M\crNA
Manchester Numerical Analysis

# Uncertainty Quantification: Does it need efficient linear algebra?

David Silvester
University of Manchester, UK

Yes.

# part I | 1991

- Incompressible flow: Navier–Stokes equations
  - fully implicit schemes and adaptive time stepping

# part I | 1991

- Incompressible flow: Navier–Stokes equations
  - fully implicit schemes and adaptive time stepping

- joint work with David Griffiths (University of Dundee)

# part II | 2011

- PDEs with random data
  - stochastic Galerkin approximation methods

- joint work with Catherine Powell (University of Manchester)

# 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

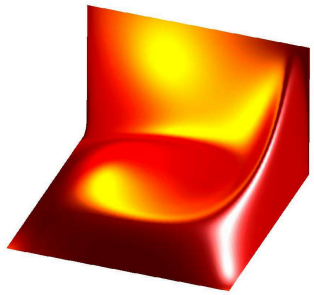
Documentation

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**.

# PART I

# References I


- 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.

# 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

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

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

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

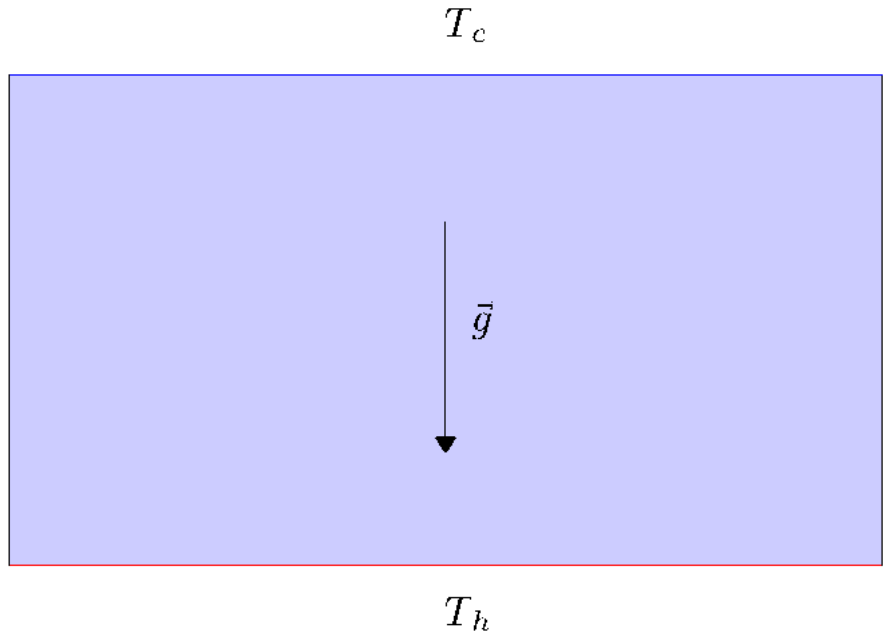
Boundary and initial conditions

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

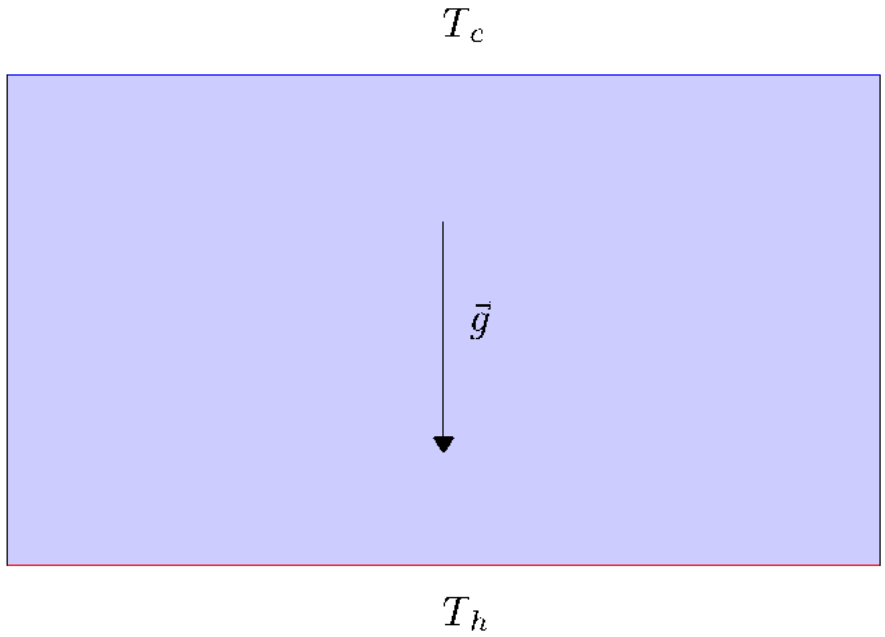
$$T = T_g \quad \text{on } \Gamma_D \times [0, T]; \qquad \nu \nabla T \cdot \vec{n} = 0 \quad \text{on } \Gamma_N \times [0, T];$$

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

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

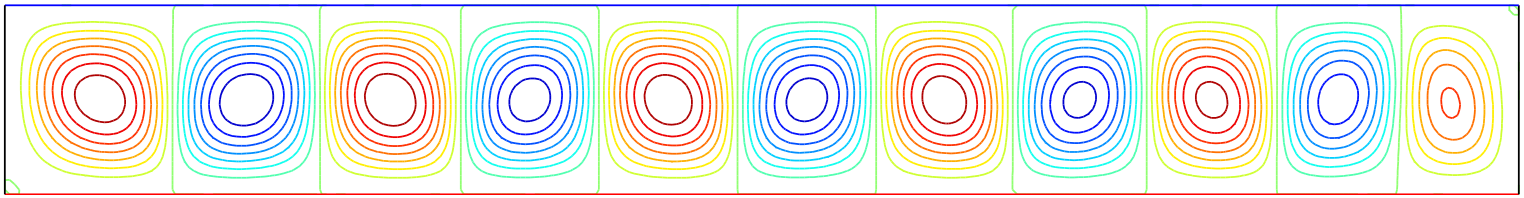



Rayleigh–Bénard $\quad | \quad Pr = 7.1, \quad Ra = 15000\ .$




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

Stationary streamlines: time = 300.00

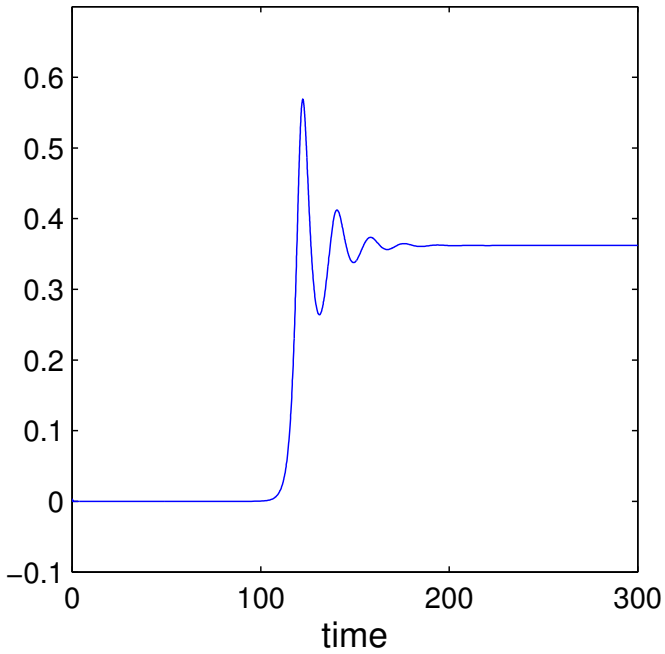


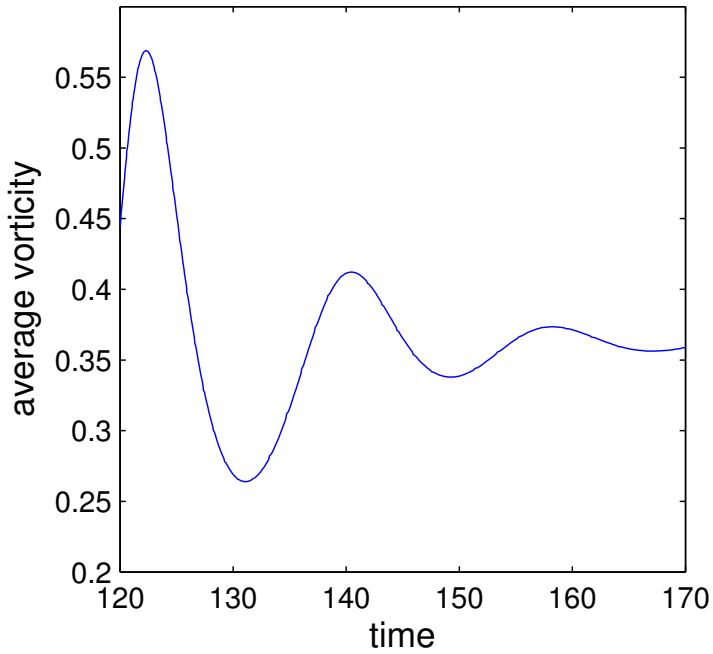
# "Smart Integrator" (SI)

- 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.

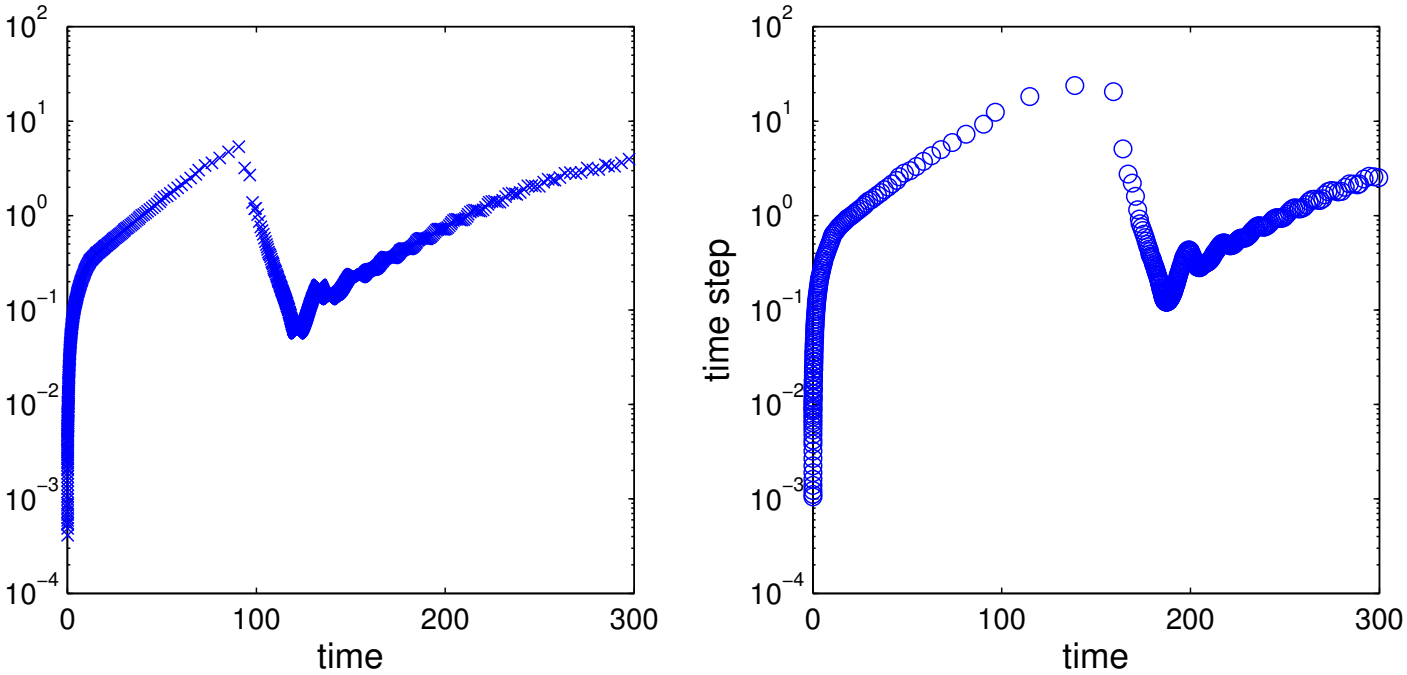
## Rayleigh–Bénard | $Pr = 7.1, \quad Ra = 1.5 \times 10^4$ .




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

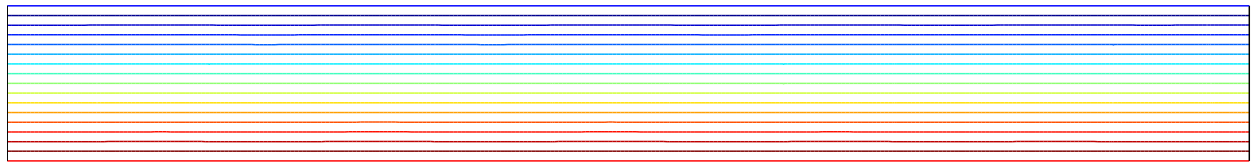
Rayleigh–Bénard | $Pr = 7.1, \quad Ra = 1.5 \times 10^4$ .



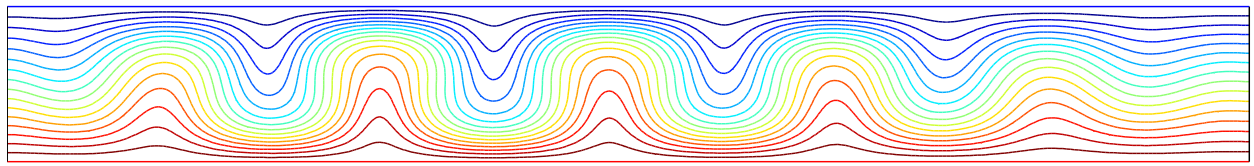

stabilized TR | $\varepsilon_t = 10^{-6}$ (left) and $\varepsilon_t = 10^{-5}$ (right).

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

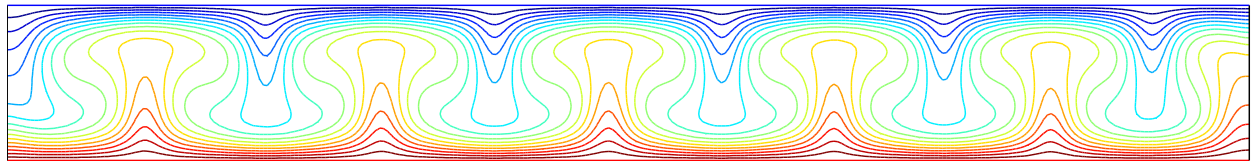
Isotherms: time = 100.72



Isotherms: time = 119.28



Isotherms: time = 300.00



# LINEAR ALGEBRA

# Trapezoidal Rule (TR) time discretization

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

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

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

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

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

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

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

# Linearization

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

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

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

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

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

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

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

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 components

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

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

The following parameters must be specified:

| | |
|---|---|
| time accuracy tolerance | $\varepsilon_t$ $(10^{-5})$ |
| GMRES tolerance | `itol` $(10^{-6})$ |
| GMRES iteration limit | `maxit` $(50)$ |

# 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:

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

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

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

See Chapter 11 of

- Howard Elman & David Silvester & Andrew Wathen
  Finite Elements and Fast Iterative Solvers: with applications in incompressible fluid dynamics
  Oxford University Press, second edition, 2014.

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 $\mathbf{x} = \mathbf{Mz}$, 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

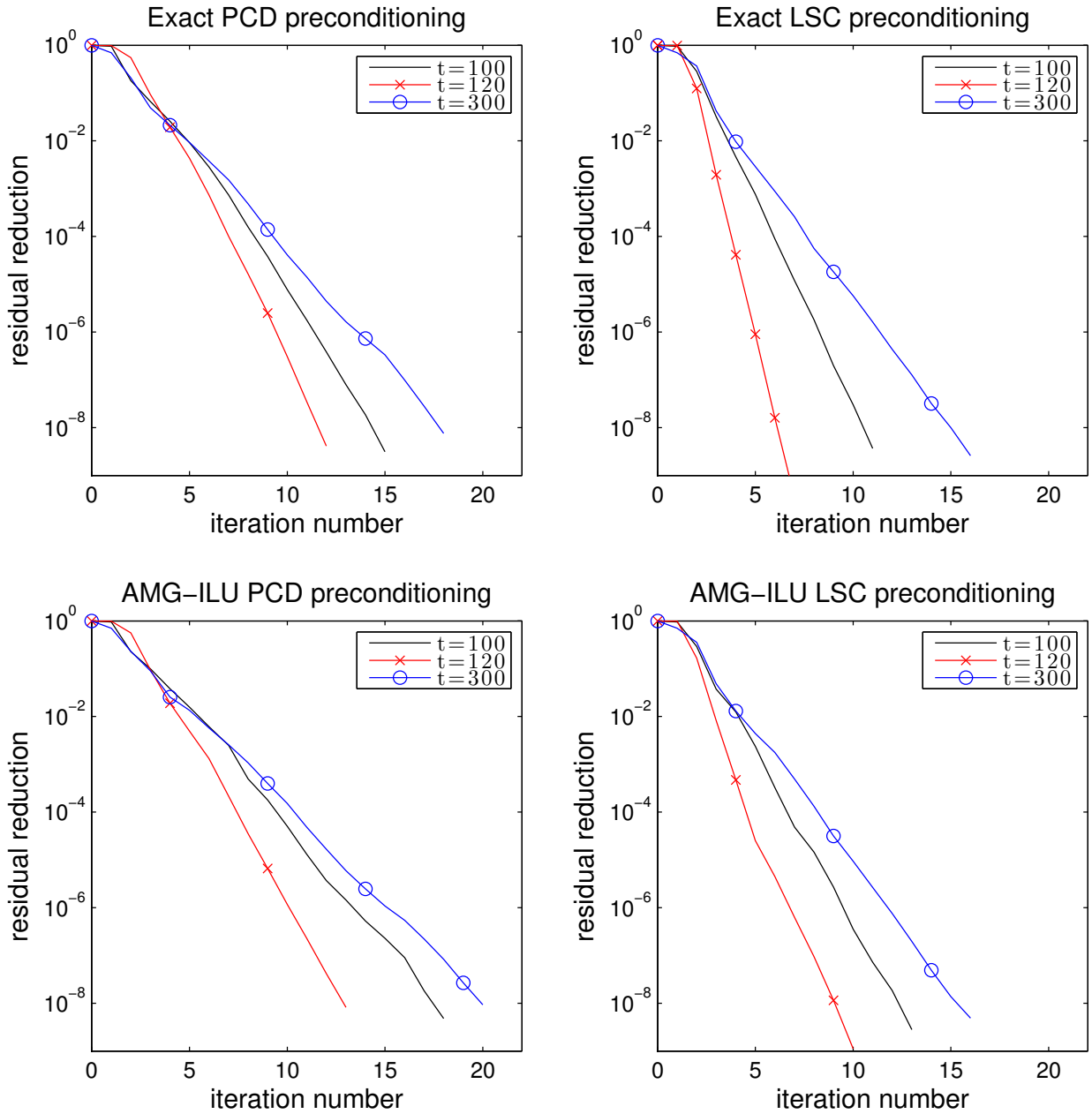
Introducing associated pressure matrices

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

gives the "pressure convection-diffusion preconditioner":

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

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



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 $\nu$ and $\nu$.

# PART II

# References II


- Catherine Powell & David Silvester
  Preconditioning steady-state Navier–Stokes equations with random data. SIAM J. Scientific Computing, vol. 34, A2482–A2506, 2012.
- David Silvester & Alex Bespalov & Catherine Powell
  A framework for the development of implicit solvers for incompressible flow problems. Discrete and Continuous Dynamical Systems — Series S, vol. 5, 1195–1221, 2012.

# Steady-state flow with random data

*Problem statement*

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

We model uncertainty in the viscosity as

$$\nu(\omega) = \mu + \sigma \xi(\omega).$$

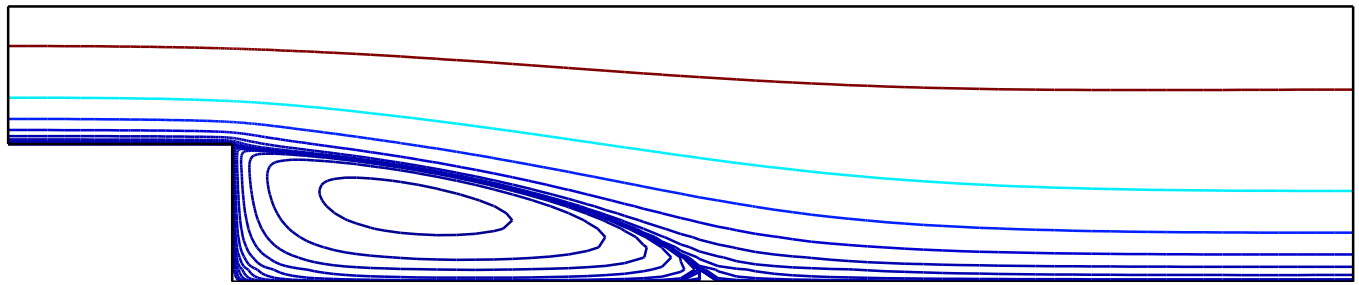
If $\xi \sim U(-\sqrt{3}, \sqrt{3})$, then $\nu$ is a uniform random variable with

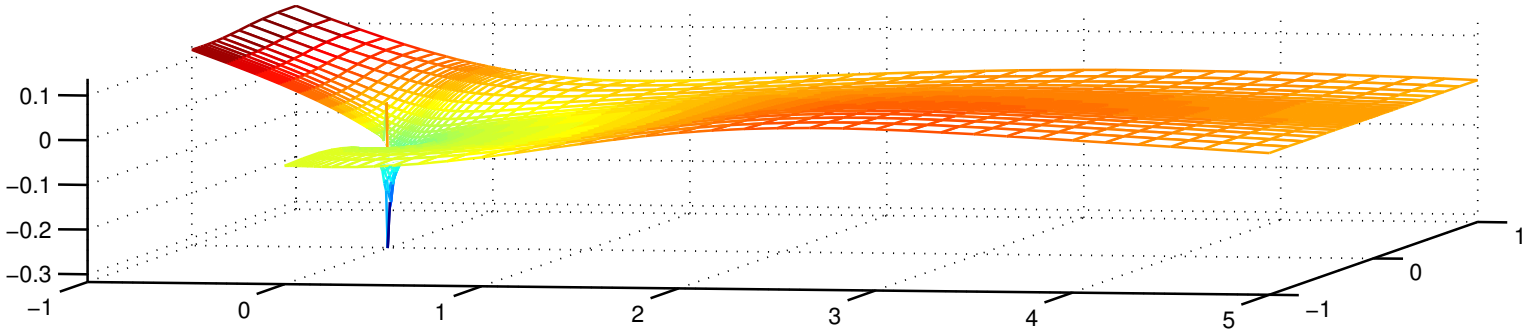
$$\mathbb{E}[\nu(\omega)] = \mu, \qquad \mathsf{Var}[\nu(\omega)] = \sigma^2.$$

# N–S example I: flow over a step

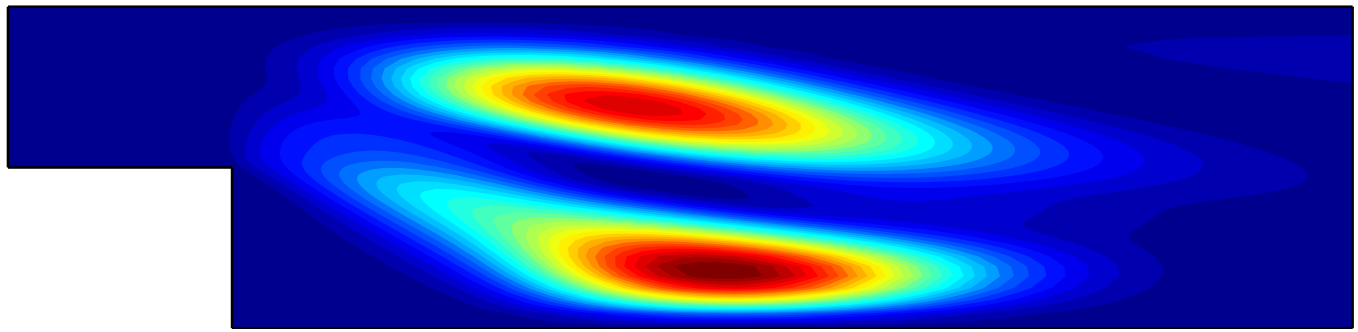
Streamlines of the mean flow field (top) and plot of the mean pressure field (bottom):

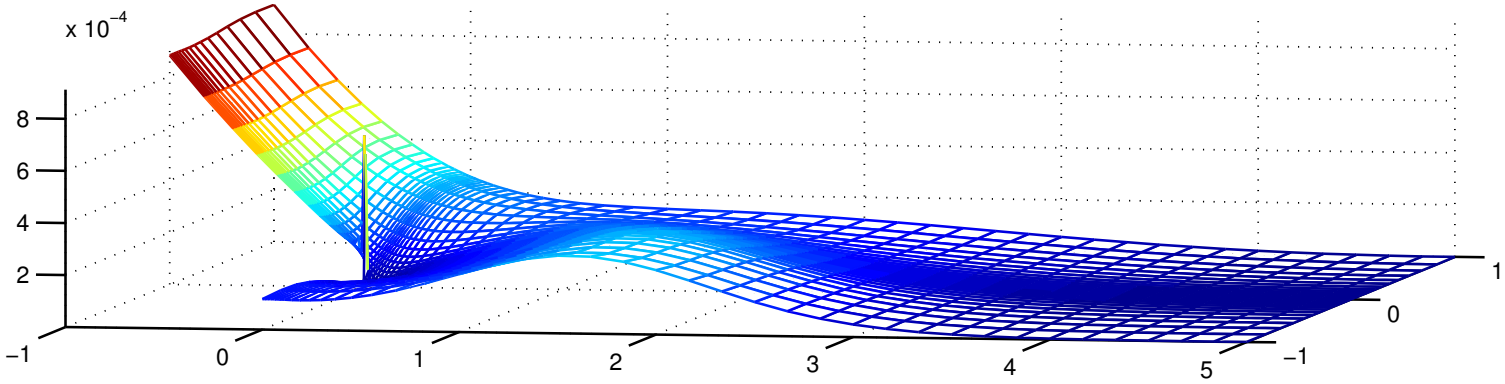
$$\mu = 1/50, \quad \sigma = \mu/10$$





Variance of the magnitude of flow field (top) and variance of the pressure (bottom)





# Stochastic discretisation methods

- Monte Carlo Methods
- Perturbation Methods
- Stochastic Galerkin Methods
- Stochastic Collocation Methods
- Stochastic Reduced Basis Methods
- ...

# Stochastic discretisation methods

- Monte Carlo Methods
- Perturbation Methods
- Stochastic Galerkin Methods
- Stochastic Collocation Methods
- Stochastic Reduced Basis Methods
- ...

*Key points*

- If the number of random variables describing the input data is small then Stochastic Galerkin and Stochastic Collocation methods can outperform Monte Carlo.
- If software for the deterministic problem is to be useful for Stochastic Galerkin approximation then specialised solvers need to be developed.

# LINEAR ALGEBRA

# Stochastic Galerkin discretisation I

*Ingredients*

- Picard iteration;
- standard finite element spaces $\mathbf{X}_E^h$ and $M^h$;
- a suitable finite-dimensional subspace $S^k \subset L^2_\rho(\Lambda)$, where $\Lambda := \xi(\Xi)$, $\Lambda \ni y$.

# Stochastic Galerkin discretisation I

*Ingredients*

- Picard iteration;
- standard finite element spaces $\mathbf{X}_E^h$ and $M^h$;
- a suitable finite-dimensional subspace $S^k \subset L^2_\rho(\Lambda)$, where $\Lambda := \xi(\Xi)$, $\Lambda \ni y$.

*Discrete formulation*

Find $\vec{u}_{hk}^{\,n+1} \in \mathbf{X}_E^h \otimes S^k$ and $p_{hk}^{n+1} \in M^h \otimes S^k$ satisfying:

$$\mathbb{E}\Big[\nu(y)\big(\nabla \vec{u}_{hk}^{\,n+1}, \nabla \vec{v}\big)\Big] + \mathbb{E}\Big[\big(\vec{u}_{hk}^{\,n} \cdot \nabla \vec{u}_{hk}^{\,n+1}, \vec{v}\big)\Big] - \mathbb{E}\Big[\big(p_{hk}^{n+1}, \nabla \cdot \vec{v}\big)\Big] = 0$$

$$\mathbb{E}\Big[\big(\nabla \cdot \vec{u}_{hk}^{\,n+1}, q\big)\Big] = 0$$

for all $\vec{v} \in \mathbf{X}_0^h \otimes S^k$ and $q \in M^h \otimes S^k$.

# Stochastic Galerkin discretisation II

*Discrete formulation*

Find $\vec{u}_{hk}^{n+1} \in \mathbf{X}_E^h \otimes S^k$ and $p_{hk}^{n+1} \in M^h \otimes S^k$ satisfying:

$$\mathbb{E}\Big[\nu(y)\big(\nabla \vec{u}_{hk}^{n+1}, \nabla \vec{v}\big)\Big] + \mathbb{E}\Big[\big(\vec{u}_{hk}^{n} \cdot \nabla \vec{u}_{hk}^{n+1}, \vec{v}\big)\Big] - \mathbb{E}\Big[\big(p_{hk}^{n+1}, \nabla \cdot \vec{v}\big)\Big] = 0$$

$$\mathbb{E}\Big[\big(\nabla \cdot \vec{u}_{hk}^{n+1}, q\big)\Big] = 0$$

for all $\vec{v} \in \mathbf{X}_0^h \otimes S^k$ and $q \in M^h \otimes S^k$.

# Stochastic Galerkin discretisation II

*Discrete formulation*

Find $\vec{u}_{hk}^{n+1} \in \mathbf{X}_E^h \otimes S^k$ and $p_{hk}^{n+1} \in M^h \otimes S^k$ satisfying:

$$\mathbb{E}\Big[\nu(y)\big(\nabla \vec{u}_{hk}^{n+1}, \nabla \vec{v}\big)\Big] + \mathbb{E}\Big[\big(\vec{u}_{hk}^{n} \cdot \nabla \vec{u}_{hk}^{n+1}, \vec{v}\big)\Big] - \mathbb{E}\Big[\big(p_{hk}^{n+1}, \nabla \cdot \vec{v}\big)\Big] = 0$$

$$\mathbb{E}\Big[\big(\nabla \cdot \vec{u}_{hk}^{n+1}, q\big)\Big] = 0$$

for all $\vec{v} \in \mathbf{X}_0^h \otimes S^k$ and $q \in M^h \otimes S^k$.

*Sets of basis functions*

$\mathbf{X}_0^h = \mathtt{span}\big\{(\phi_i(\vec{x}), 0), (0, \phi_i(\vec{x}))\big\}_{i=1}^{n_u}$; $M^h = \mathtt{span}\,\{\psi_j(\vec{x})\}_{j=1}^{n_p}$;

$S^k = \mathtt{span}\,\{\varphi_\ell(y)\}_{\ell=0}^{k}$.

# Stochastic Galerkin discretisation III

The linear system at the $(n+1)$st Picard iteration is

$$\begin{pmatrix} \mathbb{F}_{\nu}^{n} & \mathbb{B}^{T} \\ \mathbb{B} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}^{n} \\ \boldsymbol{\beta}^{n} \end{pmatrix} = \begin{pmatrix} \mathbf{f}^{n} \\ \mathbf{g}^{n} \end{pmatrix}$$

with

$$\mathbb{F}_{\nu}^{n} = \begin{pmatrix} F_{\nu}^{n} & 0 \\ 0 & F_{\nu}^{n} \end{pmatrix}, \quad \mathbb{B} = \begin{pmatrix} G_0 \otimes B_{x_1} & G_0 \otimes B_{x_2} \end{pmatrix}$$

and

$$F_{\nu}^{n} := (\mu G_0 + \sigma G_1) \otimes A + \sum_{\ell=0}^{k} H_\ell \otimes N_\ell,$$

$B_{x_1}$, $B_{x_2}$ are discrete representations of the first derivatives.

The system dimension is: $(n_u + n_p)(k+1) \times (n_u + n_p)(k+1)$.

(1-1) block: $F_\nu^n := (\mu G_0 + \sigma G_1) \otimes A + \sum_{\ell=0}^{k} H_\ell \otimes N_\ell$.

- $F_\nu^n$ is a non-symmetric matrix.
- convection matrices $N_\ell$ $(\ell = 0, \ldots, k)$ are given by

$$[N_\ell]_{ij} = (\vec{u}_{h\ell}^{\,n}(\vec{x}) \cdot \nabla \phi_i, \phi_j) \quad i, j = 0, \ldots, n_u.$$

where $\vec{u}_{h\ell}^{\,n}$ are the 'spatial coefficients' in the expansion of the lagged velocity field,

$$\vec{u}_{hk}^{\,n}(\vec{x}, y) = \sum_{\ell=0}^{k} \left( \underbrace{\sum_{i=1}^{n_u} \vec{u}_{i\ell}^{\,n} \, \phi_i(\vec{x})}_{\vec{u}_{h\ell}^{\,n}(\vec{x})} \right) \varphi_\ell(y).$$

(1-1) block: $F_\nu^n := (\mu G_0 + \sigma G_1) \otimes A + \sum_{\ell=0}^{k} H_\ell \otimes N_\ell$.

- $F_\nu^n$ is a non-symmetric matrix.
- convection matrices $N_\ell$ $(\ell = 0, \ldots, k)$ are given by

$$[N_\ell]_{ij} = (\vec{u}_{h\ell}^{\,n}(\vec{x}) \cdot \nabla \phi_i, \phi_j) \quad i, j = 0, \ldots, n_u.$$

- $G_0$, $G_1$ and $H_\ell$ are all $(k+1) \times (k+1)$ matrices:

$$\begin{aligned} G_0 := [G_0]_{\ell s} &= \mathbb{E}\left[\varphi_s(y)\, \varphi_\ell(y)\right], \\ G_1 := [G_1]_{\ell s} &= \mathbb{E}\left[y\, \varphi_s(y)\, \varphi_\ell(y)\right], \\ H_\ell := [H_\ell]_{ms} &= \mathbb{E}\left[\varphi_\ell(y)\, \varphi_s(y)\, \varphi_m(y)\right]. \end{aligned}$$

(1-1) block: $F_\nu^n := (\mu G_0 + \sigma G_1) \otimes A + \sum_{\ell=0}^{k} H_\ell \otimes N_\ell$.

- $F_\nu^n$ is a non-symmetric matrix.
- convection matrices $N_\ell$ $(\ell = 0, \ldots, k)$ are given by

$$[N_\ell]_{ij} = (\vec{u}_{h\ell}^{\,n}(\vec{x}) \cdot \nabla \phi_i, \phi_j) \quad i, j = 0, \ldots, n_u.$$

- $G_0$, $G_1$ and $H_\ell$ are all $(k+1) \times (k+1)$ matrices:

$$\begin{aligned} G_0 := [G_0]_{\ell s} &= \mathbb{E}\left[\varphi_s(y)\, \varphi_\ell(y)\right], \\ G_1 := [G_1]_{\ell s} &= \mathbb{E}\left[y\, \varphi_s(y)\, \varphi_\ell(y)\right], \\ H_\ell := [H_\ell]_{ms} &= \mathbb{E}\left[\varphi_\ell(y)\, \varphi_s(y)\, \varphi_m(y)\right]. \end{aligned}$$

If $\{\varphi_\ell(y)\}_{\ell=0}^{k}$ are scaled Legendre polynomials on $\Lambda$, then

- $G_0 = H_0 = I$, $G_1 = H_1$ is sparse (2 non-zeros per row);
- $H_\ell$ is dense for $\ell \geq 2$.

# Ideal preconditioning

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

An ideal preconditioner is given by

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

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

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

# Preconditioning I

Rearrange the (1-1) block:

$$\begin{aligned} F_\nu^n &= (\mu G_0 + \sigma G_1) \otimes A + \sum_{\ell=0}^{k} H_\ell \otimes N_\ell \\ &= I \otimes (\mu A + N_0) + \sigma G_1 \otimes A + \sum_{\ell=1}^{k} H_\ell \otimes N_\ell \end{aligned}$$

and define

$$F_0 := (\mu A + N_0).$$

# Preconditioning I

Rearrange the (1-1) block:

$$\begin{aligned} F_\nu^n &= (\mu G_0 + \sigma G_1) \otimes A + \sum_{\ell=0}^{k} H_\ell \otimes N_\ell \\ &= I \otimes (\mu A + N_0) + \sigma G_1 \otimes A + \sum_{\ell=1}^{k} H_\ell \otimes N_\ell \end{aligned}$$

and define

$$F_0 := (\mu A + N_0).$$

A natural candidate for $\mathbb{P}_F$ is the block-diagonal mean-based approximation:

$$\mathbb{P}_F = \mathbb{F}_0 := \begin{pmatrix} I \otimes F_0 & 0 \\ 0 & I \otimes F_0 \end{pmatrix}.$$

This is a good approximation when $\frac{\sigma}{\mu}$ is not too large.

# Preconditioning II

Replacing $\mathbb{F}_\nu^n$ by $\mathbb{F}_0$ in the Schur-complement gives

$$\begin{aligned}
\mathbb{S} &\approx \mathbb{B}\mathbb{F}_0^{-1}\,\mathbb{B}^T \\
&= (I \otimes B_{x_1})(I \otimes F_0^{-1})(I \otimes B_{x_1}^T) + (I \otimes B_{x_2})(I \otimes F_0^{-1})(I \otimes B_{x_2}^T) \\
&= I \otimes (B_{x_1}, B_{x_2}) F_0^{-1} (B_{x_1}, B_{x_2})^T =: I \otimes S_0 =: \mathbb{S}_0 = \mathbb{P}_S.
\end{aligned}$$

# Preconditioning II

Replacing $\mathbb{F}_\nu^n$ by $\mathbb{F}_0$ in the Schur-complement gives

$$\begin{aligned}
\mathbb{S} &\approx \mathbb{B}\mathbb{F}_0^{-1}\,\mathbb{B}^T \\
&= (I \otimes B_{x_1})(I \otimes F_0^{-1})(I \otimes B_{x_1}^T) + (I \otimes B_{x_2})(I \otimes F_0^{-1})(I \otimes B_{x_2}^T) \\
&= I \otimes (B_{x_1}, B_{x_2}) F_0^{-1} (B_{x_1}, B_{x_2})^T =: I \otimes S_0 =: \mathbb{S}_0 = \mathbb{P}_S.
\end{aligned}$$

$S_0$ is the Schur-complement corresponding to the deterministic problem with

- viscosity $\mu$
- convection coefficient $\vec{u}_{hk}^{\,0}$ (the mean component of velocity at the previous Picard step)

# Preconditioning III

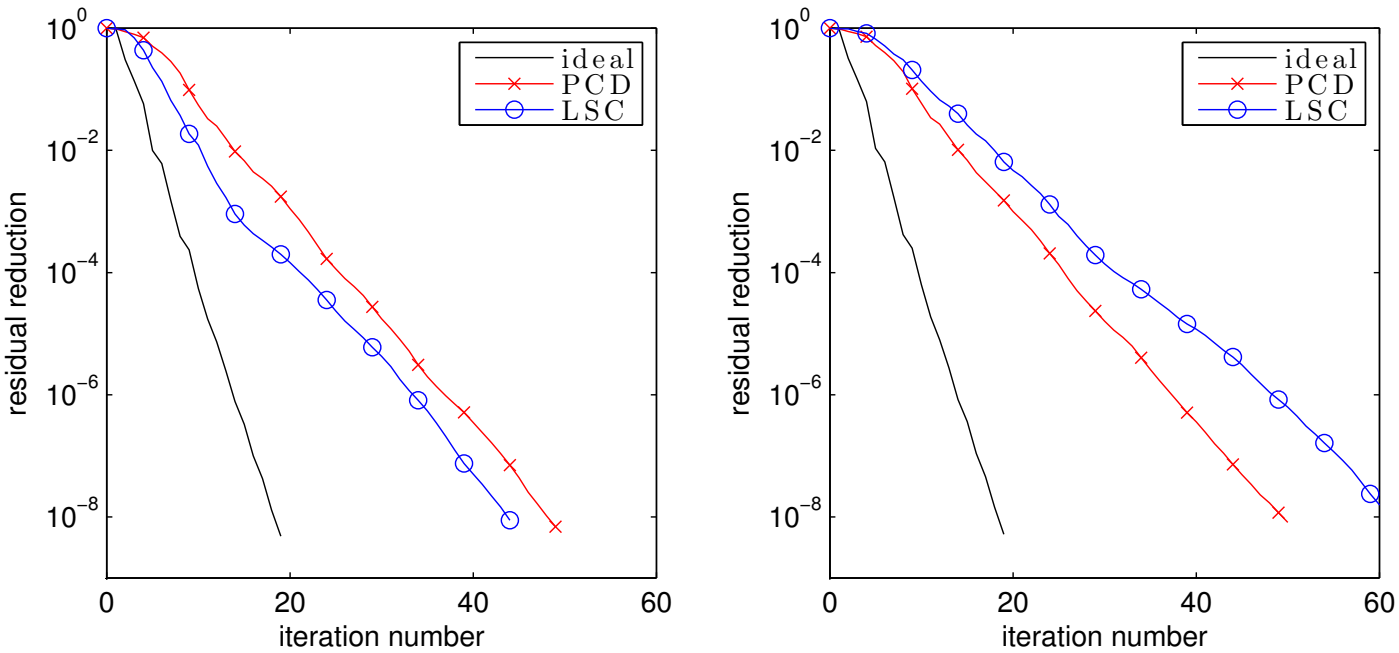
Replacing $\mathbb{F}_\nu^n$ by $\mathbb{F}_0$ in the Schur-complement gives

$$\begin{aligned}\mathbb{S} &\approx \mathbb{B}\mathbb{F}_0^{-1}\mathbb{B}^T \\ &= (I\otimes B_{x_1})(I\otimes F_0^{-1})(I\otimes B_{x_1}^T)+(I\otimes B_{x_2})(I\otimes F_0^{-1})(I\otimes B_{x_2}^T) \\ &= I\otimes(B_{x_1},B_{x_2})F_0^{-1}(B_{x_1},B_{x_2})^T =: I\otimes S_0 =: \mathbb{S}_0 = \mathbb{P}_S.\end{aligned}$$

To apply $\mathbb{P}_S^{-1}$ in each GMRES iteration requires $(k+1)$ solves with $S_0$. This can be done

- exactly (ideal preconditioner); or
- inexactly with the deterministic approaches:
  – pressure convection–diffusion approximation (PCD)
  – least–squares commutator approximation (LSC).

# Flow over a step




GMRES convergence for a coarsened grid (left) and for a reference grid (right) ($\mu = 1/50$; $\sigma = 2\mu/10$).

# Typical GMRES iteration counts

| | | | Coarse grid | | | Fine grid | | |
|---|---|---|---|---|---|---|---|---|
| | | $\mathbb{E}[Re]$ | $k=2$ | 4 | 6 | $k=2$ | 4 | 6 |
| Ideal | $\sigma = \mu/10$ | 67 | 14 | 14 | 14 | 14 | 14 | 15 |
| | $\sigma = 2\mu/10$ | 70 | 18 | 20 | 21 | 14 | 20 | 21 |
| | $\sigma = 3\mu/10$ | 74 | 25 | 28 | 29 | 25 | 28 | 29 |
| PCD | $\sigma = \mu/10$ | 67 | 37 | 38 | 39 | 37 | 39 | 39 |
| | $\sigma = 2\mu/10$ | 70 | 43 | 44 | 50 | 44 | 48 | 50 |
| | $\sigma = 3\mu/10$ | 74 | 53 | 56 | 61 | 54 | 58 | 62 |

What have we achieved?

- ♡ 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.
- ♡ Efficient linear algebra: convergence rate is independent of $h$. Convergence is also robust with respect to the spectral approximation parameter $k$ as long as the variance is not too large relative to the mean.

What have we achieved?

- ♡ 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.
- ♡ Efficient linear algebra: convergence rate is independent of $h$. Convergence is also robust with respect to the spectral approximation parameter $k$ as long as the variance is not too large relative to the mean.

Find out for yourself ...

- (S)IFISS MATLAB Toolbox

# What is the payoff?

What is the payoff?
Efficient $h$-$p$ adaptivity ...

- Alex Bespalov, Catherine Powell & David Silvester.
  A posteriori error estimation for parametric operator equations with applications to PDEs with random data.
  SIAM J. Sci. Comput, 36:A339–A363, 2014.
- Alex Bespalov & David Silvester.
  Efficient adaptive stochastic Galerkin methods for parametric operator equations.
  SIAM J. Sci. Comput, 38:A2118–A2140, 2016.