Fast Iterative Solvers for Buoyancy Driven Flow Problems

David Silvester
University of Manchester, UK

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

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

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

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

Buoyancy driven flow

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

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

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

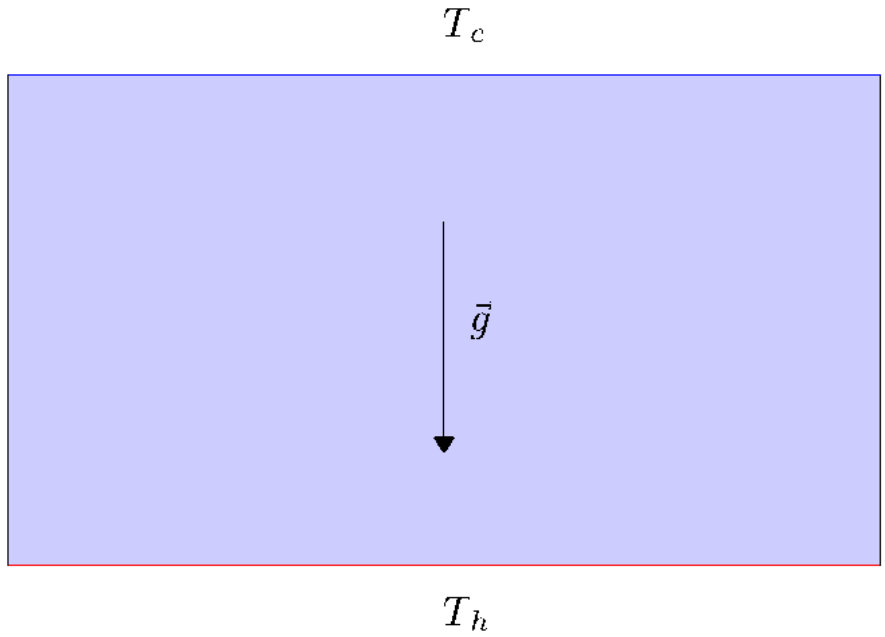
Boundary and Initial conditions

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

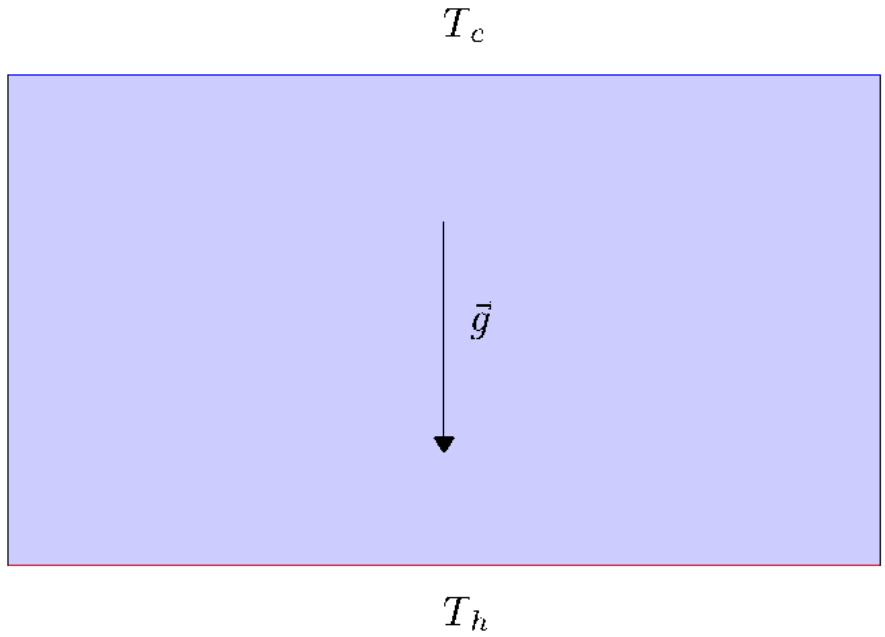
$$T = T_g \quad \text{on } \Gamma_D \times [0, T]; \qquad \nu \nabla T \cdot \vec{n} = 0 \quad \text{on } \Gamma_N \times [0, T];$$

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

$$\nu = \sqrt{Pr/Ra}\,, \qquad \nu = 1/\sqrt{Pr \cdot Ra}, \qquad T_g = (1 - e^{-10t}) T_\infty\,.$$

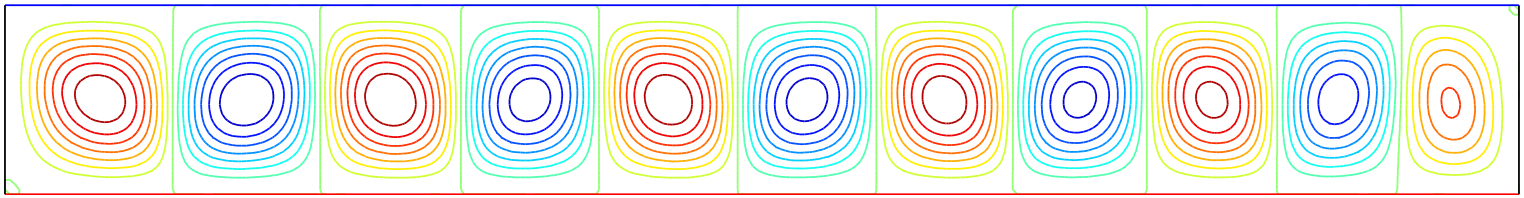


Rayleigh–Bénard $\quad | \quad Pr = 7.1, \quad Ra = 15000\ .$



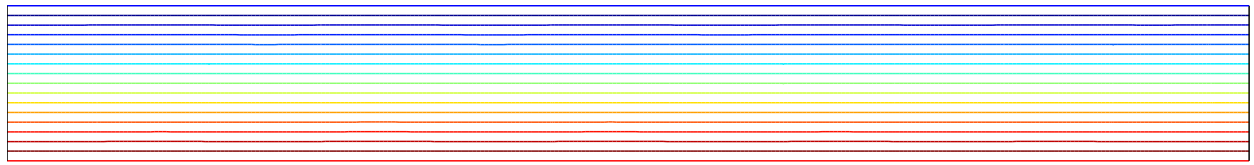
Rayleigh–Bénard | $Pr = 7.1, \quad Ra = 15000$.

Stationary streamlines: time = 300.00

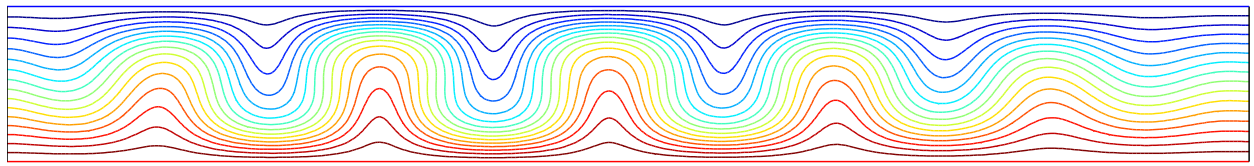


Rayleigh–Bénard | $Pr = 7.1, \quad Ra = 1.5 \times 10^4$.

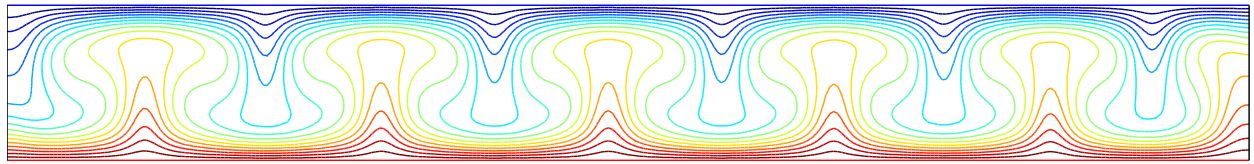
Isotherms: time = 100.72



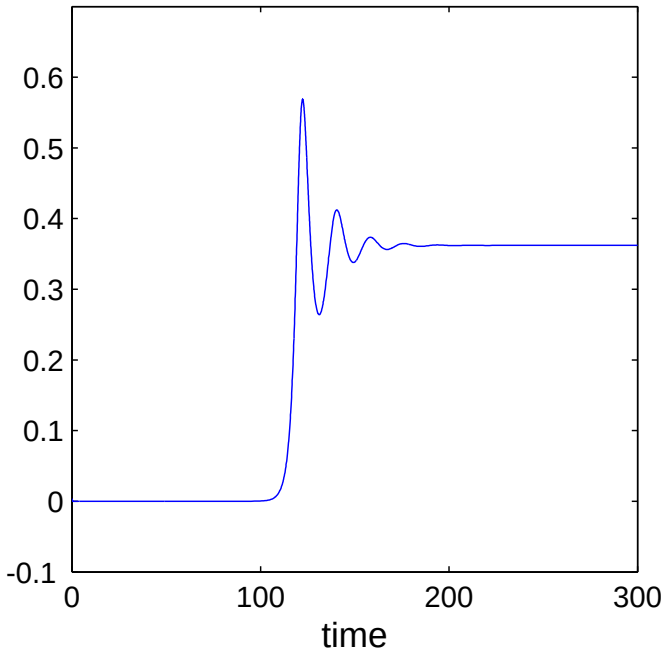
Isotherms: time = 119.28

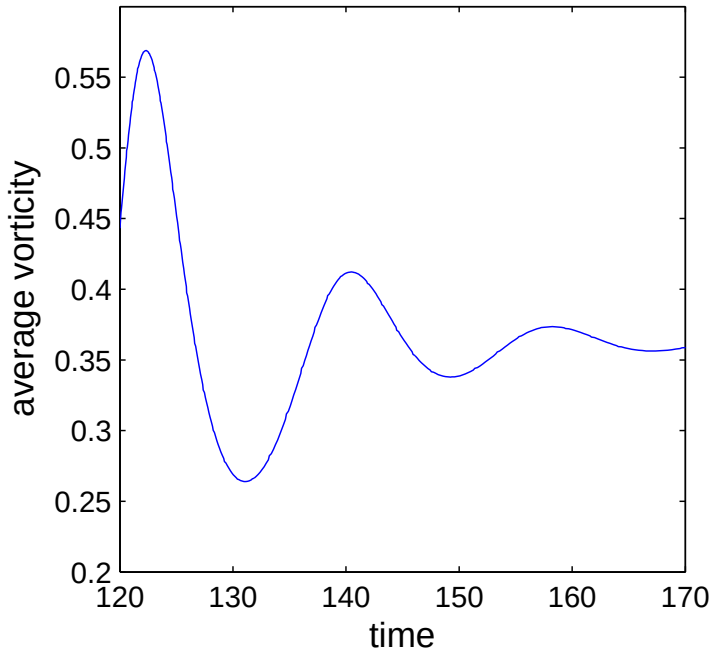


Isotherms: time = 300.00



Rayleigh–Bénard | $Pr = 7.1, \quad Ra = 1.5 \times 10^4$.

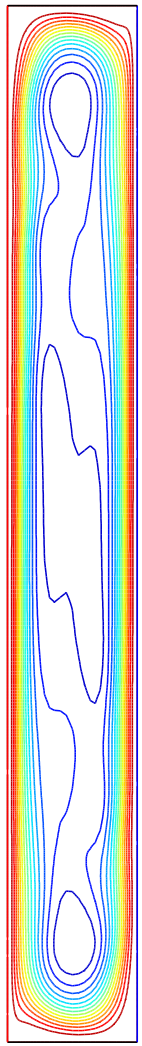




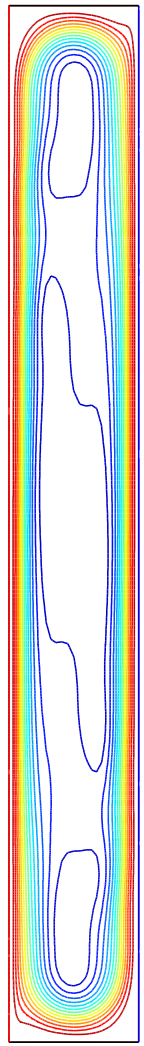
$$\omega = \nabla \times \vec{u}, \qquad \overline{\omega}_\Omega = \sqrt{\frac{1}{2\mathcal{A}} \int_\Omega \omega^2}$$

MIT test problem | $Pr = 0.71, \quad Ra = 3.4 \times 10^5$.

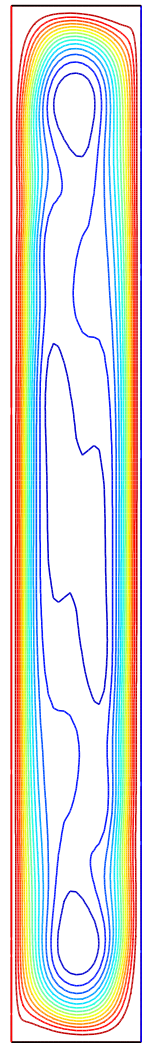
time = 826.53



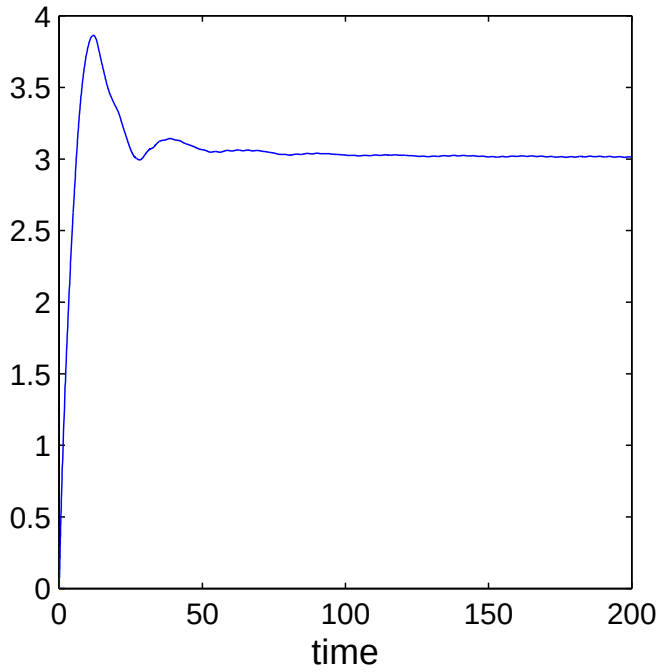
time = 828.23

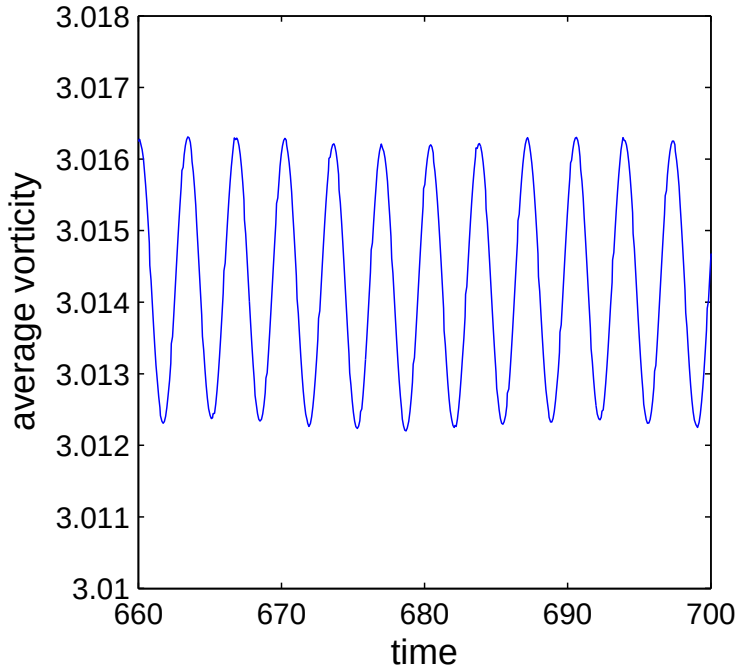


time = 829.96



MIT test problem | $Pr = 7.1, \quad Ra = 3.4 \times 10^5$.





$$\omega = \nabla \times \vec{u}, \qquad \overline{\omega}_\Omega = \sqrt{\frac{1}{2\mathcal{A}} \int_\Omega \omega^2}$$

References

- 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.
- Howard Elman, Milan Mihajlović and David Silvester.
  Fast iterative solvers for buoyancy driven flow problems
  J. Computational Physics, 230: 3900–3914, 2011.

Incompressible Flow & Iterative Solver Software

An open-source software package

Summary

IFISS is a graphical package for the interactive numerical study of incompressible flow problems which can be run under Matlab or Octave. It includes algorithms for discretization by mixed finite element methods and a posteriori error estimation of the computed solutions. The package can also be used as a computational laboratory for experimenting with state-of-the-art preconditioned iterative solvers for the discrete linear equation systems that arise in incompressible flow modelling.

Key Features

Key features include

- implementation of a variety of mixed finite element approximation methods;
- automatic calculation of stabilization parameters where appropriate;
- a posteriori error estimation for steady problems;
- a range of state-of-the-art preconditioned Krylov subspace solvers ;
- built-in geometric and algebraic multigrid solvers and preconditioners;
- fully implicit self-adaptive time stepping algorithms;
- useful visualization tools.

The developers of the IFISS package are David Silvester (School of Mathematics, University of Manchester), Howard Elman (Computer Science Department, University of Maryland), and Alison Ramage (Department of Mathematics and Statistics, University of Strathclyde).

Links

Download

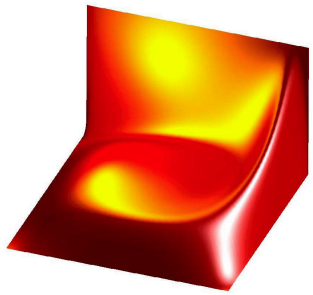
Documentation

Publications

Overview

Sample output

Contact



The IFISS logo represents the solution of the *double glazing* convection-diffusion problem. It can be reproduced in IFISS via the function **ifisslogo**.

"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 ...

PART I

Trapezoidal Rule (TR) time discretization

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

$$\frac{2}{k_{n+1}}\vec{u}^{n+1} - \nu\nabla^2\vec{u}^{n+1} + \vec{u}^{n+1}\cdot\nabla\vec{u}^{n+1} + \nabla p^{n+1} - \vec{j}T^{n+1} = \frac{2}{k_{n+1}}\vec{u}^n + \frac{\partial\vec{u}}{\partial t}^n \text{ in } \Omega$$

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

$$\vec{u}^{n+1} = \vec{0} \quad \text{on } \Gamma$$

$$\frac{2}{k_{n+1}}T^{n+1} - \nu\nabla^2 T^{n+1} + \vec{u}^{n+1}\cdot\nabla T^{n+1} = \frac{2}{k_{n+1}}T^n + \frac{\partial T}{\partial t}^n \quad \text{in } \Omega$$

$$T^{n+1} = T_g^{n+1} \quad \text{on } \Gamma_D$$

$$\nu\nabla T^{n+1}\cdot\vec{n} = 0 \quad \text{on } \Gamma_N.$$

Linearization

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

$$\frac{2}{k_{n+1}}\vec{u}^{n+1} - \nu\nabla^2\vec{u}^{n+1} + \vec{w}^{n+1}\cdot\nabla\vec{u}^{n+1} + \nabla p^{n+1} - \vec{j}T^{n+1} = \frac{2}{k_{n+1}}\vec{u}^n + \frac{\partial\vec{u}}{\partial t}^n \quad \text{in } \Omega$$

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

$$\vec{u}^{n+1} = \vec{0} \quad \text{on } \Gamma.$$

$$\frac{2}{k_{n+1}}T^{n+1} - \nu\nabla^2 T^{n+1} + \vec{w}^{n+1}\cdot\nabla T^{n+1} = \frac{2}{k_{n+1}}T^n + \frac{\partial T}{\partial t}^n \quad \text{in } \Omega$$

$$T^{n+1} = T_g^{n+1} \quad \text{on } \Gamma_D$$

$$\nu\nabla T^{n+1}\cdot\vec{n} = 0 \quad \text{on } \Gamma_N,$$

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

Adaptive Time Stepping TR–AB2

The adaptive time step selection is based on coupled physics.

Given L_2 error estimates $\| \vec{e}_h^{\,n+1} \|$ and $\| e_h^{n+1} \|$ for the velocity and temperature respectively, the subsequent TR–AB2 time step k_{n+2} is computed using

$$k_{n+2} = k_{n+1} \left(\frac{\varepsilon_t}{\sqrt{\| \vec{e}_h^{\,n+1} \|^2 + \| e_h^{n+1} \|^2}} \right)^{1/3}.$$

Adaptive time stepping components

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

$$T(\vec{x}, t) = (1 - e^{-10t})\, T_\infty \quad \text{on } \Gamma_D \times [0, T].$$

  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 $T(\vec{x}, t) = T_\infty$, we model the impulse with a time-dependent boundary condition:

$$T(\vec{x}, t) = (1 - e^{-10t})\, T_\infty \quad \text{on } \Gamma_D \times [0, T].$$

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

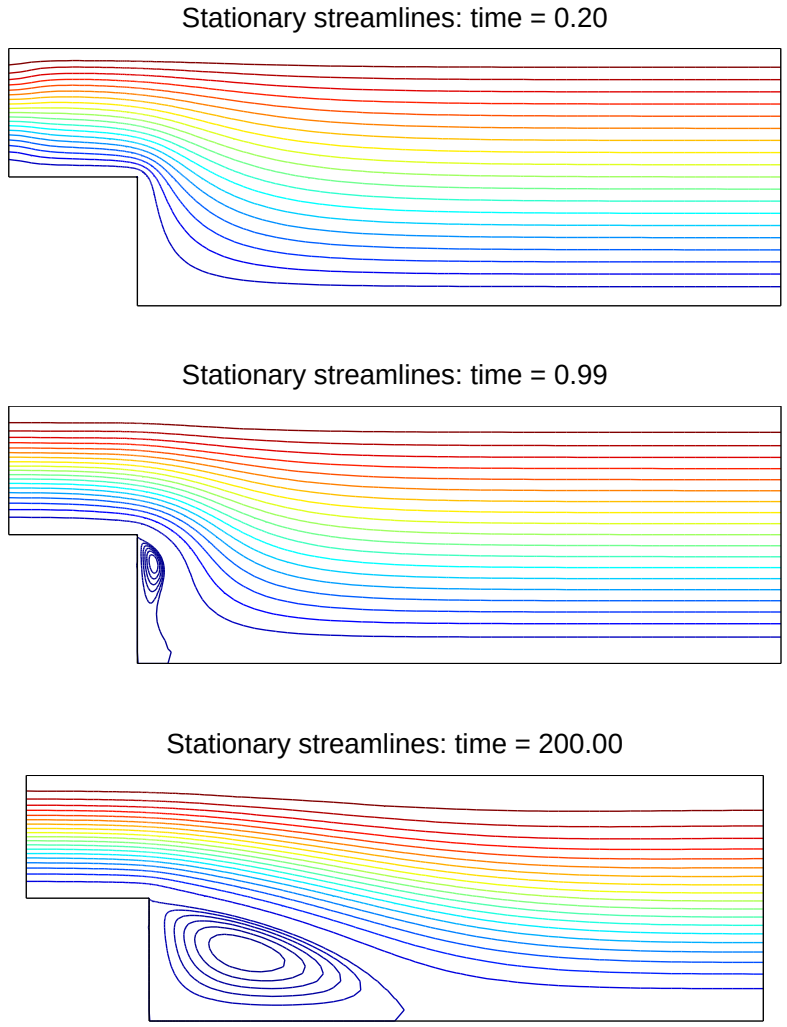
- The following parameters must be specified:

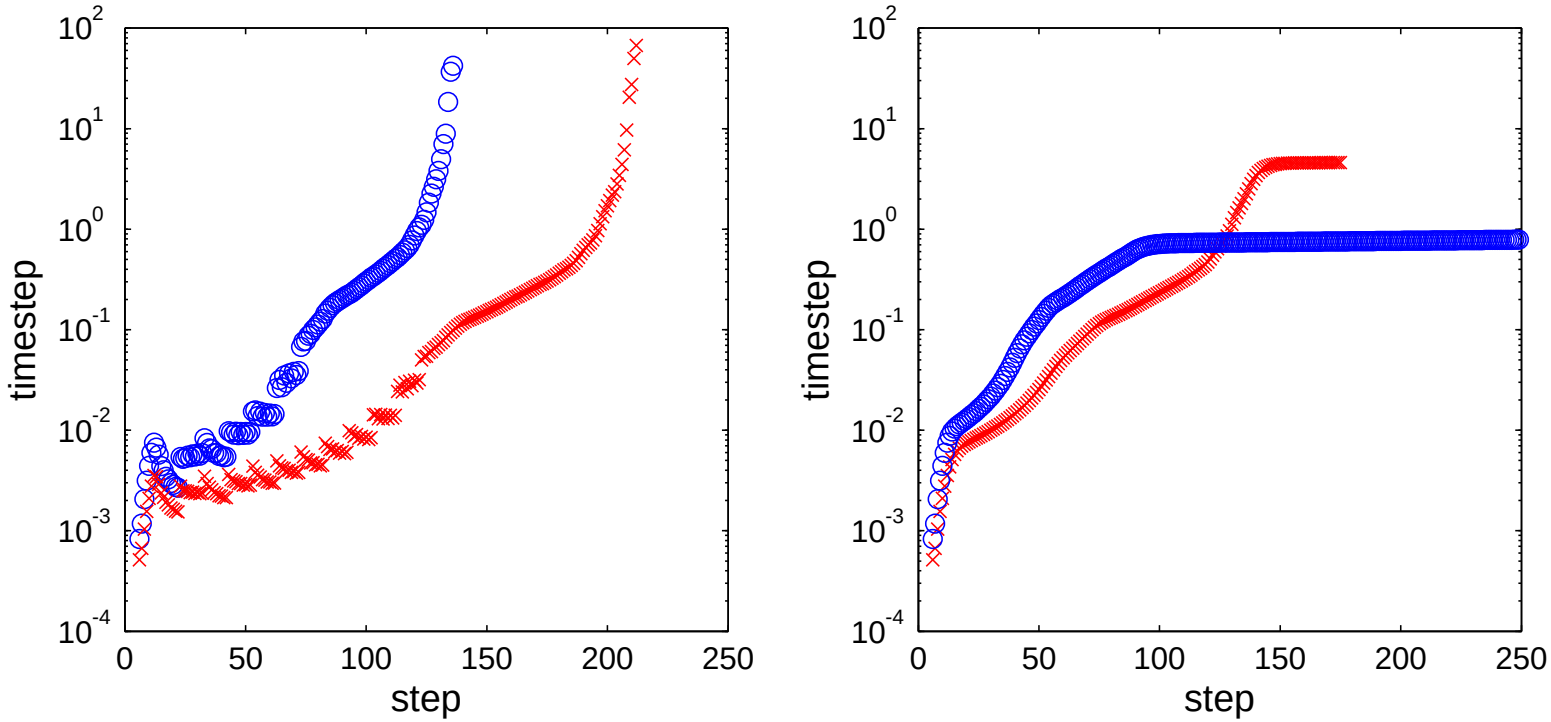
time accuracy tolerance	ε_t (10^{-5})
GMRES tolerance	`itol` (10^{-6})
GMRES iteration limit	`maxit` (50)

PROBLEM # 1

Vanilla TR is not L–stable

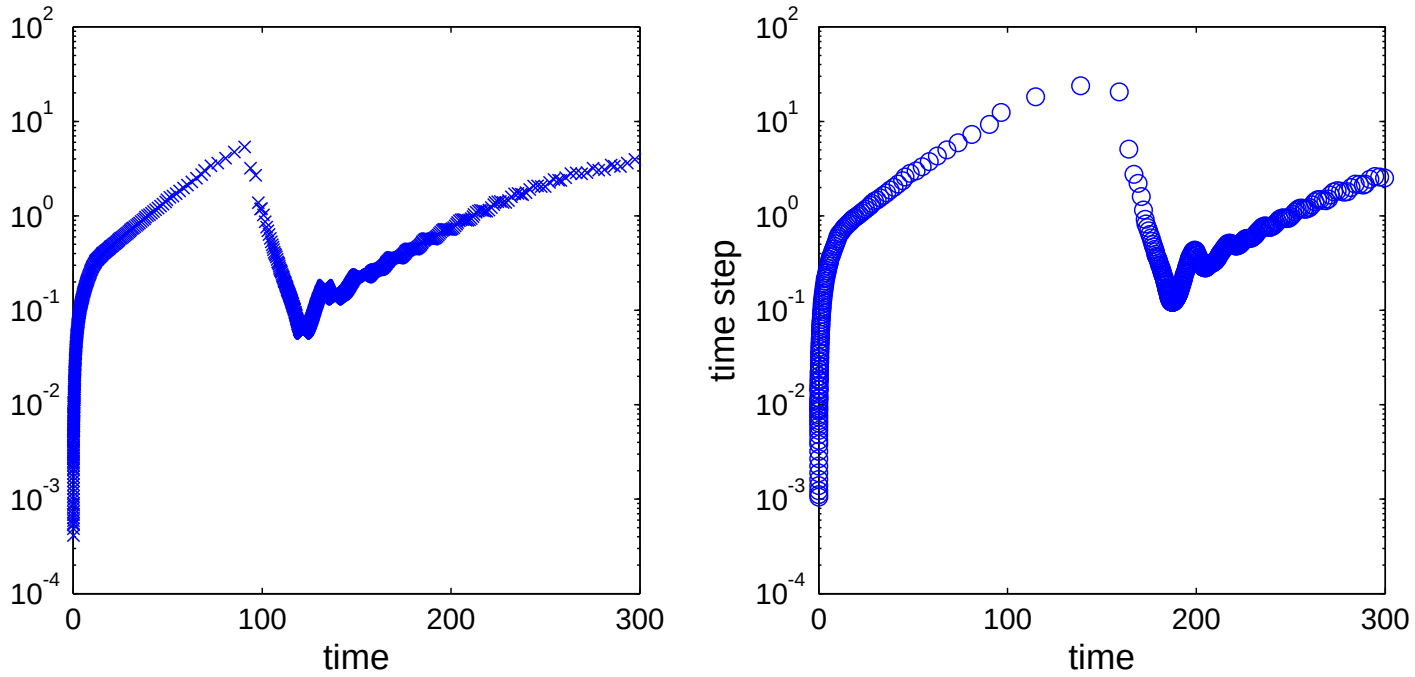
Isothermal flow over step | $Re = 100$.





stabilized TR with $n_* = 10$ (left) vs unstabilized TR (right) for error tolerance $\varepsilon_t = 10^{-4}$ (○) and $\varepsilon_t = 3 \times 10^{-5}$ (x).

Rayleigh–Bénard | $Pr = 7.1, \quad Ra = 1.5 \times 10^4$.

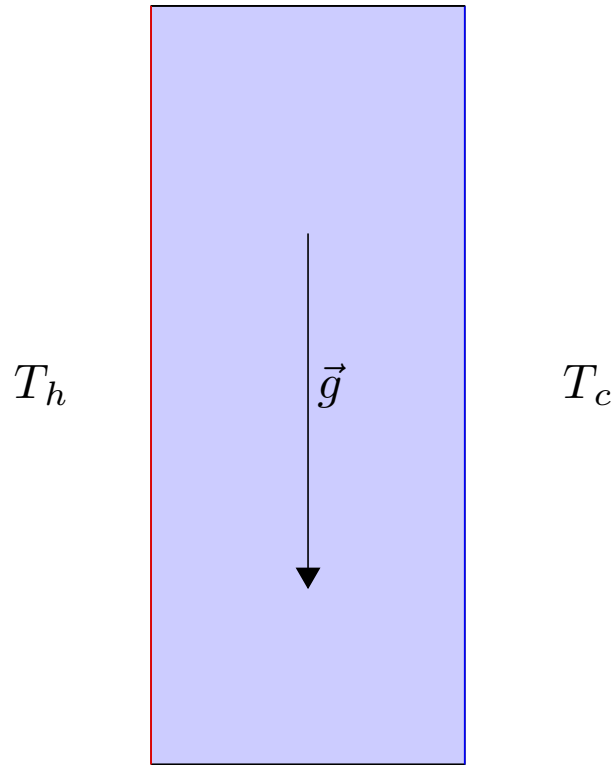


stabilized TR | $\varepsilon_t = 10^{-6}$ (left) and $\varepsilon_t = 10^{-5}$ (right).

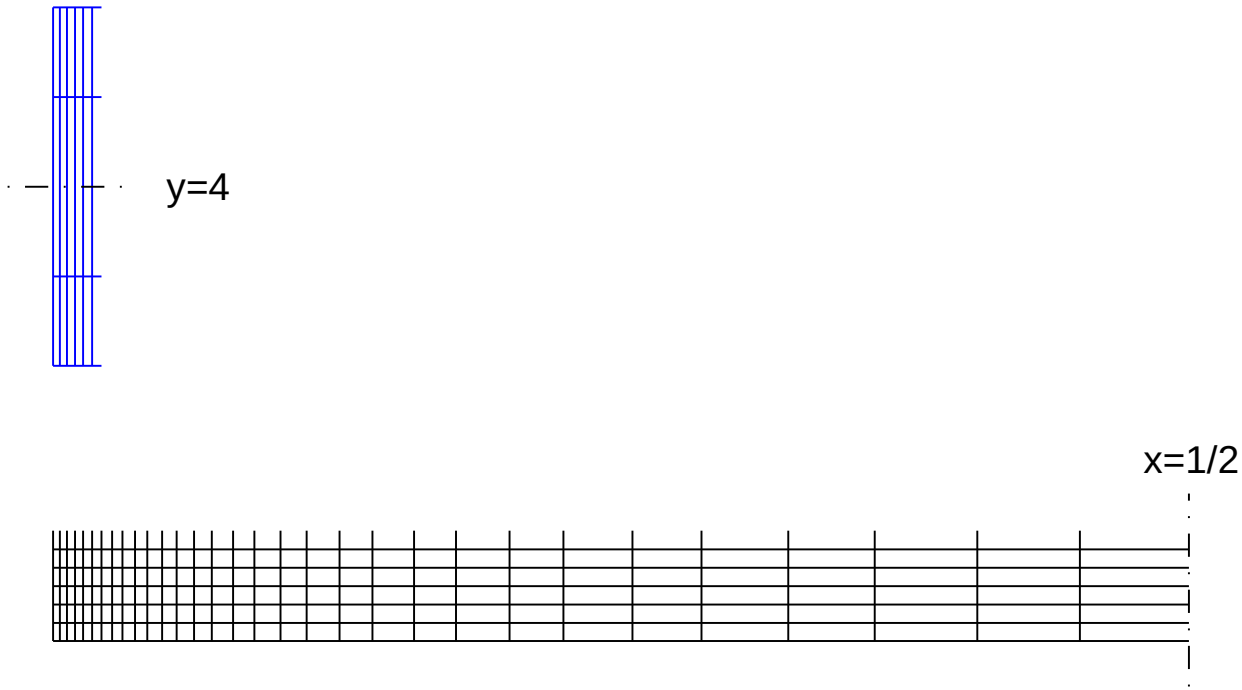
PROBLEM # 2

linearized TR is not coupled physics

MIT test problem | $Pr = 0.71, \quad Ra = 3.4 \times 10^5$.

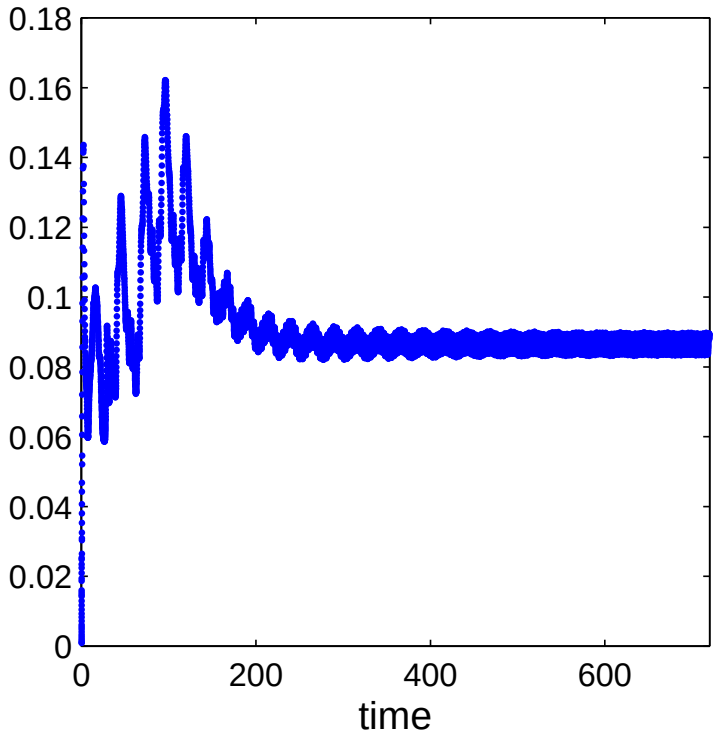


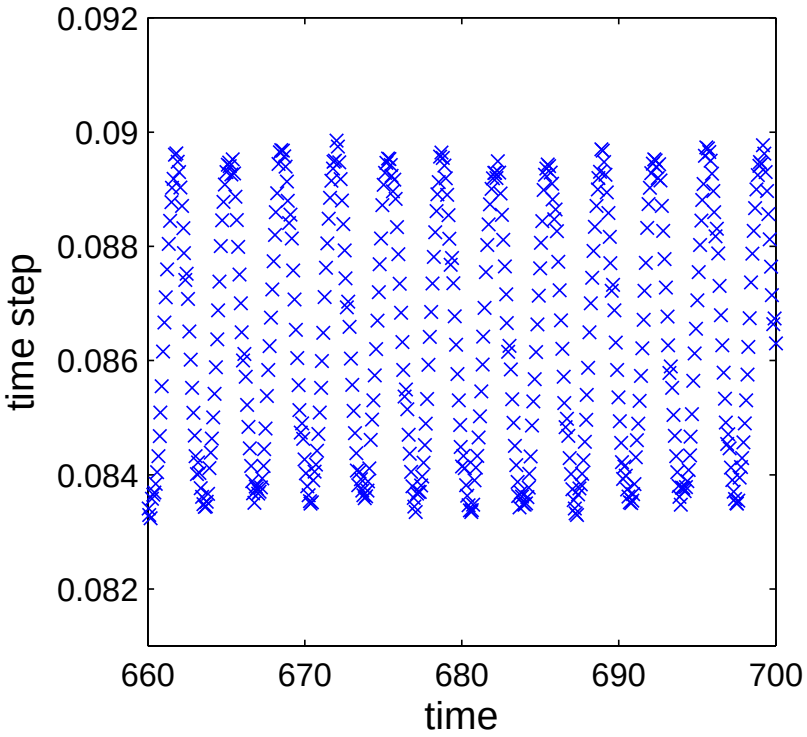
MIT test problem | $Pr = 0.71, \quad Ra = 3.4 \times 10^5$.



31×248 stretched grid

MIT test problem | $Pr = 0.71, \quad Ra = 3.4 \times 10^5$.

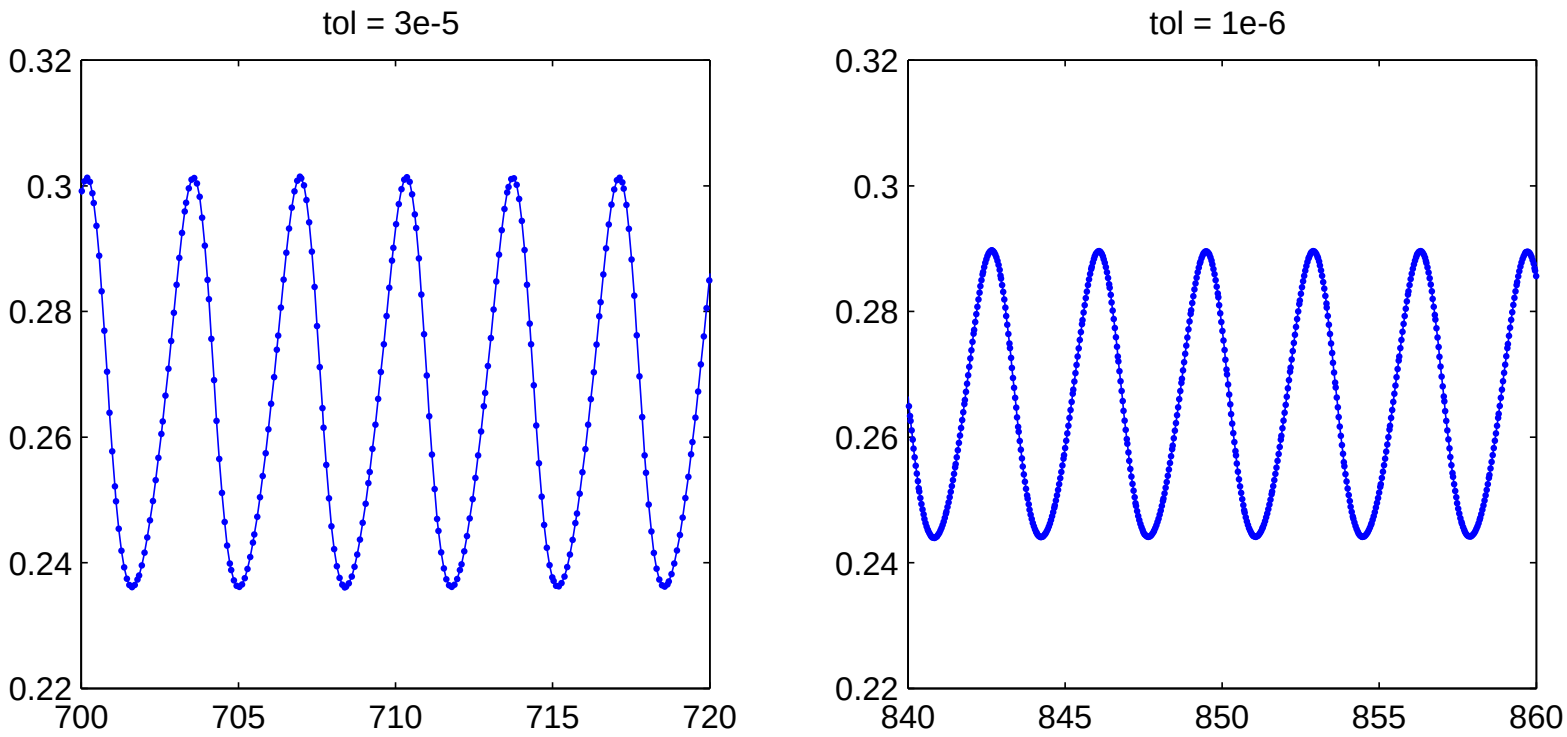




MIT test problem | $Pr = 0.71, \quad Ra = 3.4 \times 10^5$.

	MIT Benchmark	$\varepsilon_t = 3 \cdot 10^{-5}$	$\varepsilon_t = 1 \cdot 10^{-6}$
$(\Delta p)_{\min}$	-0.0125	-0.0178	-0.0135
$(\Delta p)_{\max}$	0.0074	0.0116	0.0082
$\Delta(\Delta p)$	0.0198	0.0294	0.0218
$\overline{\Delta p}$	-0.0026	-0.0031	-0.0027
$T_{\min}$	0.2461	0.2362	0.2442
$T_{\max}$	0.2872	0.3012	0.2896
ΔT	0.0411	0.0650	0.0454
$\overline{T}$	0.2666	0.2687	0.2669
Period	3.4135	3.382	3.412

MIT test problem | $Pr = 0.71, \quad Ra = 3.4 \times 10^5$.



Temperature evolution at the MIT reference point.

PART II

"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

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

gives the method-of-lines discretized system:

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

with a (vertical–) mass matrix:

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

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

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

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

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

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

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

aspentech

Science & Technology Facilities Council

HSL HSL_MI20

PACKAGE SPECIFICATION HSL 2007

1 SUMMARY

Given an $n \times n$ sparse matrix $\mathbf{A}$ and an $n-$vector $\mathbf{z}$, HSL_MI20 computes the vector $\mathbf{x} = \mathbf{M}\mathbf{z}$, where $\mathbf{M}$ is an algebraic multigrid (AMG) v-cycle preconditioner for $\mathbf{A}$. A classical AMG method is used, as described in [1] (see also Section 5 below for a brief description of the algorithm). The matrix $\mathbf{A}$ 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.

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

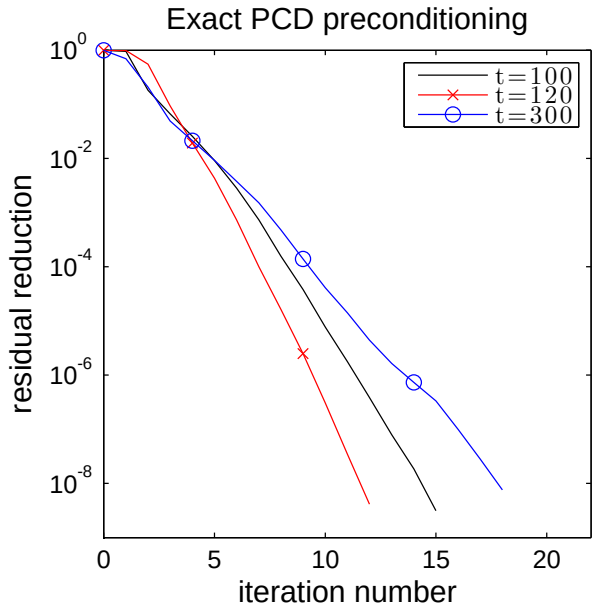
Introducing associated pressure matrices

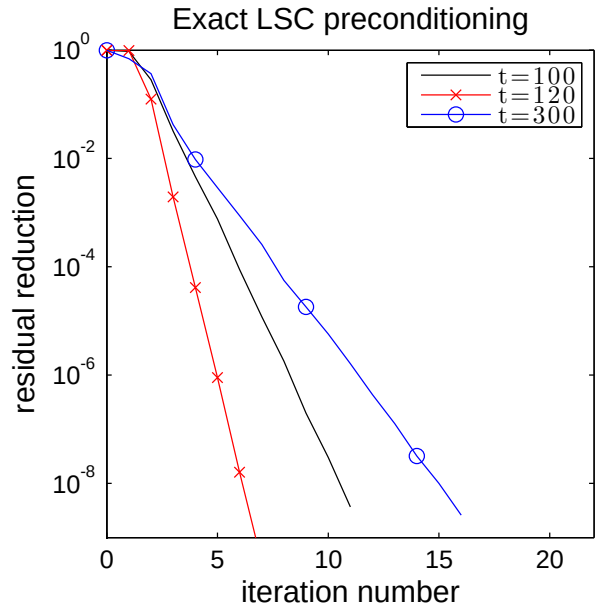
$$
\begin{aligned}
M_p &\sim (\nabla\psi_i, \nabla\psi_j), && \text{mass} \\
A_p &\sim (\nabla\psi_i, \nabla\psi_j), && \text{diffusion} \\
N_p &\sim (\,\vec{w}_h \cdot \nabla\psi_i, \psi_j), && \text{convection} \\
F_p &= \frac{2}{k_{n+1}} M_p + \nu A_p + N_p, && \text{convection-diffusion}
\end{aligned}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

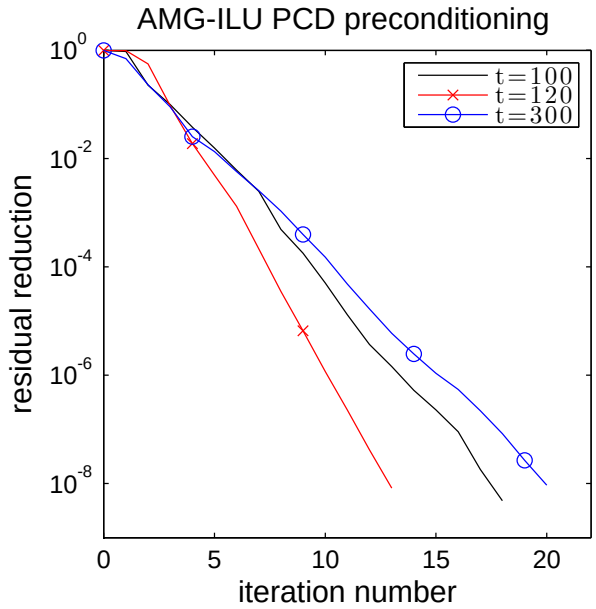
gives the "pressure convection-diffusion preconditioner":

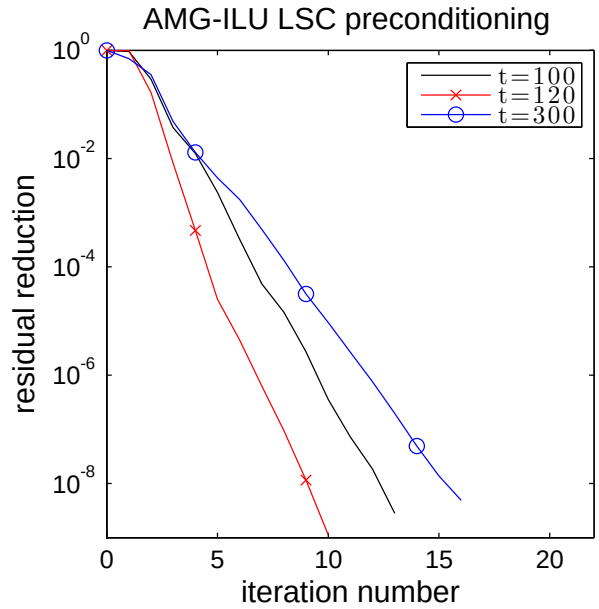
$$
(BF^{-1}B^T)^{-1} \approx M_p^{-1} F_p \underbrace{A_p^{-1}}_{amg}
$$

Rayleigh–Bénard | $Pr = 7.1, \quad Ra = 1.5 \times 10^4$.

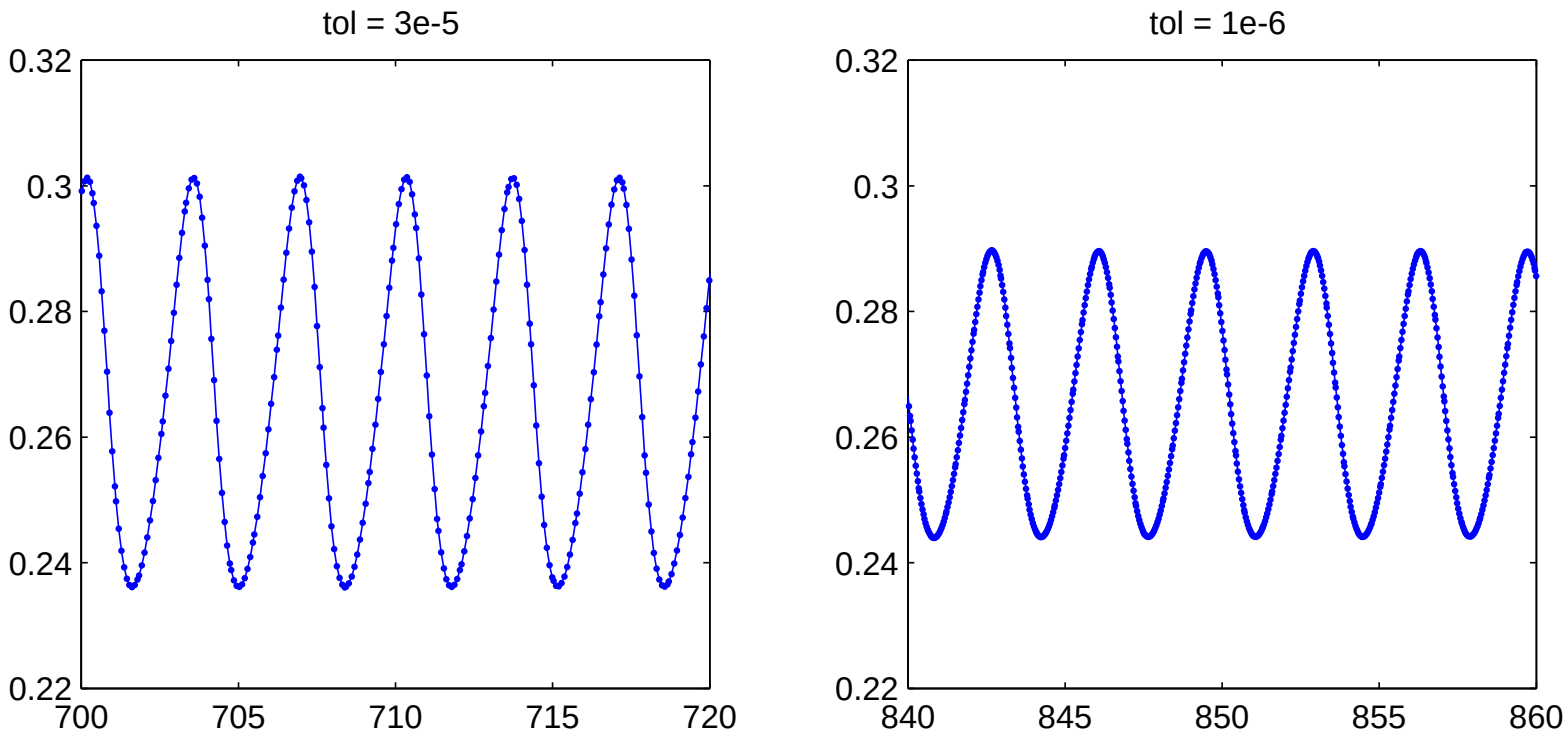






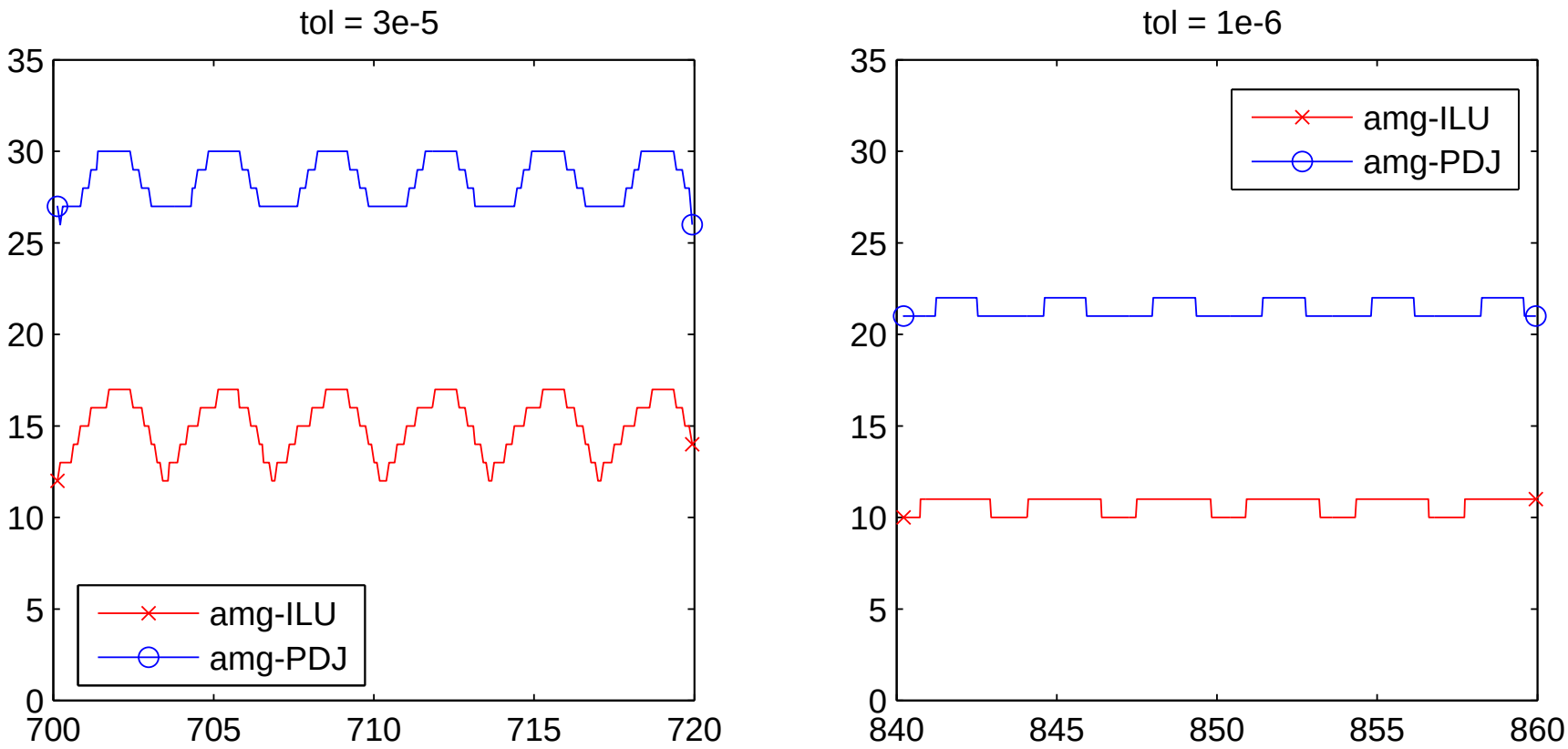


MIT test problem | $Pr = 0.71, \quad Ra = 3.4 \times 10^5$.



Temperature evolution at the MIT reference point.

MIT test problem | $Pr = 0.71, \quad Ra = 3.4 \times 10^5$.



Iteration counts using inexact PCD preconditioning.

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 diffusion parameters ν and ν.

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 diffusion parameters ν and ν.

- Silvester, D., Bespalov, A. and Powell, C.
  A framework for the development of implicit solvers for incompressible flow problems
  Discrete and Continuous Dynamical Systems — Series S (DCDS–S), 5:1195–1221, 2012.