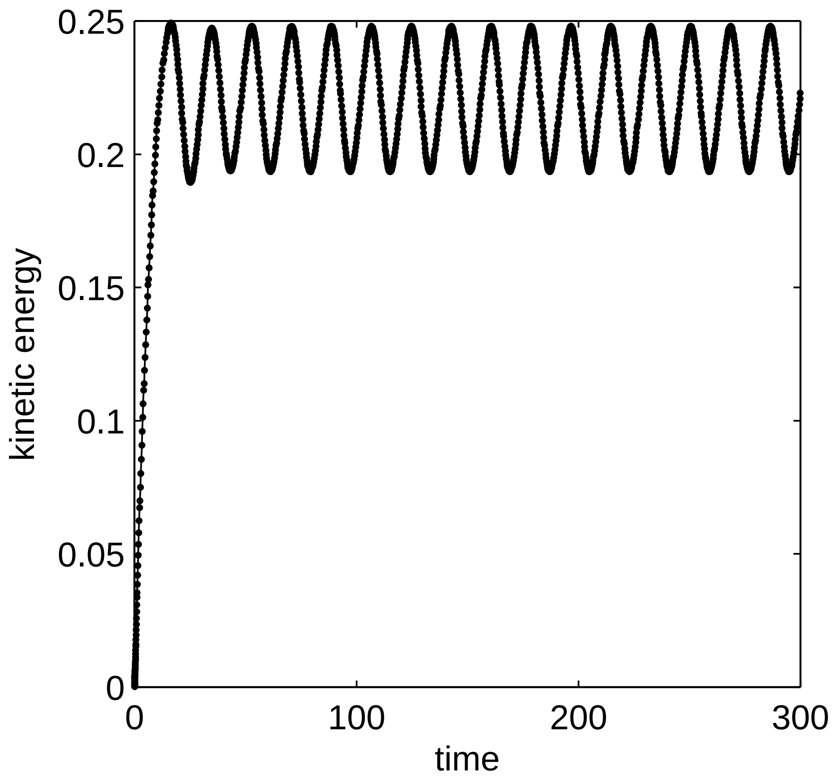
GMRES convergence close to steady state with $k_n \sim 4$.
Note that $\nu = 0.0218$ and $\nu = 0.00306$.

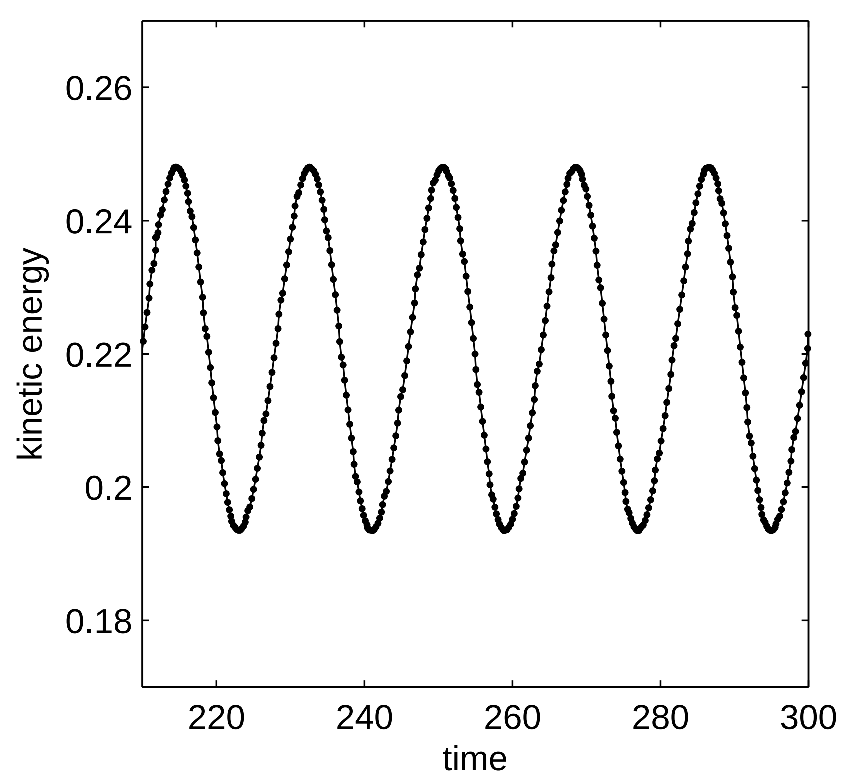
Problem II: 1:4 cavity domain

Lateral heating: Hopf Bifurcation

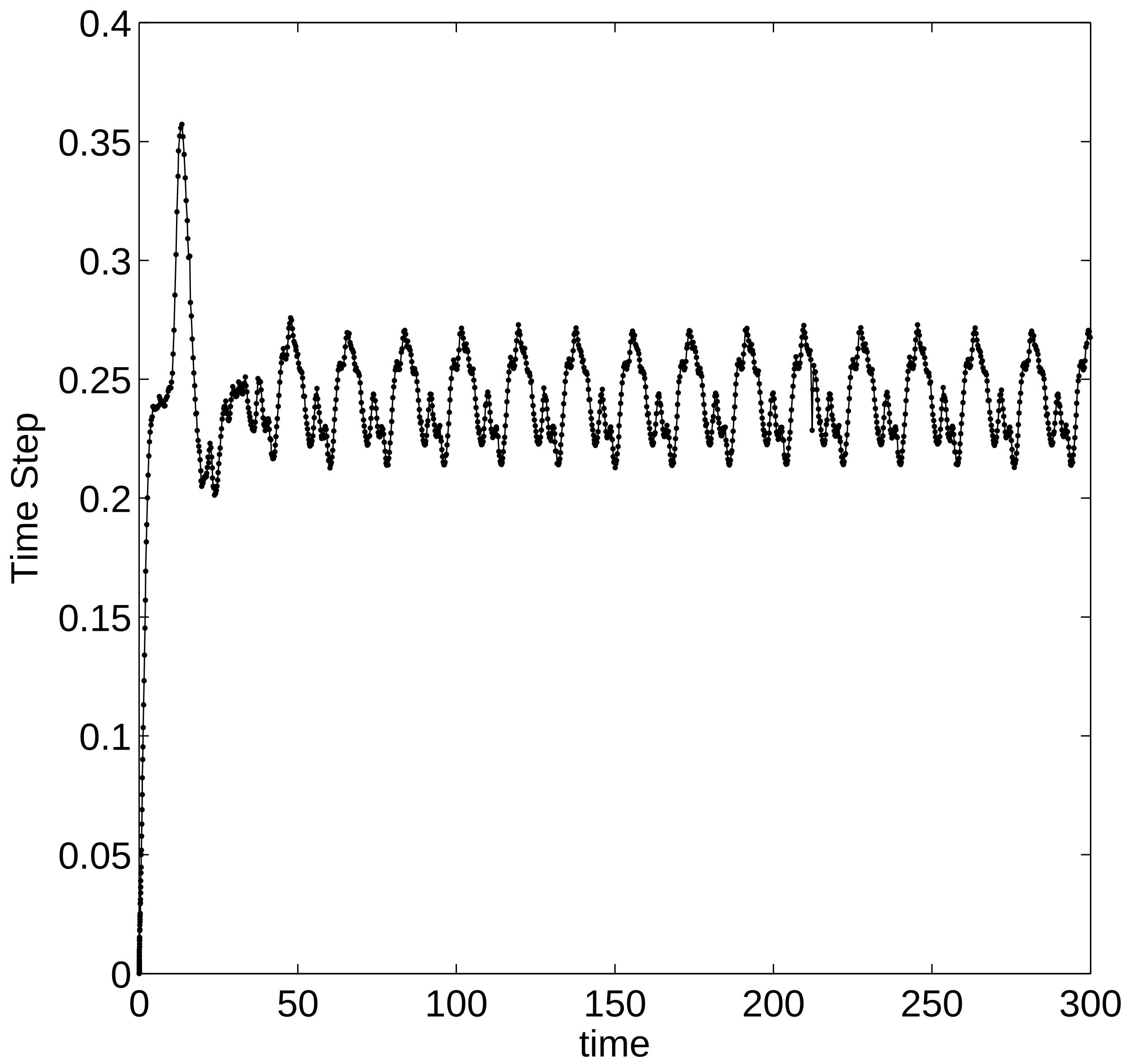


Problem II: Kinetic Energy : $\varepsilon_t = 3 \times 10^{-5}$

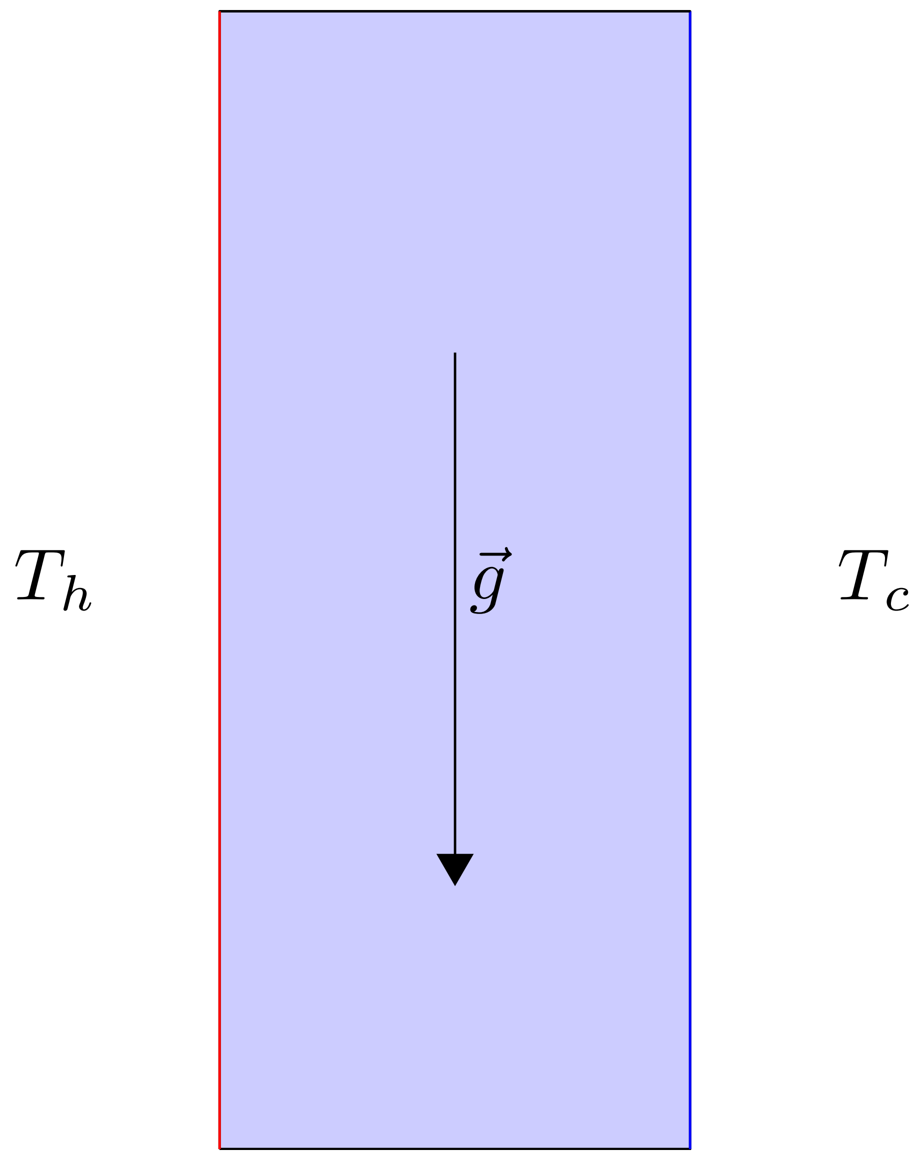




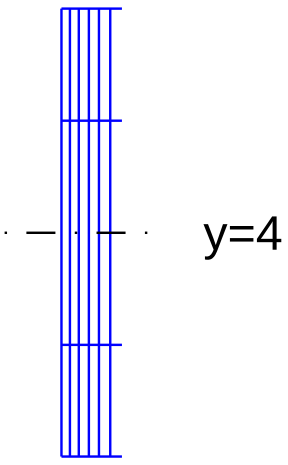
Problem II: Time step history : $\varepsilon_t = 3 \times 10^{-5}$

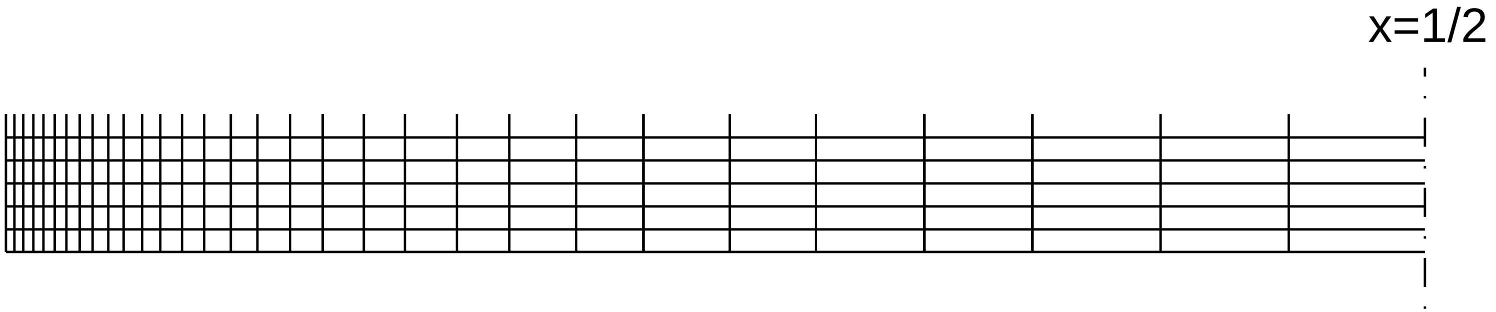


Problem XXX: 8:1 cavity domain



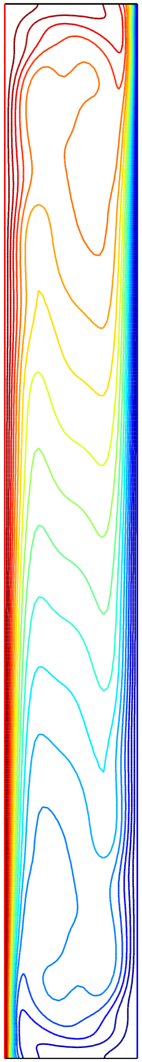
Problem XXX: 31×248 **stretched grid**



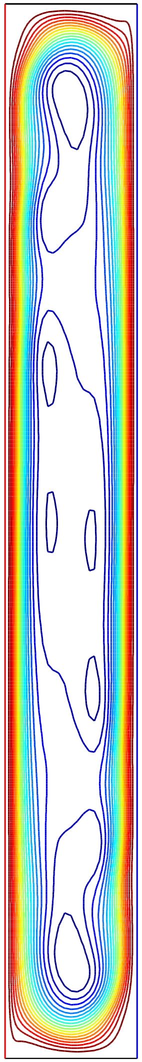


Problem XXX: Snapshot Solution

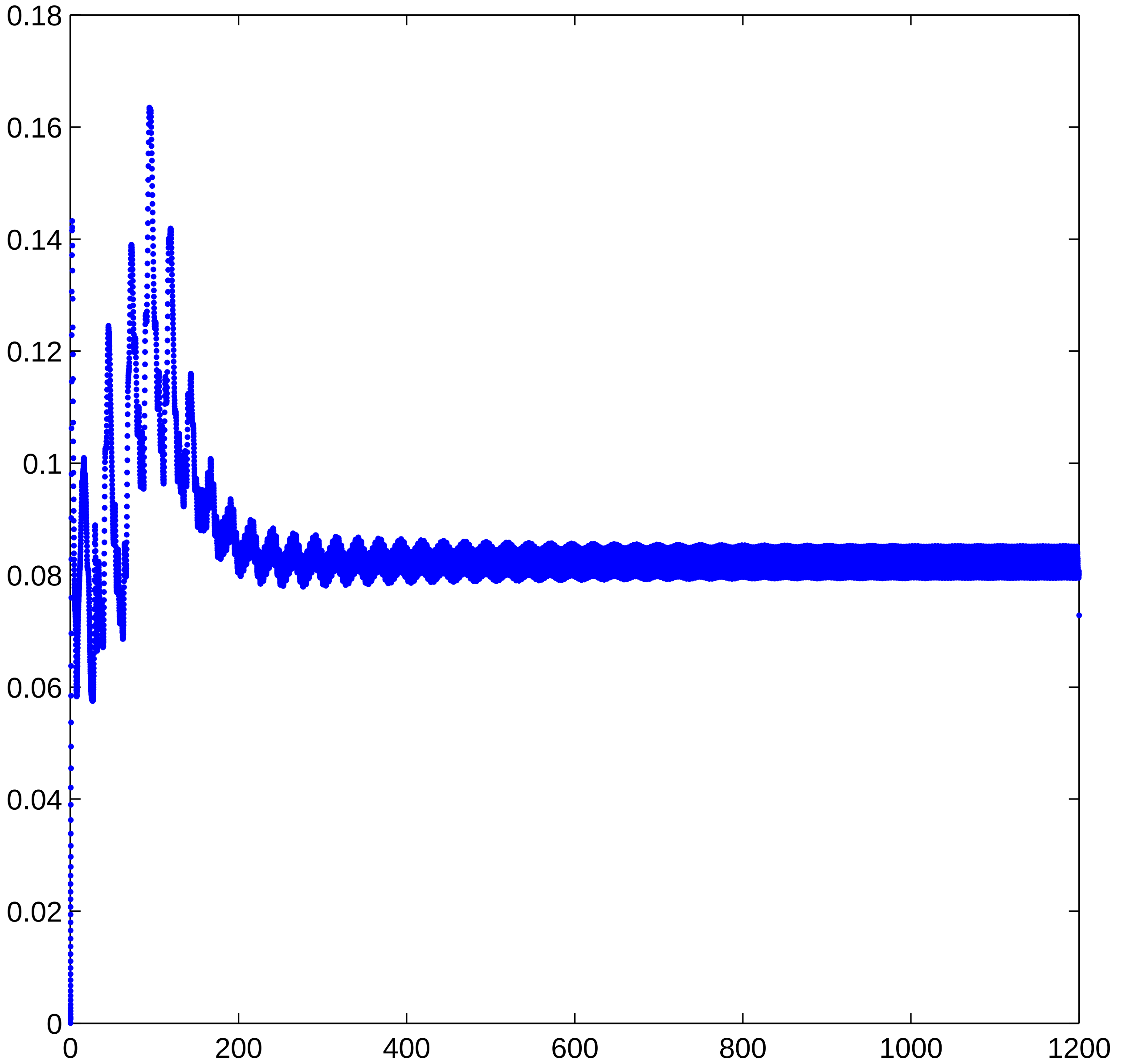
Isotherms : t=1200



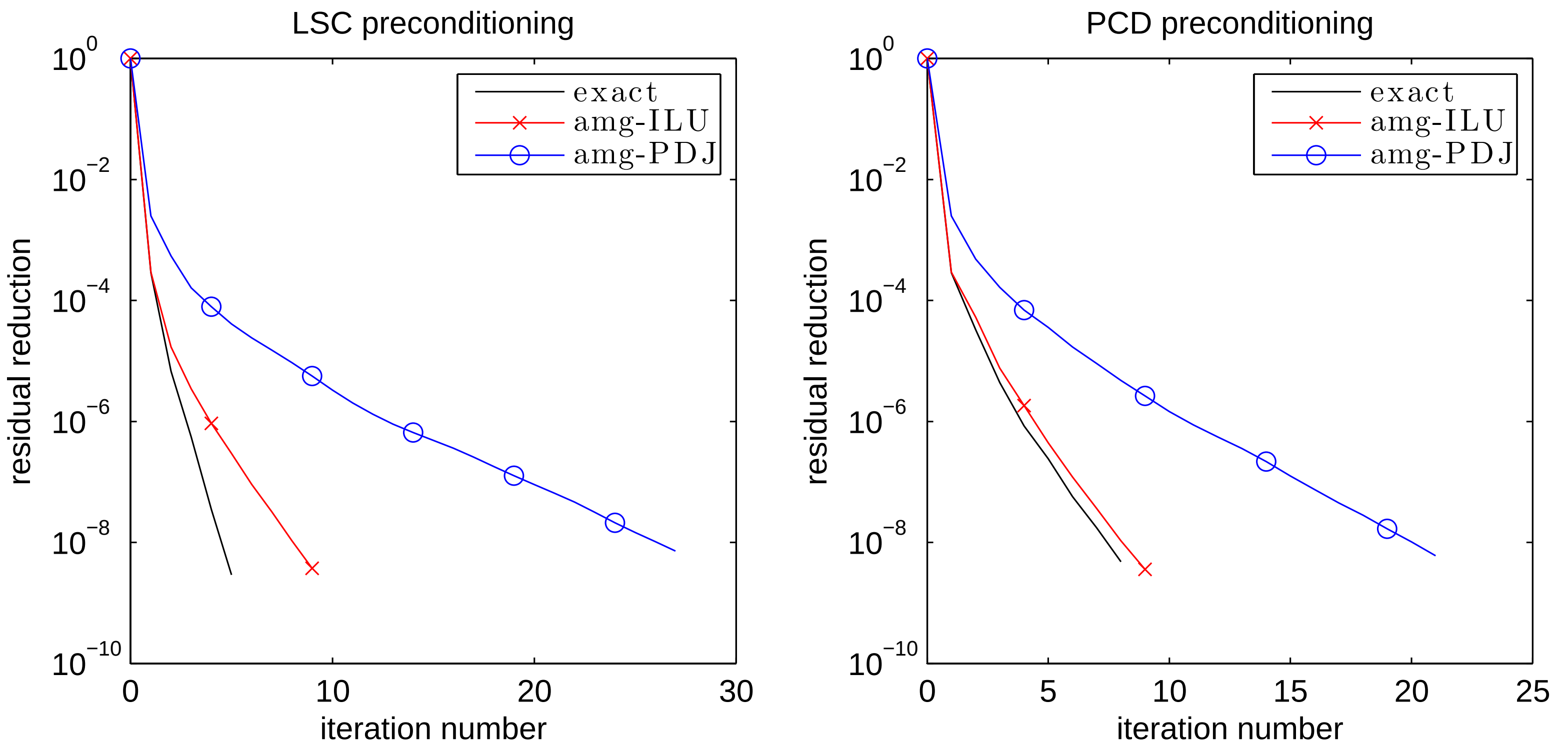
Streamlines : t=1200



Problem XXX: Time step history : $\varepsilon_t = 3 \times 10^{-5}$



Problem XXX: Solver performance



GMRES convergence for snapshot solution with $k_n \sim 0.082$.
Note that $\nu = 0.00145$ and $\nu = 0.00203$.

For further 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.

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

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":

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

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":

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