DNS of Turbulent Natural Convection in a Differentially Heated Cavity of Aspect Ratio 4 at Rayleigh numbers $6.4 \times 10^8$, $2 \times 10^9$ and $10^{10}$

F.X. Trias, M. Soria, O. Lehmkuhl, A. Oliva

Centre Tecnològic de Transferència de Calor (CTTC), Universitat Politècnica de Catalunya
C/ Colom 11, 08222 Terrassa, Barcelona, Spain, E-mail: cttc@cttc.upc.edu
Presentation outline

1. Numerical Methods
   - Governing Equations
   - Pressure-velocity coupling algorithm
   - Symmetry-preserving spatial discretization
   - Poisson solver

2. DNS of turbulent natural convection
   - Problem definition: Differentially Heated Cavity
   - Test cases
   - Flow dynamics
   - Time-averaged flows
   - Second-order statistics

3. Conclusions and Future Research
   - Conclusions
   - Future research
Governing equations

Incompressible Navier-Stokes coupled with energy transport equation:

\[
\nabla \cdot \mathbf{u} = 0
\]
\[
\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \frac{Pr}{Ra^{0.5}} \nabla^2 \mathbf{u} - \nabla p + f
\]
\[
\partial_t T + (\mathbf{u} \cdot \nabla) T = \frac{1}{Ra^{0.5}} \nabla^2 T
\]

where \( f = (0, 0, Ra Pr T) \) (Boussinesq approximation)

Assuming that the fluid is incompressible, Newtonian, of constant physical properties, using Boussinesq approximation to account for the density variations and neglecting thermal radiation.
Pressure-velocity coupling (1/2)

Momentum equation is expressed as:

\[
\frac{\partial \mathbf{u}}{\partial t} = \mathbf{R}(\mathbf{u}) - \frac{1}{\rho} \nabla p
\]

Where \( \mathbf{R}(\mathbf{u}) = - (\mathbf{u} \cdot \nabla) \mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f} \)

- **Time discretization:**
  - Central difference is used for the time derivative term
  - Fully explicit second order Adams-Bashforth scheme for \( \mathbf{R}(\mathbf{u}) \)
  - Implicit first-order Euler scheme for pressure-gradient term and mass-conservation equation

- **Spatial discretization:** Fourth-order symmetry-preserving discretization.
Pressure-velocity coupling (2/2)

Time-discrete system to be solved:

\[
\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{3}{2} \mathbf{R}^n - \frac{1}{2} \mathbf{R}^{n-1} - \frac{1}{\rho} \nabla p^{n+1}
\]
\[
\nabla \cdot \mathbf{u}^{n+1} = 0
\]

Predictor velocity is defined as \( \mathbf{u}^p = \mathbf{u}^n + \Delta t \left( \frac{3}{2} \mathbf{R}^n - \frac{1}{2} \mathbf{R}^{n-1} \right) \)

Then, the unknown velocity is \( \mathbf{u}^{n+1} = \mathbf{u}^p - \nabla \tilde{p} \)

To evaluate \( \tilde{p} = \frac{\Delta t}{\rho} p^{n+1} \), mass conservation equation is imposed:

\[
\nabla \cdot \mathbf{u}^{n+1} = \nabla \cdot \mathbf{u}^p - \nabla \cdot (\nabla \tilde{p}) = 0
\]

This leads to a Poisson equation \( \nabla^2 \tilde{p} = \nabla \cdot \mathbf{u}^p \) that must be solved to evaluate \( \tilde{p} \) and then \( \mathbf{u}^{n+1} \)

This approach is similar in all the segregated formulations for incompressible flows
Symmetry-preserving discretization (1/2)

The spatially discrete incompressible Navier-Stokes equations can be expressed as

\[
H \frac{d u_h}{dt} + C(u_h)u_h = Du_h - HGp_h
\]

\[
M u_h = 0
\]

\[\implies\] It can be shown that the convective matrix \( C(u_h) \) has to be skew-symmetric,

\[
C(u_h) + C^T(u_h) = 0
\]

and the discrete gradient operator \( G \) has to be exactly equal to...

\[
G = -H^{-1} M^T
\]
Symmetry-preserving discretization (2/2)

...in order to **conserve exactly** the following **inviscid invariants**

- **Kinetic energy**
  \[ \int_{\Omega} u \cdot ud\Omega \]

- **Enstrophy (in 2D)**
  \[ \int_{\Omega} (\nabla \times u) \cdot (\nabla \times u) d\Omega \]

- **Helicity (in 3D)**
  \[ \int_{\Omega} (\nabla \times u) \cdot ud\Omega \]

in a **discrete sense**.
Loosely coupled parallel computers

Low cost PC clusters are *loosely coupled* parallel computers:

- Good floating point power per-processor (excellent ratio **CPU power / cost**)
- Comparatively slow network (low bandwidth - **high latency**)

Parallel algorithms must be tolerant to slow networks to run efficiently on a PC cluster

For PCFD, unlikely other applications, latency is often the most critical problem
Direct Schur-Fourier Decomposition algorithm - Overview

- DSFD is a combination of a **FFT-based** method and **Direct Schur** method:
  - Fourier diagonalization is applied to reduce the 3D problem in a set of 2D problems.
  - The mesh must be uniform in $x$-direction.
  - Domain is decomposed only in directions $y$, $z$ to avoid doing parallel FFT, that is very inefficient (in our context) on loosely coupled computers
  - Each 2D problem is solved with a **Direct Schur** decomposition method.

- After pre-processing stage, DSFD allows the direct solution of Poisson equation with **only one message**.
**Problem definition: Differentially Heated Cavity**

Boundary conditions:
- **Isothermal vertical walls**
- **Adiabatic horizontal walls**
- **Periodic** boundary conditions in the $x$-direction, orthogonal to the main flow

Dimensionless governing numbers:
- $Ra_z = \frac{\beta \Delta T L_z^3 g}{\alpha \nu}$
- $Pr = \frac{\nu}{\alpha}$
- Height aspect ratio $A_z = \frac{L_z}{L_y}$
- Depth aspect ratio $A_x = \frac{L_x}{L_y}$
Differentially Heated Cavity (DHC) Problem

Definition of our problem

\begin{align*}
A_z &= 4 \\
Pr &= 0.71 \\
Ra_z &= 6.4 \times 10^8, \ 2 \times 10^9, \ 10^{10}
\end{align*}

- For this configuration, there is a Hopf bifurcation for \( Ra_z = 1.03 \times 10^8 \) and chaotic behaviour is first observed at \( 2.3 \times 10^8 \).

- Xin and Le Quére carried out simulations for the same \( Pr, A_z \) and \( Ra_z \) numbers, assuming a two-dimensional behaviour.

- In this DNS work, our main goal has been to study the effect of the three-dimensional fluctuations over the flow statistics for a wide range of \( Ra \)-numbers.

- Periodic boundary conditions allow to study the 3D effects due to instability of the main flow and not to the boundary conditions.
**Differentially Heated Cavity (DHC) Problem**

**Physical and numerical simulation parameters**

<table>
<thead>
<tr>
<th>Case</th>
<th>$Ra$</th>
<th>$Nx$</th>
<th>$Ny$</th>
<th>$Nz$</th>
<th>$Lx$</th>
<th>$\gamma_y$</th>
<th>$\gamma_z$</th>
<th>$dt$</th>
<th>Total time</th>
<th>Average time</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$6.4 \times 10^8$</td>
<td>128</td>
<td>156</td>
<td>312</td>
<td>2.0</td>
<td>1.5</td>
<td>1.5</td>
<td>$1.26 \times 10^{-3}$</td>
<td>1000</td>
<td>800</td>
<td>$2^{nd}$</td>
</tr>
<tr>
<td>B</td>
<td>$2 \times 10^9$</td>
<td>64</td>
<td>144</td>
<td>318</td>
<td>1.0</td>
<td>1.75</td>
<td>0.0</td>
<td>$1.27 \times 10^{-3}$</td>
<td>800</td>
<td>550</td>
<td>$4^{th}$</td>
</tr>
<tr>
<td>C</td>
<td>$10^{10}$</td>
<td>64</td>
<td>138</td>
<td>326</td>
<td>0.5</td>
<td>2.0</td>
<td>0.0</td>
<td>$1.49 \times 10^{-3}$</td>
<td>800</td>
<td>550</td>
<td>$4^{th}$</td>
</tr>
<tr>
<td>AA2D</td>
<td>$6.4 \times 10^8$</td>
<td>—</td>
<td>218</td>
<td>438</td>
<td>—</td>
<td>1.5</td>
<td>1.0</td>
<td>$6.96 \times 10^{-4}$</td>
<td>450</td>
<td>300</td>
<td>$2^{nd}$</td>
</tr>
<tr>
<td>BB2D</td>
<td>$2 \times 10^9$</td>
<td>—</td>
<td>202</td>
<td>448</td>
<td>—</td>
<td>1.75</td>
<td>0.0</td>
<td>$6.51 \times 10^{-4}$</td>
<td>325</td>
<td>200</td>
<td>$4^{th}$</td>
</tr>
<tr>
<td>CC2D</td>
<td>$10^{10}$</td>
<td>—</td>
<td>194</td>
<td>459</td>
<td>—</td>
<td>2.0</td>
<td>0.0</td>
<td>$7.56 \times 10^{-4}$</td>
<td>265</td>
<td>175</td>
<td>$4^{th}$</td>
</tr>
</tbody>
</table>
Differentially Heated Cavity (DHC) Problem

Main features

Complexity of the flow:

- Vertical boundary layers
- Stratified cavity core
- Internal waves
- Recirculation areas
Flow dynamics - Instantaneous isotherms

2D

\[ Ra = 6.8 \times 10^8 \]

\[ Ra = 2 \times 10^9 \]

\[ Ra = 10^{10} \]
Flow dynamics - Instantaneous isotherms
2D vs 3D

$Ra = 6.8 \times 10^8$

$Ra = 2 \times 10^9$

$Ra = 10^{10}$
Time-averaged flow - Streamlines

2D

\[ Ra = 6.8 \times 10^8 \quad Ra = 2 \times 10^9 \quad Ra = 10^{10} \]
Time-averaged flow - Streamlines
2D vs 3D

\[ Ra = 6.8 \times 10^8 \]

\[ Ra = 2 \times 10^9 \]

\[ Ra = 10^{10} \]
Averaged flow - Temperature profiles

2D

3D

Averaged temperature profile at the vertical mid-width plane.

- Experimental studies yield a dimensionless stratification of about 0.5 while numerical simulations predict values about 1.
- 3D assumption does not seem to improve the discrepancies in thermal stratification.
Averaged flow - Local Nusselt number

$$Ra = 2 \times 10^9$$

$$Ra = 10^{10}$$
Second-order statistics - Turbulent kinetic energy

2D

\[ Ra = 6.8 \times 10^8 \quad Ra = 2 \times 10^9 \quad Ra = 10^{10} \]
Second-order statistics - Turbulent kinetic energy

2D vs 3D

\[ Ra = 6.8 \times 10^8 \]  \[ Ra = 2 \times 10^9 \]  \[ Ra = 10^{10} \]
Second-order statistics - Turbulent kinetic energy dissipation

2D

\[ Ra = 6.8 \times 10^8 \]

\[ Ra = 2 \times 10^9 \]

\[ Ra = 10^{10} \]
Second-order statistics - Turbulent kinetic energy dissipation
2D vs 3D

$Ra = 6.8 \times 10^8$

$Ra = 2 \times 10^9$

$Ra = 10^{10}$
Conclusions

- 2D and 3D direct numerical simulations of natural convection in a cavity have been carried out using a fourth-order symmetry-preserving discretization.
- The 2D results obtained are in good agreement with the previous results.
- Respect to the comparison between 2D and 3D results:
  - The basic features and the first-order statistics such as the local and overall \( \text{Nu} \) are correctly predicted by the 2D simulations.
  - However, the second-order statistics are substantially different:
    - At \( Ra = 6.4 \times 10^8 \) the main differences occur at the vertical boundary layers, where the 2D simulations incorrectly predicts very low values for all turbulent statistics.
    - For the two highest \( Ra \)-numbers, the differences become more marked: for 2D simulation the area of large fluctuations increase whereas for 3D simulations it decreases shrinking to the downstream corners.
- Thermal stratification is still an open question.
Future Research

- The present results are being extended to $Ra = 10^{11}$ using a mesh of $\approx 111M$ nodes.

- DNS results are being used to test the performance of symmetry-preserving regularization models as a new turbulence simulation shortcut.
Questions?
Fourier Diagonalization (1/8)
Second-order discretization

Heptadiagonal system to be solved: \( A^{3d} x^{3d} = b^{3d} \)

\[
a_{i,j,k}^p x_{i,j,k}^p + \sum_{nb} a_{i,j,k}^{nb} x_{i,j,k}^{nb} = b_{i,j,k}
\]

To express it with block matrices, vectors \( x^{3d} \) and \( b^{3d} \) are divided into \( N_y N_z \) subvectors with \( N_x \) components each:

\[
x^{3d} = \begin{bmatrix} x_{1,1}, x_{2,1}, \cdots, x_{j,k}, \cdots, x_{Ny,Nz} \end{bmatrix}^t
\]

where:

\[
x_{j,k} = \begin{bmatrix} x_{1,j,k}, x_{2,j,k}, \cdots, x_{Nx,j,k} \end{bmatrix}^t
\]
Fourier Diagonalization (2/8)
Second-order discretization

\[
\begin{bmatrix}
A^{p}_{1,1} & A^{n}_{1,1} & \cdots & A^{t}_{1,1} \\
A^{s}_{2,1} & A^{p}_{2,2} & A^{n}_{2,1} & \cdots & A^{t}_{2,1} \\
& \ddots & \ddots & \ddots \\
A^{b}_{j,k} & \cdots & A^{s}_{j,k} & A^{p}_{j,k} & A^{n}_{j,k} & \cdots & A^{t}_{j,k} \\
& & A^{b}_{Ny,Nz} & \cdots & A^{s}_{Ny,Nz} & A^{p}_{Ny,Nz} & A^{n}_{Ny,Nz} & A^{t}_{Ny,Nz}
\end{bmatrix}
\begin{bmatrix}
x_{1,1} \\
x_{2,1} \\
\vdots \\
x_{j,k} \\
x_{Ny,Nz}
\end{bmatrix}
= 
\begin{bmatrix}
b_{1,1} \\
b_{2,1} \\
\vdots \\
b_{j,k} \\
b_{Ny,Nz}
\end{bmatrix}
\]

\[A^{b}_{j,k} x_{j,k-1} + A^{s}_{j,k} x_{j-1,k} + A^{p}_{j,k} x_{j,k} + A^{n}_{j,k} x_{j+1,k} + A^{t}_{j,k} x_{j,k+1} = b_{j,k}\]

- \(A^{n}_{j,k}, A^{s}_{j,k}, A^{t}_{j,k}\) and \(A^{b}_{j,k}\) are \(N_x \times N_x\) diagonal matrices
- \(A^{p}_{j,k}\) are \(N_x \times N_x\) circulant tridiagonal matrices. They contain the coefficients that link the unknowns of plane \(i\) with the unknowns of the neighbouring planes \(i - 1\) and \(i + 1\). If they could be diagonalized, each plane could be solved separately.

\[
A^{p}_{j,k} = \begin{bmatrix}
\beta & \alpha & & & \\
\alpha & \beta & \alpha & & \\
& \ddots & & \ddots & \\
\alpha & & \alpha & \beta & \\
\end{bmatrix}
\]

where \(\alpha = a^{w}_{j,k} = a^{e}_{j,k}\) and \(\beta = a^{p}_{j,k}\)
Fourier Diagonalization (3/8)
Second-order discretization

All the circulant matrices of order $N_x$ have the same base of eigenvectors

Let $Q$ be the matrix whose columns are the eigenvectors of all the $A_{p,j,k}^p$

The product $x = Q\bar{x}$ inverse Fourier transform:

$$x_i = \frac{1}{2} \bar{x}_1 + \sum_{\nu=1}^{N_x-1} \left( \bar{x}_{2\nu} \cos \left( \nu i \frac{2\pi}{N_x} \right) + \bar{x}_{2\nu+1} \sin \left( \nu i \frac{2\pi}{N_x} \right) \right) + \frac{1}{2} \bar{x}_{N_x} (-1)^i \quad i = 1 \cdots N_x$$

and the product $\bar{x} = Q^{-1}x$ is a direct Fourier transform:

$$\bar{x}_1 = \frac{2}{N_x} \sum_{i=1}^{N_x} x_i$$

$$\bar{x}_{2\nu} = \frac{2}{N_x} \sum_{i=1}^{N_x} x_i \cos \left( \nu i \frac{2\pi}{N_x} \right) \quad \nu = 1 \cdots \frac{N_x}{2} - 1$$

$$\bar{x}_{2\nu+1} = \frac{2}{N_x} \sum_{i=1}^{N_x} x_i \sin \left( \nu i \frac{2\pi}{N_x} \right) \quad \nu = 1 \cdots \frac{N_x}{2} - 1$$

$$\bar{x}_{N_x} = \sum_{i=1}^{N_x} x_i (-1)^i$$
Fourier Diagonalization (4/8)
Second-order discretization

All the matrices \( A_{j,k}^p \) have tridiagonal form in the same base:

\[
Q^{-1} A_{j,k}^p Q = \lambda_{j,k}
\]

where \( \lambda_{j,k} \) is a **diagonal matrix** whose elements are:

\[
\lambda_1 = \beta + 2\alpha \\
\lambda_{2\nu} = \lambda_{2\nu+1} = -4\alpha \sin^2 \left( \frac{\nu \pi}{N_x} \right) + \beta + 2\alpha \quad \nu = 1 \cdots \frac{N_x}{2} - 1 \\
\lambda_{N_x} = \beta - 2\alpha
\]

Thus, expressing \( x_{j,k} \) as \( Q \overline{x}_{j,k} \) and premultiplying by \( Q^{-1} \), the block equation becomes:

\[
A_{j,k}^b \overline{x}_{j,k-1} + A_{j,k}^s \overline{x}_{j-1,k} + \lambda_{j,k} \overline{x}_{j,k} + A_{j,k}^n \overline{x}_{j+1,k} + A_{j,k}^t \overline{x}_{j,k+1} = \overline{b}_{j,k}
\]

As the non-diagonal entries of matrices \( A_{j,k}^p \) have been eliminated, **unknown \( \overline{x}_{i,j,k} \) is only coupled with unknowns in the same plane** \( i \).
Fourier Diagonalization (5/8)

Second-order discretization

Selecting of the $i$ component of each of the $N_y N_z$ block equations, we obtain a penta-diagonal scalar equation system in $\overline{x}$

$$a^b_{j,k} \overline{x}_{i,j,k-1} + a^s_{j,k} \overline{x}_{i,j-1,k} + a^p_{i,j,k} \overline{x}_{i,j,k} + a^n_{j,k} \overline{x}_{i,j+1,k} + a^t_{j,k} \overline{x}_{i,j,k+1} = \overline{b}_{i,j,k}$$

The operations to be performed to solve the heptadiagonal equation system are:

1. **Direct FFT.** Calculate the $N_y N_z$ transformed right-hand-side sub-vectors, $\overline{b}_{j,k} = Q^{-1} b$
2. **Solve** the $N_x$ decoupled pentadiagonal equation systems $\overline{A}_i \overline{x}_i = \overline{b}_i$.
3. **Inverse FFT.** Carry out the antitransformation of the $N_y N_z$ solution sub-vectors $\overline{x}_{j,k} = Q^{-1} x_{j,k}$

FFT are very cheap if done sequentially, but in our conditions they can not be carried out efficiently with loosely coupled parallel computers (to the knowledge of the authors)

This is why the domain is only decomposed in directions $y, z$

Solution of the pentadiagonal systems is done with a Direct Schur Decomposition using just one message for all the pentadiagonal equation systems
Fourier Diagonalization (6/8)
Fourth-order discretization

The system $A^{3d}x^{3d} = b^{3d}$ than arises from the fourth-order discretization has 19 diagonals. $A^p_{j,k}$ are heptadiagonal $N_x \times N_x$ circulant matrices,

$$A^p_{j,k} = \begin{bmatrix}
\beta & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_3 & \alpha_2 & \alpha_1 \\
\alpha_1 & \beta & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_3 & \alpha_2 \\
\alpha_2 & \alpha_1 & \beta & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_3 \\
\alpha_3 & \alpha_3 & \alpha_2 & \alpha_1 & \beta & \alpha_1 & \alpha_2 \\
\alpha_2 & \alpha_3 & \alpha_3 & \alpha_2 & \alpha_1 & \beta & \alpha_1 \\
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_3 & \alpha_2 & \alpha_1 & \beta \\
\end{bmatrix}$$

where

$$\alpha_1 = a^{w_1}_{j,k} = a^{e_1}_{j,k}$$
$$\alpha_2 = a^{w_2}_{j,k} = a^{e_2}_{j,k}$$
$$\alpha_3 = a^{w_3}_{j,k} = a^{e_3}_{j,k}$$
$$\beta = a^p_{j,k}$$
Fourier Diagonalization (7/8)

Fourth-order discretization

However, as $A_{j,k}^p$ are also $N_x \times N_x$ circulant matrices, they have the same base of eigenvectors

$$Q^{-1} A_{j,k}^p Q = \lambda_{j,k}$$

Now, the eigenvalues will be given by

$$\lambda_j = \beta + 2 \sum_{i=1}^{3} \alpha_i \cos \left( \frac{2\pi i (j - 1)}{N_x} \right) \quad j = 1, \ldots, N_x$$

And, after diagonalization, we will obtain a 13-diagonal scalar equation system in $\bar{x}$.

$$\sum_{d=1}^{3} \left( a_{j,k}^{b,d} \bar{x}_{i,j,k-d} + a_{j,k}^{s,d} \bar{x}_{i,j-d,k} + a_{j,k}^{n,d} \bar{x}_{i,j+d,k} + a_{j,k}^{t,d} \bar{x}_{i,j,k+d} \right) + a_{i,j,k}^p \bar{x}_{i,j,k} = b_{i,j,k}$$
In conclusion,

- **Only** the eigenvalues calculation needs to be **changed** respect to Fourier diagonalization of the system that arises from a second-order discretization.

- The **computational cost** of the Fourier diagonalization does **NOT**\(^1\) depend on the **order** of accuracy of the Poisson equation.

\(^1\)If we do not consider the slight differences in the eigenvalues calculation, of course.
Direct Schur Decomposition - Main ideas (1/7)

Based on **non-overlapping** subdomains with **implicit treatment** of the interface.

Partition and reordering of the unknowns for the second-order(left) and fourth-order(right) discrete Poisson equations.

Only one all-to-all communication episode is needed.
Direct Schur Decomposition - Overview (2/7)

No property of $A$ (i.e., symmetry or positive-definiteness) or the underlying mesh is required.

Each processor has to solve twice its own subdomain and cooperate to solve an interface equation to obtain the exact solution of the problem.

After a pre-processing step only one all-to-all communication episode is needed.

It can be applied to 2D and 3D problems. Its main limitation for 3D problems is the memory. This is why the Fourier decomposition is carried out first.

Compared with matrix inversion:

- $DSD(A)$ needs far less storage memory than $A^{-1}$
- Its faster to preprocess $DSD(A)$ than $A^{-1}$
- When $DSD(A)$ is available, it is faster to solve $Ax = b$ than to evaluate the matrix-vector product $A^{-1}b$
Direct Schur Decomposition (3/7)

Each of the pentadiagonal systems is denoted by: $Ax = b$

Decomposition (a subvector is assigned to each processor): $x = [x_0, x_1, \cdots, x_{P-1}, x_s]^t$

After reordering, and using block matrices, the system is:

$$
\begin{bmatrix}
A_{0,0} & 0 & \cdots & 0 & A_{0,s} \\
0 & A_{1,1} & \cdots & 0 & A_{1,s} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & A_{P-1,P-1} & A_{P-1,s} \\
A_{s,0} & A_{s,1} & \cdots & A_{s,P-1} & A_{s,s}
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
\vdots \\
x_{P-1} \\
x_s
\end{bmatrix}
=
\begin{bmatrix}
b_0 \\
b_1 \\
\vdots \\
b_{P-1} \\
b_s
\end{bmatrix}
$$

Assuming non-singular local matrices, Block Gaussian elimination allows to express it as:

$$
\begin{bmatrix}
A_{0,0} & 0 & \cdots & 0 & A_{0,s} \\
0 & A_{1,1} & \cdots & 0 & A_{1,s} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & A_{P-1,P-1} & A_{P-1,s} \\
0 & 0 & \cdots & 0 & \tilde{A}_{s,s}
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
\vdots \\
x_{P-1} \\
x_s
\end{bmatrix}
=
\begin{bmatrix}
b_0 \\
b_1 \\
\vdots \\
b_{P-1} \\
\tilde{b}_s
\end{bmatrix}
$$
Direct Schur Decomposition (4/7)

The interface equation \( \tilde{A}_{s,s} x_s = \tilde{b}_s \) allows to solve the interface nodes before the rest!

\[
\tilde{A}_{s,s} = A_{s,s} - \sum_{p=0}^{P-1} A_{s,p} A^{-1}_{p,p} A_{p,s} \\
\tilde{b}_s = b_s - \sum_{p=0}^{P-1} A_{s,p} A^{-1}_{p,p} b_p
\]

After solving the interface equation, each local subproblem can be easily solved by a processor without additional communications:

\[
A_{p,p} x_p = b_p - A_{p,s} x_s
\]
Direct Schur Decomposition (5/7)

Evaluation of interface equation \( \tilde{A}_{s,s} x_s = \tilde{b}_s \)

\[
\tilde{A}_{s,s} = A_{s,s} - \sum_{p=0}^{P-1} A_{s,p} A_{p,p}^{-1} A_{p,s} \\
\tilde{b}_s = b_s - \sum_{p=0}^{P-1} A_{s,p} A_{p,p}^{-1} b_p
\]

To avoid evaluation of \( A_{p,p}^{-1} \):

1. Evaluation of interface matrix \( \tilde{A}_{s,s} = A_{s,s} - \sum_{p=0}^{P-1} \tilde{A}_{s,s}^p \)

   Contribution from processor \( p \): \( \tilde{A}_{s,s}^p = A_{s,p} A_{p,p}^{-1} A_{p,s} \)

   Column \( c \) of \( \tilde{A}_{s,s}^p \) is evaluated as:
   
   (a) Solve \( t \) from \( A_{p,p} t = [A_{p,s}]_c \)
   
   (b) \( [\tilde{A}_{s,s}^p]_c \leftarrow A_{s,p} t \)

2. Evaluation of right-hand-side vector \( \tilde{b}_s = b_s - \sum_{p=0}^{P-1} \tilde{b}_s^p \)

   Contribution from processor \( p \): \( \tilde{b}_s^p = A_{s,p} A_{p,p}^{-1} b_p \)

   (a) Solve \( t \) from \( A_{p,p} t = b_p \)

   (b) \( \tilde{b}_s^p \leftarrow A_{s,p} t \)
The algorithm has two parts:

1. **Preprocessing stage**
   - Used only once
   - As $\tilde{A}_{s,s}$ only depends on $A$ (and not $b$), it is evaluated and inverted for each plane

2. **Solution stage**
   - Called for each $b$ to be solved ($10^5 \cdots 7$ times!)
   - (a) $\tilde{b}_s$ is evaluated. Each processor has to solve a local equation system to do so. **Here, one all-to-all message is needed.** The information for all the pentadiagonal equations is packed in the same all-to-all message
   - (b) Each processor performs the part of the matrix vector product $\left[ \tilde{A}_{s,s} \right]^{-1} b_s$ needed to evaluate the nodes of $x_s$ adjacent to its own subdomain
   - (c) Each processor solves a local equation to determine its subvector $x_p$
Direct Schur Decomposition (7/7)

The algorithm is not conceptually complex but its efficiency depends on certain details:

- In order to evaluate $\left[\tilde{A}_{s,s}\right]^{-1}b_s$, a **parallel block-LU algorithm** is used to improve the efficiency on loosely coupled systems.
- After the block-LU decomposition, the inverse is calculated **row by row**. The block LU of $\tilde{A}_{s,s}^t$ is evaluated and then inverse and transpose operators are permuted.
- **Distributed storage** of $\left[\tilde{A}_{s,s}\right]^{-1}$. Each processor only stores the rows of the inverse needed to perform its part of $\left[\tilde{A}_{s,s}\right]^{-1}b_s$.
- The local equation systems are solved using a **band-LU algorithm**.
- Different **matrix data structures** are needed:
  - Matrices $A_{s,p}$, $A_{p,s}$, $A_{s,s}$ are treated as **sparse**.
  - Matrices $A_{p,p}$ are **banded**.
  - Matrix $\tilde{A}_{s,s}$ is treated as a **block matrix** and **distributed by rows** at the different processors.
  - Matrix $\left[\tilde{A}_{s,s}\right]^{-1}$ is treated as **dense** and **stored by rows** at the different processors.
Benchmark (1/9) - Speedup of DSD on a PC cluster

Totally direct, second order equation

900 MHz K7 processors; Switched 100 Mbits/s network

The cost of DSFD is roughly $N \times x$ times the cost of one pentadiagonal equation. Irregular behaviour for a fixed $N$ is due to different bandwidths of local problems depending on $P$.

Super-linear $S$ is due to the non-linear cost of local band-LU solver and low cost of interface solution and communications.
Benchmark (2/9) - DSD solution time on a PC cluster

Totally direct, second order equation

900 MHz K7 processors; Switched 100 Mbits/s network

For $N = 125 \times 10^3$, $P = 24$ Direct Shur Complement is $\approx 30$ times faster than sequential ACM multigrid with $\epsilon^* = 10^{-3}$
Benchmark (3/9) - DSFD solution time on a PC cluster
Totally direct, second order equation
900 MHz K7 processors; Switched 100 Mbits/s network

For $N = 3.1 \times 10^6$, each equation can be solved almost to machine accuracy in about three seconds
Benchmark (4/9) - Decomposition of total time with 36 processors
Direct Solution of Second-order Poisson Equation

\[ u^{n+1} = u^p - \nabla \tilde{p} \]
Benchmark (5/9) - Decomposition of total time with 24 processors
Direct Solution of Second- and Fourth-order Poisson Equations
2600 MHz K7 processors; Switched 100 Mbits/s network
Benchmark (6/9) - DSFD solution time on a PC cluster
Direct Solution of Second- and Fourth-order Poisson Equations
2600 MHz K7 processors; Switched 100 Mbits/s network
Benchmark (7/9) - Speedup of DSFD on a PC cluster

Direct Solution of Second- and Fourth-order Poisson Equations

2600 MHz K7 processors; Switched 100 Mbits/s network
The total number of array positions that each processor needs is approximately

\[ N_t^{2d}(M, I, N_{sp}, P) \approx 3M \left( \frac{N_{sp}}{P} \right)^{3/2} + 8I^2 N_{sp} \frac{P^{1/2} - 1}{P^{1/2}} \]

where \( P \) is the number of processors, \( I \) is the interface size, \( M \) the scheme stencil size and \( N_{sp} \) the number of unknowns per plane.

As \( M \) and \( I \) are increasing functions of the order of the scheme\(^2\), RAM memory requirements **may become a serious obstacle** to use DSD for **high-order schemes**.

\(^2\)In our case:

\[ M = o - 1 \quad I = o - 1 \]
In particular,

- **Second-order:**

\[ N_t^{2d} \approx 3 \left( \frac{N_{sp}}{P} \right)^{3/2} + 8N_{sp} \frac{P^{1/2} - 1}{P^{1/2}} \]

\( \text{local band–LU problems + interface equation} \)

- **Fourth-order:**

\[ N_t^{2d} \approx 9 \left( \frac{N_{sp}}{P} \right)^{3/2} + 72N_{sp} \frac{P^{1/2} - 1}{P^{1/2}} \]

\( \text{local band–LU problems + interface equation} \)

Note that the **main difficulty** arises from the **interface equation** whose RAM memory requirement does NOT decrease with the number of processors.
Stationary Iterative Approach for Schur Problems (1/3)

The aim is to **reduce the RAM memory requirements** maintaining the results accurate enough with **reasonable computing times**.

To do so:

- The Fourier diagonalization is carried out as in the fully direct methods
- To solve each 2D equation, the Direct Schur part is replaced by a stationary iterative method

Let each 2D equation be:

\[ Ax = b \]

Matrix \( A \) is separated into a \( A_d \) (to be solved with DSD) and \( A_i \)

\[ A = A_d + A_i \]

The **iteration** takes the form

\[ A_d x^{k+1} = b - A_i x^k \]

Now the question is how to choose \( A_d \) and \( A_i \)
Option 1. Save memory in the local band-LU problems and in the interface equation $A_{d}^{1,1}$: Reduce the stencil to one node in all the domain

Option 2. Save memory only in the interface equation $A_{d}^{3,1}$: it reduces the stencil size to one node but only for the nodes near the interface
Benchmark (8/9) - DSFD Iterative approach (4th order eq.)

CPU time versus RAM memory

2600 MHz K7 processors; Switched 100 Mbits/s network; \( N \approx 3.1 \times 10^6; \epsilon = 10^{-6} \)

With the same RAM memory requirement **slightly smaller** computing times are obtained when both iterative approaches \( A_d^{3,1} \) and \( A_d^{1,1} \) are properly combined.
Benchmark (9/9) - DSFD Iterative approach (4th order eq.)

number of iterations needed for each plane

24 2600 MHz K7 processors; Switched 100 Mbits/s network; $N = 64 \times 156 \times 312$; $\epsilon = 10^{-6}$

The number of iterations needed for high-frequency planes is smaller because the systems are better conditioned
Governing equations

Incompressible Navier-Stokes coupled with energy transport equation:

\[
\nabla \cdot u = 0
\]

\[
\partial_t u + C(u, u) = Pr\mathcal{D}(u) - \nabla p + f
\]

\[
\partial_t T + C(u, T) = \mathcal{D}(T)
\]

where \( f = (0, 0, Ra Pr T) \) (Boussinesq approximation) and the nonlinear convective term is given by

\[
C(u, v) = (u \cdot \nabla)v
\]

and the linear dissipative term is given by

\[
\mathcal{D}(u) = \frac{1}{Ra^{0.5}} \nabla^2 u
\]
Regularization modelling

A **dynamically less complex mathematical formulation** is sought. We consider smooth approximations (regularizations) of the nonlinearity,

\[ \partial_t u_\epsilon + \tilde{C}(u_\epsilon, u_\epsilon) = PrD(u_\epsilon) - \nabla p_\epsilon + f \]
\[ \partial_t T_\epsilon + \tilde{C}(u_\epsilon, T_\epsilon) = D(T_\epsilon) \]

such approximations may fall in the **Large-Eddy Simulation** (LES) concept,

\[ \partial_t \bar{u}_\epsilon + C(\bar{u}_\epsilon, \bar{u}_\epsilon) = PrD(\bar{u}_\epsilon) - \nabla \bar{p}_\epsilon + f + M_1(\bar{u}_\epsilon, \bar{u}_\epsilon) \]
\[ \partial_t \bar{T}_\epsilon + C(\bar{u}_\epsilon, \bar{T}_\epsilon) = D(\bar{T}_\epsilon) + M_2(\bar{u}_\epsilon, \bar{T}_\epsilon) \]

if the model terms were given by

\[ M_1(\bar{u}_\epsilon, \bar{u}_\epsilon) = C(\bar{u}_\epsilon, \bar{u}_\epsilon) - \overline{\tilde{C}(\bar{u}_\epsilon, \bar{u}_\epsilon)} \]
\[ M_2(\bar{u}_\epsilon, \bar{T}_\epsilon) = C(\bar{u}_\epsilon, \bar{T}_\epsilon) - \overline{\tilde{C}(\bar{u}_\epsilon, \bar{T}_\epsilon)} \]
Existing regularization modellings
Leray and Navier-Stokes-$\alpha$ models

The regularization methods basically alters the convective term to restrain the production of small scales of motion.

- Leray model:

$$\partial_t u_\epsilon + C(\bar{u}_\epsilon, u_\epsilon) = D(u_\epsilon) - \nabla p_\epsilon$$

- Navier-Stokes-$\alpha$ model:

$$\partial_t u_\epsilon + C_r(u_\epsilon, \bar{u}_\epsilon) = D(u_\epsilon) - \nabla \pi_\epsilon$$

where the $\pi = p + u^2/2$ and the convetive operator in rotational form is defined as

$$C_r(u, v) = (\nabla \times u) \times v$$

However, in doing so some of the inviscid invariants (kinetic energy, enstrophy in 2D and helicity in 3D) are not conserved.
Symmetry-preserving regularization models (1/2)

In order to conserve exactly the following inviscid invariants

- **Kinetic energy**
  \[ \int_\Omega u \cdot ud\Omega \]

- **Enstrophy (in 2D)**
  \[ \int_\Omega (\nabla \times u) \cdot (\nabla \times u) d\Omega \]

- **Helicity (in 3D)**
  \[ \int_\Omega (\nabla \times u) \cdot ud\Omega \]

the approximate convective operator has to be skew-symmetric:

\[ (\tilde{C}(u, v), w) = - (\tilde{C}(u, w), v) \]
Symmetry-preserving regularization models (2/2)

This criterion yields the following class of approximations...

\[ \partial_t u_\epsilon + C_n(u_\epsilon, u_\epsilon) = D(u_\epsilon) - \nabla p_\epsilon \]

in which the convective term is smoothened according to:

\[ C_2(u, v) = \overline{C(\bar{u}, \bar{v})} \]

\[ C_4(u, v) = C(\bar{u}, \bar{v}) + \overline{C(\bar{u}, v')} + \overline{C(u', \bar{v})} \]

\[ C_6(u, v) = C(\bar{u}, \bar{v}) + C(\bar{u}, v') + C(u', \bar{v}) + \overline{C(u', v')} \]

where \( u' = u - \bar{u} \) and \( C_n(u, v) = C(u, v) + O(\epsilon^n) \) for any symmetric filter.
Discretization of the convective operator: a symmetry-preserving discretization

The spatially discrete incompressible Navier-Stokes equations can be expressed as

\[
H \frac{d u_h}{dt} + C(u_h) u_h + D u_h - M^T p_h = 0 \\
M u_h = 0
\]

It can be shown that the convective matrix \( C(u_h) \) has to be **skew-symmetric**, 

\[
C(u_h) + C^T(u_h) = 0
\]

to **preserve** the continuous **invariants** (kinetic energy, enstrophy in 2D and helicity in 3D) in a **discrete sense**.
Choice of the filter

Let us consider a generic linear filter

\[ \tilde{u}_\epsilon = F u_\epsilon \]

Then, three basic properties are required for the filter:

\[ \tilde{u}_\epsilon = u_\epsilon + O(\epsilon^2) \]
\[ (HF) = (HF)^T \]
\[ F1 = 1 \]

Our filter is based on the elliptic differential operator

\[ (1 - \alpha_1^2 \partial_{xx}^2 - \alpha_2^2 \partial_{yy}^2 - \alpha_3^2 \partial_{zz}^2)\tilde{u}_\epsilon = u_\epsilon \]

where filter length is defined by

\[ \epsilon_i = \alpha_i \sqrt{24} \]
Results for differentially heated cavity at $Ra = 10^{10}$

- Regularization model $C_4$ is tested.
- Two very coarse meshes are considered

<table>
<thead>
<tr>
<th></th>
<th>DNS</th>
<th>RM1</th>
<th>RM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Nx$</td>
<td>64</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>$Ny$</td>
<td>136</td>
<td>17</td>
<td>13</td>
</tr>
<tr>
<td>$Nz$</td>
<td>324</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>$\Delta x_{min}$</td>
<td>$7.81 \times 10^{-3}$</td>
<td>$6.25 \times 10^{-2}$</td>
<td>$6.25 \times 10^{-2}$</td>
</tr>
<tr>
<td>$\Delta y_{min}$</td>
<td>$1.11 \times 10^{-3}$</td>
<td>$8.88 \times 10^{-3}$</td>
<td>$1.16 \times 10^{-2}$</td>
</tr>
<tr>
<td>$\Delta z_{min}$</td>
<td>$1.23 \times 10^{-2}$</td>
<td>$9.96 \times 10^{-2}$</td>
<td>$1.33 \times 10^{-1}$</td>
</tr>
</tbody>
</table>

- Ratio $\epsilon/h$ (filter length to the average grid width) is kept constant in all three spatial directions.
Results for differentially heated cavity at $Ra = 10^{10}$

Mean fields

$8 \times 13 \times 30$

$8 \times 17 \times 40$

Averaged vertical velocity profile at the horizontal mid-height plane for different $\epsilon/h$ ratios.
Results for differentially heated cavity at $Ra = 10^{10}$

Mean fields

$8 \times 13 \times 30$  

$8 \times 17 \times 40$

Averaged temperature profile at the horizontal mid-height plane for different $\epsilon/h$ ratios.
Results for differentially heated cavity at $Ra = 10^{10}$

Convergence studies

The maximum of the averaged vertical velocity at the horizontal mid-height plane and the overall averaged Nusselt number as a function of the ratio of the filter length $\epsilon$ to the average grid width $h$. 
Results for differentially heated cavity at $Ra = 10^{10}$

**Turbulent statistics**

$8 \times 13 \times 30$  
$8 \times 17 \times 40$

Turbulent kinetic energy $k = \overline{u'_i u'_i}$ profile at the horizontal mid-height plane for different filter lengths $\epsilon$. 


How the $\tilde{C}_4$ symmetry-preserving regularization modelling behave for finer grids?

First preliminary results on a finer $16 \times 34 \times 80$ grid does not seem to improve the agreement with DNS results...

Averaged vertical velocity and temperature profiles at the horizontal mid-height plane for different filter lengths $\epsilon$.

$\rightarrow$ The problem seems to be related with the linear filter.
Conclusions and Future Research

The first results shown illustrate the potential of conservative smoothing as a new simulation shortcut.

The main advantages with respect exiting LES models can be summarized:

- **Robustness.** As the smoothed governing equations preserve the symmetry properties of the original Navier-Stokes equations the solution can not blow up (in the energy-norm, in 2D also: enstrophy-norm). It seems that even for very coarse meshes reasonably results can be obtained.

- **Universality.** No *ad hoc* phenomenological arguments that can not be formally derived for the Navier-Stokes equations are used.

However, once the robustness of the method has been shown for very coarse meshes, future research should focus on the construction of more appropriate linear filters.