DNS of Natural Convection in a Differentially Heated Cavity. Effect of the 3D Fluctuations.

F.X. Trias, M. Soria, C.D. Pérez-Segarra, A. Oliva

Centre Tecnològic de Transferència de Calor (CTTC)
Lab. de Termodinàmica i Energètica, Universitat Politècnica de Catalunya (UPC)
ETSEIT, c/ Colom 11, 08222 Terrassa, Spain
e-mail: manel@labtie.mmt.upc.es, web page: http://www.upc.es/lte
Presentation outline

- 1-Numerical methods
  - Governing equations
  - Pressure-velocity coupling. Poisson equation. Difficulties of incompressible flows
  - Loosely-coupled parallel computers
  - Alternatives to solve the Poisson equation
  - DSFD Poisson solver
  - Benchmarks

- 2-DNS of turbulent natural convection in a cavity
  - Natural convection flows in cavities. Problem definition
  - Verification of the DNS code
  - Verification of the DNS simulations
  - Instantaneous flows
  - First-order statistics of the flow
  - Kinetic energy balances and spectroconsistent discretization
  - Second-order statistics of the flow
  - Comparison of 2D and 3D profiles

- 3-Conclusions
Introduction - Governing equations

Direct numerical simulation of turbulent natural convection flows.

Navier-Stokes coupled with energy transport equation.

\[
\begin{align*}
\nabla \cdot \mathbf{u} &= 0 \\
\frac{\partial \mathbf{u}}{\partial t} &= -\mathbf{u} \cdot \nabla \mathbf{u} + \nu \nabla^2 \mathbf{u} - \frac{1}{\rho} \nabla p + \mathbf{f} \\
\frac{\partial T}{\partial t} &= -\mathbf{u} \cdot \nabla T + \alpha \nabla^2 T
\end{align*}
\]

Where \( f_z = g \beta T \)

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 (1/3)

Momentum equation is expressed as:

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

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

- **Spatial discretization**: Second order spectro-consistent discretization (fourth-order scheme is being implemented)
- **Time discretization**:
  - Central difference is used for the time derivative term
  - Fully explicit second order Adams-Bashforth scheme for \( \mathbf{R} \)
  - Implicit first-order Euler scheme for pressure-gradient term and mass-conservation equation
Pressure velocity coupling (2/3)

Time-discrete system to be solved:

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

\[
\nabla \cdot u^{n+1} = 0
\]

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

\( u^p \) is calculated with data available from previous time steps

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

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

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

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

This approach is similar in all the segregated formulations for incompressible flows
Pressure velocity coupling (3/3)

Why is Poisson equation so difficult? A physical argument

Sound velocity is \( c = \sqrt{\left( \frac{\partial p}{\partial \rho} \right)_s} \)

Incompressibility \( \rightarrow \frac{\partial \rho}{\partial p} = 0 \rightarrow c = \infty \rightarrow \) local changes affect instantaneously all the domain

This behaviour is inherited by the Poisson equation, without time derivatives

\( \nabla^2 \tilde{p} = \nabla \cdot u^p \)

It has to be solved implicitly even if the rest of the formulation is explicit

At least one large linear equation system, coupling distant nodes, has to be solved per time step/iteration:

\( Ax = b \)

However, under certain conditions matrix \( A \) remains constant during all the problem

Our problem is actually to solve: \( Ax_i = b_i \quad i = 1 \cdots M \) where \( M \approx 10^{5\cdots7} \)
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
Matrix-vector product on a cluster and a Cray T3E

Minimum number of nodes per processor for efficiency > 50% ?

- Cray T3E (300 MHz) versus 900 MHz K7 (100 Mbit/s fast ethernet)
- $\frac{N}{P}$: number of nodes assigned to each processor.
Alternatives to solve the Poisson equation

The main challenge for DNS on a loosely-coupled system is the efficient solution of the Poisson equation

- **MG algorithms**
  - Very efficient on sequential systems
  - Need very low latency parallel computers; on high latency systems they must be combined with direct parallel solvers
- **Krylov subspace algorithms**
  - Parallelize well
  - Depend on good preconditioners to be efficient
  - Preconditioners tend to degrade with $P$
- Fast-Poisson solvers and **FFT-based methods** are restricted to determined classes of problems
- **Schur Complement methods**
  - Iterative
  - Direct - **Use the fact that matrix $A$ is constant**
DSFD is a combination of a **FFT-based** method and a **Direct Schur** method:
- **Fourier** diagonalization is applied to reduce the heptadiagonal equation to a family of independent pentadiagonal equations (this imposes certain restrictions)
- The pentadiagonal equations are solved with a **Direct Schur** decomposition method
- It is an **algebraic** approach: only the discrete equation system is used
- **Direct** method, based on a preprocessing stage where only matrix $\mathbf{A}$ is used
- After pre-processing, DSFD allows the solution of heptadiagonal equations using parallel computers with just one message
- DSFD is an interesting option to consider if:
  - A loosely coupled parallel computer is to be used
  - The matrix $\mathbf{A}$ has to be used many times with different right-hand-sides
  - The problem to be solved is periodic in one direction

E.g., for LES/DNS with Beowulf clusters
The mesh must be uniform in $x$ direction (usually periodic).

Domain is decomposed only in directions $y$, $z$ to avoid doing parallel FFT, that is very inefficient (in our context) on loosely coupled computers.
Fourier Diagonalization (1/5)

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

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

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:

\[
x^{3d} = [x_{1,1}, x_{2,1}, \cdots x_{j,k}, \cdots, x_{N_y,N_z}]^t
\]

where:

\[
x_{j,k} = [x_{1,j,k}, x_{2,j,k}, \cdots, x_{N_x,j,k}]^t
\]
Fourier Diagonalization (2/5)

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

\[
A_{j,k}^b x_{j,k-1} + A_{j,k}^s x_{j-1,k} + A_{j,k}^P x_{j,k} + A_{j,k}^n x_{j+1,k} + A_{j,k}^t x_{j,k+1} = b_{j,k}
\]

- \(A_{j,k}^n, A_{j,k}^s, A_{j,k}^t\) and \(A_{j,k}^b\) are \(N_x \times N_x\) diagonal matrices
- \(A_{j,k}^P\) are \(N_x \times N_x\) **circulant** tridiagonal matrices:

\[
A_{j,k}^p = \begin{bmatrix}
\beta & \alpha & \alpha \\
\alpha & \beta & \alpha \\
\alpha & \cdots & \alpha & \beta \\
\end{bmatrix}
\]

where \(\alpha = a_{j,k}^w = a_{j,k}^e\) and \(\beta = a_{j,k}^p\)
Fourier Diagonalization (3/5)

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 $A_{j,k}^P$

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

$$x_i = \frac{1}{2}x_1 + \sum_{\nu=1}^{N_x-1} \left( \bar{x}_{2\nu} \cos \left( \nu \frac{2\pi}{N_x} \right) + \bar{x}_{2\nu+1} \sin \left( \nu \frac{2\pi}{N_x} \right) \right) + \frac{1}{2} \bar{x}_{N_x} (-1)^i \quad i = 1 \cdots N_x$$

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

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

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

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

$$\bar{x}_{N_x} = \sum_{i=1}^{N_x} x_i (-1)^i$$
All the matrices \( A^p_{j,k} \) have tridiagonal form in the same base:

\[
Q^{-1} A^p_{j,k} Q = \lambda_{j,k}
\]

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

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

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

\[
A^b_{j,k} \overline{x}_{j-1,k} + A^s_{j,k} \overline{x}_{j-1,k} + \lambda_{j,k} \overline{x}_{j,k} + A^n_{j,k} \overline{x}_{j+1,k} + A^t_{j,k} \overline{x}_{j,k+1} = b_{j,k}
\]

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

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

\[
a_{j,k}^b \bar{x}_{i,j,k-1} + a_{j,k}^s \bar{x}_{i,j-1,k} + a_{i,j,k}^p \bar{x}_{i,j,k} + a_{j,k}^n \bar{x}_{i,j+1,k} + a_{j,k}^t \bar{x}_{i,j,k+1} = \bar{b}_{i,j,k}
\]

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, $\bar{b}_{j,k} = Q^{-1} b$
2. **Solve** the $N_x$ decoupled pentadiagonal equation systems $A_i \bar{x}_i = \bar{b}_i$.
3. **Inverse FFT.** Carry out the antitransformation of the $N_y N_z$ solution sub-vectors $\bar{x}_{j,k} = Q^{-1} x_{j,k}$

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

![Diagram showing subdomains and interface nodes](image)

**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 than the matrix-vector product $A^{-1}b$
- Direct Schur decomposition is more appropriated to parallel computers
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:

$$
\begin{bmatrix}
A_{0,0} & 0 & \cdots & 0 & A_{0,s} \\
0 & A_{1,1} & \cdots & 0 & A_{1,s} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & A_{P-1,P-1} & A_{P-1,s} \\
0 & 0 & \cdots & 0 & A_{s,s}
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
\vdots \\
x_{P-1} \\
x_s
\end{bmatrix}
= 
\begin{bmatrix}
b_0 \\
b_1 \\
\vdots \\
b_{P-1} \\
b_s
\end{bmatrix}
$$
The interface equation \( \tilde{A}_{s,s}x_s = \tilde{b}_s \) allows to solve the interface nodes before the rest!

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

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

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 \( \tilde{A}_{s,s} x_s = \tilde{b}_s \)

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

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

1. Evaluation of interface matrix \( \tilde{A}_{s,s} = A_{s,s} - \sum_{p=0}^{P-1} \tilde{A}_p \)
   Contribution from processor \( p \): \( \tilde{A}_p = A_{s,p} A_{p,p}^{-1} A_{p,s} \)
   Column \( c \) of \( \tilde{A}_s \) is evaluated as:
   a) Solve \( t \) from \( A_{p,p} t = [A_{p,s}]_c \)
   b) \( \left[ \tilde{A}_s \right]_c \leftarrow A_{s,p} t \)

2. Evaluation of right-hand-side vector \( \tilde{b}_s = b_s - \sum_{p=0}^{P-1} \tilde{b}_p \)
   Contribution from processor \( p \): \( \tilde{b}_p = A_{s,p} A_{p,p}^{-1} b_p \)
   a) Solve \( t \) from \( A_{p,p} t = b_p \)
   b) \( \tilde{b}_s \leftarrow A_{s,p} t \)
Direct Schur Decomposition (6/7)

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 ($10^5\ldots7$ 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 determinate 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 (1/4) - Speedup of DSD on a PC cluster

900 MHz K7 processors; Switched 100 Mbits/s network

The cost of DSFD is roughly $N_x$ 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 (2/4) - DSD solution time on a PC cluster
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 (3/4) - DSFD solution time on a PC cluster

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 (4/4) - Decomposition of total time with 36 processors

- total
  - statistic analysis
- NS
  - update halos
  - momentum-energy
- mass
  - \( \nabla \cdot \mathbf{u}^p \)
  - DSFD
  - \( u^{n+1} = u^p - \nabla \tilde{p} \)
- DSFD
  - FFT
  - global summation
  - band-LU, matrix-vector
- total
  - CPU
  - network, load unbalance
Natural Convection Flows in Cavities

The majority of the natural convection problems in closed cavities can be classified in three groups:

- Cavities where the flow is due to **internal heat generation**
- Cavities **heated from below** (Rayleigh-Bénard configuration)
- **Differentially heated cavities** (DHC)
  - Laminar flows: 2D - 3D
  - Turbulent flows:
    - RANS models
    - LES models
    - **DNS**: in all previous DNS studies a 2D behavior has been assumed

**Our goal**: clarify the effects of the assumption of 2D over the statistics of a turbulent DHC flow
Differentially Heated Cavity (DHC) Problem (1/2)

General schema

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 (2/2)

Definition of our problem

\[ Ra_z = 6.4 \times 10^8 \]
\[ Pr = 0.71 \]
\[ A_z = 4 \]

- 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 Quéré carried out simulations for the same \( Pr, A_z \) and \( Ra_z = 6.4 \times 10^8, 2 \times 10^9 \) and \( 10^{10} \), 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.
- Periodic boundary conditions allow to study the 3D effects due to instability of the main flow and not to the boundary conditions.
- Our aim is to consider the problems with \( Ra_z 2 \times 10^9 \) and \( 10^{10} \) in the near future.
Two basic verifications are necessary in order to ensure the numerical results are reasonably close to the analytic solution

1. Verification of the code
   - Show that the code solves the governing equations with the expected order of accuracy
   - A method (MMS) based on the systematic discretization convergence tests using analytic solutions has been used

2. Verification of the simulation
   - In order to select the mesh size, integration period, depth length, etc, a compromise between accuracy and time computing must be made
   - It is important to evaluate if the numerical results obtained are “reasonably close” to the asymptotic solution
   - In our case, as the flow is chaotic, we are actually checking if the statistics of the flow are close to the asymptotic solution
Code verification using the MMS method (1/4)

Assume that the functions for the integration of mass and momentum equations are to be verified

MMS is carried out in three basic steps:

1. Generation of the **analytic velocity and source terms** fields:
   - An arbitrary analytic non-trivial function \( u_a(x, t) \) which accomplishes selected boundary conditions and incompressibility constraint (\( \nabla \cdot u_a = 0 \)) is chosen
   - The source term \( f_a(x, t) \) that would match with the arbitrary solution assuming that pressure gradient is null, is calculated analytically as:
     \[
     f_a = \frac{\partial u_a}{\partial t} + u_a \cdot \nabla u_a - Pr \nabla^2 u_a
     \]

2. Obtention of the **numerical solution**:
   - The analytic source term \( f_a(x, t) \) is evaluated at the discretization nodes and then used as input data for the numerical code.
3. Evaluation of the **numerical errors** and **order of accuracy** schemes verification:

- The discrete numerical solution $u_n(x, t)$ is compared with $u_a(x, t)$:

$$\|e\|_\infty = \|u_a - u_n\|_\infty = \text{Max} | u_a(x_i, y_j, z_k, t_n) - u_n(x_i, y_j, z_k, t_n) |$$

- This measure is repeated for systematically refined grids. As $\|e\|_\infty$ must tend to zero with an expected order of accuracy

$$\|e\|_\infty = C_t \Delta t^{p_t} + C_h \Delta h^{p_h} + H.O.T$$

Numerical results of $p_t$ and $p_h$ can be evaluated separately and compared with the theoretical values.

For the case of the mass-momentum system, since we assumed a null $\nabla p$ field, an additional verification is necessary.

For each $u^p$ there is **only one** $p$ scalar field (except a constant) such that $u = u^p - \nabla p$ is divergence-free ($\nabla \cdot u$).

Thus, if the velocity fields evaluated by the code are divergence free, the field $\nabla p$ must be correct.

However, it must be verified that the operator $\nabla \cdot$ is correct (i.e., with the expected order of accuracy).
Code verification using the MMS method (3/4)

Error versus mesh size $\Delta h = L_x/N_x$ for meshes concentrated in axis $y$ and $z$ with $N_x = N_y = N_z$. In parentheses the temporal order of accuracy ($p_h$).
Code verification using the MMS method (4/4)

Errors versus time step $\Delta t$. In parentheses the temporal order of accuracy ($p_t$).
Verification of the simulation (1/5)

After the absence of errors in the code has been ensured, it is necessary to determinate if the simulation parameters used allow to obtain numerical results reasonably close to the asymptotic solution.

The parameters are:

- Mesh size
- Mesh concentration
- Time step
- Domain length in the direction orthogonal to the main flow ($L_x$)
- Beginning of the averaging period ($t_0$)
- Integration time to evaluate the flow statistics ($\Delta t_a$)

For all them, a compromise between accuracy and computing time must be accepted.
Verification of the simulation (2/5)

A total of **eleven** 2D and 3D simulations, with meshes between $4.8 \times 10^4$ to $3.1 \times 10^6$ (for 3D) have been carried out and compared in order to estimate the accuracy of the results.

<table>
<thead>
<tr>
<th></th>
<th>$N_x$</th>
<th>$N_y$</th>
<th>$N_z$</th>
<th>$L_x$</th>
<th>$\gamma_y$</th>
<th>$\gamma_z$</th>
<th>$\Delta y_{min}$</th>
<th>$\Delta y_{max}$</th>
<th>$\Delta z_{min}$</th>
<th>$\Delta z_{max}$</th>
<th>$\Delta t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>64</td>
<td>156</td>
<td>312</td>
<td>1</td>
<td>1.5</td>
<td>1.5</td>
<td>$4.95 \times 10^{-4}$</td>
<td>$2.70 \times 10^{-3}$</td>
<td>$9.75 \times 10^{-4}$</td>
<td>$5.35 \times 10^{-3}$</td>
<td>$4.73 \times 10^{-8}$</td>
</tr>
<tr>
<td>B</td>
<td>32</td>
<td>78</td>
<td>156</td>
<td>1</td>
<td>2.0</td>
<td>2.0</td>
<td>$5.08 \times 10^{-4}$</td>
<td>$6.83 \times 10^{-3}$</td>
<td>$9.75 \times 10^{-4}$</td>
<td>$1.35 \times 10^{-2}$</td>
<td>$5.14 \times 10^{-8}$</td>
</tr>
<tr>
<td>B2X</td>
<td>64</td>
<td>78</td>
<td>156</td>
<td>2</td>
<td>2.0</td>
<td>2.0</td>
<td>$5.08 \times 10^{-4}$</td>
<td>$6.83 \times 10^{-3}$</td>
<td>$9.75 \times 10^{-4}$</td>
<td>$1.35 \times 10^{-2}$</td>
<td>$5.15 \times 10^{-8}$</td>
</tr>
<tr>
<td>C</td>
<td>16</td>
<td>39</td>
<td>78</td>
<td>1</td>
<td>2.0</td>
<td>2.0</td>
<td>$1.10 \times 10^{-3}$</td>
<td>$1.40 \times 10^{-2}$</td>
<td>$2.03 \times 10^{-3}$</td>
<td>$2.73 \times 10^{-2}$</td>
<td>$6.25 \times 10^{-8}$</td>
</tr>
<tr>
<td>A2D</td>
<td>-</td>
<td>156</td>
<td>312</td>
<td>-</td>
<td>1.5</td>
<td>1.5</td>
<td>$4.95 \times 10^{-4}$</td>
<td>$2.70 \times 10^{-3}$</td>
<td>$9.75 \times 10^{-4}$</td>
<td>$5.35 \times 10^{-3}$</td>
<td>$4.89 \times 10^{-8}$</td>
</tr>
<tr>
<td>B2D</td>
<td>-</td>
<td>78</td>
<td>156</td>
<td>-</td>
<td>2.0</td>
<td>2.0</td>
<td>$5.08 \times 10^{-4}$</td>
<td>$6.83 \times 10^{-3}$</td>
<td>$9.75 \times 10^{-4}$</td>
<td>$1.35 \times 10^{-2}$</td>
<td>$3.17 \times 10^{-8}$</td>
</tr>
<tr>
<td>C2D</td>
<td>-</td>
<td>39</td>
<td>78</td>
<td>-</td>
<td>2.0</td>
<td>2.0</td>
<td>$1.10 \times 10^{-3}$</td>
<td>$1.40 \times 10^{-2}$</td>
<td>$2.03 \times 10^{-3}$</td>
<td>$2.73 \times 10^{-2}$</td>
<td>$6.25 \times 10^{-8}$</td>
</tr>
<tr>
<td>AB2D</td>
<td>-</td>
<td>156</td>
<td>312</td>
<td>-</td>
<td>2.0</td>
<td>2.0</td>
<td>$2.44 \times 10^{-4}$</td>
<td>$3.38 \times 10^{-3}$</td>
<td>$4.78 \times 10^{-4}$</td>
<td>$6.70 \times 10^{-3}$</td>
<td>$1.19 \times 10^{-8}$</td>
</tr>
<tr>
<td>AA2D</td>
<td>-</td>
<td>218</td>
<td>438</td>
<td>-</td>
<td>1.5</td>
<td>1.0</td>
<td>$3.48 \times 10^{-4}$</td>
<td>$1.90 \times 10^{-3}$</td>
<td>$1.26 \times 10^{-3}$</td>
<td>$3.00 \times 10^{-3}$</td>
<td>$2.42 \times 10^{-8}$</td>
</tr>
</tbody>
</table>
Verification of the simulation (3/5)

First-order statistics

Comparison of $u_3$ in the section $z = \frac{15}{16} L_z$
Verification of the simulation (4/5)
Second-order statistics

Comparison of $u'_3 u'_3$ in the section $z = \frac{15}{16} L_z$ (where the discrepancies are largest).
Verification of the simulation (5/5)

Estimation of the integration period $\Delta t_a$ needed to evaluate first- and second-order statistics. $\Delta t_a > 2 \times 10^{-2}$ has been used in all the cases. (Xin and Le Quéré used $1.56 \times 10^{-2}$ for this $Ra$ number).

Maximum $u_3' u_3'$ at the central $z$ plane versus $\Delta t_a$
Instantaneous Isotherms
Temperature and Nusselt Instantaneous Fields
Averaged Temperature Field and Streamlines

3D

2D
Kinetic Energy Balances. Spectro-consistent discretization (1/4)

The **transport equation for kinetic energy**, \( e = \frac{1}{2} u \cdot u \) is obtained from the scalar product of velocity vector and momentum equation,

\[
\frac{\partial e}{\partial t} = -\nabla \cdot (e u) + Pr \nabla \cdot \left[ u \cdot \left( \nabla u + \nabla u^t \right) \right] - Pr \phi - \nabla \cdot (pu) + u \cdot f
\]

Where \( \phi (u) = (\nabla u + \nabla u^t) : \nabla u \). The term \(-Pr \phi\) is the **kinetic energy dissipation ratio** that arises from the viscous forces term \(Pr \nabla^2 u\).

Integration of previous expression in the domain \( \Omega \) yields:

\[
\frac{d}{dt} E = \int_{\partial \Omega} \left[ -e u + Pr u \cdot \left( \nabla u + \nabla u^t \right) + pu \right] \cdot dS + \int_{\Omega} [u \cdot f - Pr \phi] \, d\Omega
\]

Where \( E = \int_{\Omega} ed\Omega \) is the total kinetic energy.
Kinetic Energy Balances. Spectro-consistent discretization (2/4)

For our boundary conditions, the surface integral is null:

\[
\frac{d}{dt} E = \int_\Omega [u \cdot f - Pr \phi] \, d\Omega = \int_\Omega [Ra Pr Tu_3 - Pr \phi] \, d\Omega
\]

The only terms that contribute to the evolution of the total kinetic energy arise from:

- The **viscous term**: \(-Pr\phi\), that necessarily **dissipates kinetic energy** to thermal energy (as \(\phi \geq 0\)).
- The **body force term**: \(u \cdot f\) that can either **generate or dissipate kinetic energy**.
Kinetic Energy Balances. Spectro-consistent discretization (3/4)

Averaging for a long enough $\Delta t_a$, a global kinetic energy balance is obtained, that (per volume unit) is:

$$\frac{Ra}{V} \int_{\Omega} \left[ \overline{u_3 T} + \overline{u'_3 T'} \right] d\Omega = \frac{1}{V} \int_{\Omega} \left[ \phi (\overline{u}) + \phi (\overline{u'}) \right] d\Omega$$

That is, for a statistically stationary flow: $E_g = E_g$.

- $\overline{E_g}$, the averaged kinetic energy generation rate (only due to the bouyancy forces in our case)
- $\overline{E_d}$, the averaged kinetic energy dissipation rate due to viscous forces.
Kinetic Energy Balances. Spectro-consistent discretization (4/4)

A spectro-consistent or symmetry-preserving second-order discretization of the momentum equations (according to the formulation by Verstappen and Veldman, J. Comp. Physics, 2003) has been implemented.

In this formulation, the discretizations of the differential operators inherit the symmetry properties of the continuous operators, and therefore, the kinetic energy balances are exactly satisfied, even for coarse meshes.

In particular, the convective term does not dissipate kinetic energy.

This has been tested as an additional verification of the code.

The code is currently being extended to fourth order.
Second-order Statistics (1/4)
Turbulent kinetic energy and turbulent kinetic energy dissipation rate

\[ \overline{u_i u_i} \]

\[ \phi(u'); \phi(u) = (\nabla u + \nabla u^t) : \nabla u \]
Second-order Statistics (2/4)
Variance of temperature and ratio of turbulent kinetic energy generation

\[ \overline{T'T'} \quad \overline{u'_3T'} \]
Second-order Statistics (3/4)
Components of Reynolds stress tensor
Second-order Statistics (4/4)
Components of Reynolds stress tensor

2D

3D

2D

3D

\( \overline{u'_3 u'_3} \)

\( \overline{u'_2 u'_3} \)
Vertical profiles at $y = 0.5L_y$ of $u_2$, $u_3$ and $T$.
2D vs 3D profiles (2/2)

Averaged local Nusselt numbers and standard deviation of local Nusselt numbers.
Conclusions

- 2D and 3D direct numerical simulations of natural convection in a cavity have been carried out.
- To verify the accuracy of the results, eleven simulations with different meshes have been carried out.
- 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 $\nu$ are correctly predicted by the 2D simulations.
  - However, the second-order statistics are substantially different, far beyond the possible numerical inaccuracies.
  - The main differences are at the vertical boundary layers, where the 2D simulations incorrectly predicts very low values for all the second-order statistics.
- DSFD Poisson solver allows the use of loosely coupled parallel computers for DNS/LES.

The present results will be extended to higher $Ra$ numbers, using more accurate (fourth order) numerical schemes.
Please let me finish with a quote from the paper by Verstappen and Veldman (J.Comp.Physics, 2003):

“More than one and a half century ago Claude Navier (1822) and George Stokes (1845) derived and excellent mathematical model for turbulent flow”