Parallel CFD 2005
24-27 May 2005. University of Maryland, College Park, Maryland, USA

Direct Numerical Simulation of Turbulent Flows on low cost PC Clusters

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

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: cttc@cttc.upc.edu, web page: http://www.cttc.upc.edu
Presentation outline

- 1-Introduction. Loosely coupled parallel computers
- 2-Numerical methods
  - Governing equations
  - Kinetic Energy Balance. Symmetry-preserving discretization
  - Pressure velocity coupling
  - Alternatives to solve de Poisson equation
  - DSFD Poisson solver
  - Code verification and Simulation verification
- 3-Direct Numerical Simulations
  - Natural convection flows in air-filled cavities
    - DHC - $Ra = 6,4 \times 10^8$. Verification of the simulation and 2D vs 3D results.
    - DHC - $Ra = 2 \times 10^9$
    - DHC - $Ra = 10^{10}$
  - Flow past a square cylinder. $Re = 22,000$
- 4-Conclusions
Introduction - 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, unlike other applications, latency is often the most critical problem.
*We have developed DSFD algorithm to solve each Poisson equation with just one message*
Governing equations

Direct numerical simulation of turbulent natural convection flows.

Navier-Stokes coupled with energy transport equation.

\[
\frac{\partial \mathbf{u}}{\partial t} = - (\mathbf{u} \cdot \nabla) \mathbf{u} + \nu \nabla^2 \mathbf{u} - \frac{1}{\rho} \nabla p + \mathbf{f} \\
\frac{\partial T}{\partial t} = - (\mathbf{u} \cdot \nabla) T + \alpha \nabla^2 T \\
\nabla \cdot \mathbf{u} = 0
\]

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.
Global Kinetic Energy Balance. Symmetry-preserving discretization (1/3)

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, using \( \nabla \cdot u = 0 \)

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

Where \( \phi(u) = (\nabla u + \nabla u^t) : \nabla u \geq 0 \).

Integration of previous expression in the domain \( \Omega \) yields a global kinetic energy balance equation:

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

Where \( E = \int_{\Omega} e d\Omega \) is the total kinetic energy.
Global Kinetic Energy Balance. Symmetry-preserving discretization (2/3)

To focus the attention in the inner domain, let us consider the situations where the surface integral is null (e.g., periodic or $u = 0$):

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

We can see that not all the momentum equation terms contribute to the evolution of the total kinetic energy:

- The **viscous term** $-Pr\phi$, always **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.
- The **convective term** do **NOT contribute to the total kinetic energy**.
- The **pressure gradient term** do **NOT contribute to the total kinetic energy**.

All the discrete approximations of the Navier-Stokes equations satisfy the previous global kinetic energy balance equation for fine enough meshes.
In our DNS code, a spectro-consistent or symmetry-preserving discretization of the momentum equations (according to the formulation by Verstappen and Veldman, J. Comp. Physics, 2003) has been implemented. This aim of this formulation, rather than concentrating on reducing local truncation error, is to force the discrete operators to satisfy a discrete global kinetic energy balance equation that is equivalent to the previous continuous equation.

Therefore, the global kinetic energy balances are exactly satisfied, even for coarse meshes.

In particular, the convective term does NOT contribute to the global kinetic energy.

This absence of artificial dissipation is specially important for DNS codes. Artificial dissipation “interferes with the subtle balance between convective transport and physical dissipation, especially at the smallest scales of motion” ... distorting the essence of turbulence.
Symmetry-Preserving Discretization (1/3)

Discretized mass and momentum equations can be expressed in a generic form as:

\[
\begin{align*}
Mv &= 0 \\
H \frac{d \mathbf{v}}{dt} &= C \mathbf{v} + D \mathbf{v} + H \mathbf{f} + H \mathbf{P}^*
\end{align*}
\]

Where:
- \( \mathbf{v} \) is the discrete velocity vector (with the three components).
- \( \mathbf{p}^* \) is the discrete pressure vector.
- \( C \) is the non-linear convection operator (Note that actually \( C \) is a function of \( \mathbf{v} \)).
- \( D \) is the diffusive operator.
- \( M \) is the divergence operator.
- \( H \) is a diagonal matrix with the volume associated to each control volume.
- \( G \) is minus the discrete gradient.
- \( \mathbf{f} \) is the discrete body force vector.

Note that operators \( C, D, M \) include a volume integral.
Symmetry-Preserving Discretization (2/3)

The total discrete kinetic energy is defined as \( E = \frac{1}{2} \mathbf{v}^t \mathbf{H} \mathbf{v} \)

Its balance equation (obtained from discrete momentum equation) is:

\[
\frac{d}{dt} E = \mathbf{v}^t \left( \mathbf{C} + \mathbf{C}^t \right) \mathbf{v} + \mathbf{v}^t \left( \mathbf{D} + \mathbf{D}^t \right) \mathbf{v} + \mathbf{v}^t \mathbf{H} \mathbf{f} \mathbf{v} + \mathbf{p}^* \mathbf{G}^t \mathbf{H} \mathbf{v}
\]

We will force it to be equivalent to the continuous \( E \) equation:

\[
\frac{d}{dt} E = \int_{\Omega} \left[ -Pr \phi + \mathbf{u} \cdot \mathbf{f} \right] d\Omega
\]

Recall that this equation was obtained assuming that \( \nabla \cdot \mathbf{u} = 0 \), this is \( \mathbf{M} \mathbf{v} = 0 \).

Thus, we must impose:

- **Convection.** If \( \mathbf{v} \) verifies \( \mathbf{M} \mathbf{v} = 0 \), then \( \mathbf{v}^t \left( \mathbf{C} + \mathbf{C}^t \right) \mathbf{v} = 0 \)
- **Diffusion.** \( \mathbf{v}^t \left( \mathbf{D} + \mathbf{D}^t \right) \mathbf{v} \leq 0 \)
- **Pressure.** If \( \mathbf{v} \) verifies \( \mathbf{M} \mathbf{v} = 0 \), then \( \mathbf{p}^* \mathbf{G}^t \mathbf{H} \mathbf{v} = 0 \)
Symmetry-Preserving Discretization (3/3)

It can be verified that these conditions are satisfied only if:

- Non-linear **convection** operator $C$.
  Matrix $C$ is a function of $v$.
  If $Mv = 0$, then $v^t (C + C^t) v = 0$
  Therefore, $C$ must be **skew-symmetric**,
  - If $Mv = 0$, diagonal entries must be null
  - If $Mv = 0$, $C_{i,j} = -C_{j,i}$

- **Diffusion** operator $D$.
  For all $v$, $v^t (D + D^t) v \leq 0$
  $D$ must be:
  - Symmetric
  - Negative definite

- Relation between discrete **gradient** and **divergence** operators.
  If $Mv = 0$, then $p^t G^t H v = 0$
  Therefore, $M = -G^t H$ (recall that $G$ was defined as minus the gradient)
Pressure velocity coupling - time discretization (1/5)

Time discretization:

- Central difference is used for the time derivative term
- Fully explicit second order Adams-Bashforth scheme for $R$
- Implicit first-order Euler scheme for pressure-gradient term and mass-conservation equation

The discrete system to be solved is:

$$Mv^{n+1} = 0$$

$$\frac{v^{n+1} - v^n}{\Delta t} = \frac{3}{2}R^n - \frac{1}{2}R^{n-1} + Gp^{*n+1}$$

Where

$$R = Cv + Dv + Hf$$
Pressure velocity coupling (2/5)

The unknown velocity $v^{n+1}$ is isolated from the discrete momentum equation to obtain

$$v^{n+1} = v^p + G\tilde{p}$$

where the predictor velocity $v^p$, that can be explicitly evaluated, is

$$v^p = v^n + \Delta t \left[ \frac{3}{2} R^n - \frac{1}{2} R^{n-1} \right]$$

and $\tilde{p} = \Delta t \ p^{*n+1}$
Pressure velocity coupling (3/5)

To determinate $\tilde{p}$, discrete continuity constraint is imposed

$$Mv^{n+1} = Mv^p + MG\tilde{p} = 0$$

Combining gradient and divergence operators we obtain the well known discrete Poisson equation for the pressure

$$L\tilde{p} = -Mv^p$$

Where $L$ is the discrete Laplacian operator.
Pressure velocity coupling (4/5)

Discrete convective, gradient, divergence and Laplacian operators are related by the expressions:

- If $Mv = 0$, then $v_t (C + C^t) v = 0$
- $M = -G^tH$
- $L = MG$

Therefore, all them should be discretized consistently, e.g., if a fourth-order expression is used for $C$ in the momentum equation, $M$ and $L$ should also be fourth-order if a spectro-consistent scheme is desired.

Thus, the only algebraic equation that must be solved, $L\tilde{p} = -Mv^p$ must be discretized with a specific fourth order scheme.

This is a very unfortunate consequence, as the bandwidth of the fourth order Poisson equation is significantly higher than the bandwidth of the second order equation.

Is it still possible to solve it to machine accuracy with only one message?
The **algorithm** for the integration of each time step is:

1. Evaluate $\mathbf{R}^n = \mathbf{R}(\mathbf{v}^n)$
2. Evaluate $\mathbf{v}^p = \mathbf{v}^n + \Delta t \left[ \frac{3}{2} \mathbf{R}^n - \frac{1}{2} \mathbf{R}^{n-1} \right]$
3. Evaluate $\mathbf{Mv}^p$ and solve the Poisson equation.
4. Evaluate the velocity as $\mathbf{v}^{n+1} = \mathbf{v}^p + G\tilde{p}$. 
Why is Poisson equation so difficult for parallel computers?

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: \( Ax_i = b_i \quad i = 1 \cdot \cdot \cdot M \) where \( M \approx 10^{5 \cdot 7} \)
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 - Overview (1/2)

- DSFD is a combination of a FFT-based method and a Direct Schur method:
  - Fourier diagonalization is applied to reduce the 3D equation to a family of independent 2D equations (this imposes certain restrictions)
  - The 2D 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 $A$ is used
- After pre-processing, DSFD allows the solution of arbitrary order 3D Poisson equations to machine accuracy 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 $A$ 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.
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.
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
DSFD Conclusions

- After a pre-processing stage, DSFD methods allows to solve second- and fourth-order Poisson equations to machine accuracy with just one message.
- This allows the algorithm to run efficiently even on loosely coupled parallel computers, where it is significantly faster than Krylov subspace solvers.
- Even on PC clusters with 100 Mbits/s networks, the algorithm scales well up to 36-48 processors.
- DSFD algorithm has allowed us to run DNS simulations on a low cost PC cluster.
- The DSD part can be extended to non-structured meshes.
Code and Simulation Verifications

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
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 verification of the **order of accuracy** of the schemes:
   - The discrete numerical solution $u_n(x, t)$ is compared with $u_a(x, t)$:
     \[
     \|e\|_\infty = \|u_a - u_n\|_\infty = \max |u_a(x_i, y_j, z_k, t_n) - u_n(x_i, y_j, z_k, t_n)|
     \]
   - 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)
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 ($p_h$).
Errors versus time step $\Delta t$. In parentheses the temporal order of accuracy ($p_t$).
Error versus mesh size $\Delta h = L_x/N_x$ for meshes concentrated in axis $y$ and $z$ with $N_x = N_y = N_z$.

Observe that even for coarse meshes fourth-order discretization gives more accurate results.
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

\[ Ra_z = 6.4 \times 10^8, 2 \times 10^9, 10^{10} \]
\[ Pr = 0.71 \]
\[ A_z = 4 \]

- 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.
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)

For $Ra_z = 6.4 \times 10^8$, 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

3D

2D

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

3D

2D

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

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

![Graph showing the maximum $u'_3u'_3$ at the central z plane versus $\Delta t_\alpha$.]
Results for $Ra_z = 6.4 \times 10^8$. Instantaneous Isotherms
Results for \( Ra_z = 6.4 \times 10^8 \). Temperature and Nusselt Instantaneous Fields
Results for $Ra_z = 6.4 \times 10^8$. Averaged Temperature Field and Streamlines
Results for $Ra_z = 6.4 \times 10^8$. Second-order Statistics (1/4)

Turbulent kinetic energy and turbulent kinetic energy dissipation rate

\[ \overline{u'_i u'_i} \]

\[ \phi(u'); \phi(u) = (\nabla u + \nabla u^t) : \nabla u \]
Results for $Ra_z = 6.4 \times 10^8$. Second-order Statistics (2/4)

Variance of temperature and ratio of turbulent kinetic energy generation

$\overline{T'T'}$ $\overline{u'T'}$
Results for $Ra_z = 6.4 \times 10^8$. Second-order Statistics (3/4)

Components of Reynolds stress tensor
Results for $Ra_z = 6.4 \times 10^8$. Second-order Statistics (4/4)

Components of Reynolds stress tensor

\[ \overline{u'_3 u'_3} \quad \overline{u'_2 u'_3} \]
Results for $Ra_z = 6.4 \times 10^8$. 2D vs 3D profiles (1/2)

Vertical profiles at $y = 0.5L_y$ of $u_2$, $u_3$ and $T$. 

avg($u_2$), avg($u_3$)

avg(T)
Results for $Ra_z = 6.4 \times 10^8$. 2D vs 3D profiles (2/2)

Averaged local Nusselt numbers and standard deviation of local Nusselt numbers.
Results for $R\alpha_z = 2 \times 10^9$. Instantaneous Isotherms.

2D
Results for $Ra_z = 2 \times 10^9$. Instantaneous Isotherms.

3D
Results for $Ra_z = 10^{10}$. Instantaneous Isotherms.

3D
Flow past a square cylinder. Problem definition (1/2)

General schema

Boundary conditions:
- Prescribed laminar inflow.
- Non-slip boundary conditions at the horizontal walls.
- Periodic boundary conditions in the x direction, orthogonal to the main flow.
- Non-slip boundary conditions at the surface of the square cylinder.

Dimensionless governing numbers:
- \( Re = \frac{\rho UD}{\mu} \)
Flow past a square cylinder. Problem definition (2/2)

Definition of our problem

\[ Re = 2.2 \times 10^4 \]

\[ N = 64 \times 280 \times 210 \]
Instantaneous flows (1/4)

\[ t^* = 0.351 \times 10^{-3} \]
Instantaneous flows (2/4)

\[ t^* = 3.51 \times 10^{-3} \]
Instantaneous flows (3/4)

\[ t^* = 7.02 \times 10^{-3} \]
Instantaneous flows (4/4)

\[ t^* = 10.53 \times 10^{-3} \]
Conclusions

- **Main features** of our DNS code:
  - **Symmetry preserving fourth-order discretization.** The **global** kinetic energy balances are exactly satisfied, even for coarse meshes.
  - **Direct Schur-Fourier Decomposition** (DSFD) allows to solve each Poisson equation with just one message. It runs well even on PC clusters with conventional (100 Mbit/s) networks.

- **Verification of the code:** The MMS, based on the systematic discretization convergence tests using analytic solutions allows to “prove” that the code has second/fourth order convergence. Exact satisfaction of global kinetic energy balances has been used as an additional test.

- **Verification of the DNS simulations:** Ensure that the numerical results are reasonably close to the asymptotic solution. To do so, several simulations with different meshes are carried out and compared.

- **First application:** DNS of natural convection in a cavity. Effects of three-dimensional fluctuations. Previous simulations assumed the instantaneous flows to be 2D. In our work, the effect of the 3D fluctuations over the flow statistics has been evaluated.

- Of course, the code can also be used for other problems.
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 an excellent mathematical model for turbulent flow”
Symmetry-Preserving Discretization - Convective term (1/6)

We have shown that if we want that the discrete Navier-Stokes equations behave like the continuous equations respect to the global kinetic energy balance, then the discrete convection matrix $C$ must be skew-symmetric.

If a finite control volume discretization is used, this condition imposes restrictions about how the velocity values and the flow rates are evaluated at the control volume faces.

Note that from the imposition of a global conservation property, we have derived local conditions that the operators must satisfy.

In next slides, we will show:

- How to satisfy these local conditions in second-order and fourth-order formulations
- An example of a numerical scheme that violates these conditions and dissipates kinetic energy (UPWIND scheme)
Symmetry-Preserving Discretization - Convective term (2/6)

Basic second-order method

Convective discretization

\[
[Cv]_e \approx - \int_{\Omega_x} \nabla \cdot (uv) = - \int_{\partial \Omega_x} (uv) \cdot n
\]

\[
= - \int_E uu + \int_P uu - \int_{ne} uv + \int_{se} uv
\]

\[
[Cv]_e = - u_E f_E + u_P f_P - u_{ne} f_{ne} + u_{se} f_{se}
\]

\[
= c_e u_e + c_{ee} u_{ee} + c_w u_w + c_{Ne} u_{Ne} + c_{Se} u_{Se}
\]

Where \( c_e, c_{ee}, \ldots, c_{Se} \) are entries of matrix \( C \) corresponding to the row associated with \( u_e \), that are a function of \( v \) as convection operator is non-linear.
The approximations used to evaluate the \textit{velocities} and \textit{flow rates} at the control volume faces determinate the properties of \( C \).

To obtain a skew-symmetric \( C \), a mid-point interpolation is used for the face values of velocities and flow rates (\textit{even on a non-uniform mesh}), i.e.,

\[
    u_P = \frac{1}{2} (u_e + u_w)
\]

\[
    f_{se} = \frac{1}{2} (f_S + f_{SE}) = \frac{1}{2} (v_S \delta_{x-} + v_{SE} \delta_{x+})
\]
Symmetry-Preserving Discretization - Convective term (4/6)

This yields the following C entries,

\[ c_e = \frac{1}{2} (-f_E + f_P - f_{ne} + f_{se}) \]
\[ c_w = \frac{f_P}{2} \]
\[ c_{ee} = -\frac{f_E}{2} \]
\[ c_{Se} = \frac{f_{se}}{2} \]
\[ c_{Ne} = -\frac{f_{ne}}{2} \]

If \( Mv = 0 \), matrix C must be skew-symmetric, \( C + C^t = 0 \).

1. If \( Mv = 0 \), the diagonal entries are the \( c_P \) coefficients and they are null because they are the flow rate balances on the staggered mesh. They are a linear combination of the mass conservation equation on the main mesh

\[ c_e = \frac{1}{2} (-f_E + f_P - f_{ne} + f_{se}) \]
\[ = \frac{1}{4} (-f_E + f_w - f_{nP} + f_{sP}) + \]
\[ \frac{1}{4} (-f_{ee} + f_e - f_{nE} + f_{sE}) \]

2. Respect to the off-diagonal elements, the coefficients that affect to the same discrete velocity are equal and of opposite sign:

\[ c_{ee_{i,j}} = -c_{w_{i+1,j}} \]
\[ c_{Se_{i,j}} = -c_{Ne_{i,j-1}} \]

Therefore, in this case, matrix C is skew-symmetric.
Symmetry-Preserving Discretization - Convective term (5/6)

An example of Non-Symmetry-Preserving Discretization:
The First-order *Upwind-biased* Scheme

The face values of velocities are calculated as

\[
u_E = u_e \left[ \left( \frac{f_E}{|f_E|}, 0 \right) \right] + u_{ee} \left[ \left( \frac{-f_{ee}}{|f_{ee}|}, 0 \right) \right]
\]

where \([[] \ )\] returns the maximum.

This yields the following C matrix entries

\[
c_e = \left[ \left[ -f_P, 0 \right] \right] - \left[ \left[ f_E, 0 \right] \right] + \left[ \left[ -f_{se}, 0 \right] \right] - \left[ \left[ f_{ne}, 0 \right] \right]
\]
\[
c_w = \left[ \left[ f_P, 0 \right] \right]
\]
\[
c_{ee} = - \left[ \left[ -f_E, 0 \right] \right]
\]
\[
c_{Se} = \left[ \left[ f_{se}, 0 \right] \right]
\]
\[
c_{Ne} = - \left[ \left[ -f_{ne}, 0 \right] \right]
\]
Therefore,

- For non-null velocity fields, $c_e < 0$.
- $C$ is not a skew-symmetric matrix.
- Using Gershgorin theorem, it can be shown that $C$ is a negative semi-definite matrix.

Thus, the **global effect** of the convective term discretized with a first-order **Upwind-biased** scheme is to **dissipate kinetic energy**, and therefore it does not reproduce correctly the physics of the flow.
Fourier Diagonalization (1/8)
Second-order discretization

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

\[
\begin{align*}
\alpha_{i, j, k}^{p} x_{i, j, k}^{p} + \sum_{nb} \alpha_{i, j, k}^{nb} x_{i, j, k}^{nb} &= b_{i, j, k} \\
\end{align*}
\]

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:

\[
\begin{align*}
\mathbf{x}^{3d} &= \left[ x_{1,1}, x_{2,1}, \cdots x_{j,k}, \cdots, x_{N_y,N_z} \right]^{t} \\
\end{align*}
\]

where:

\[
\begin{align*}
\mathbf{x}_{j,k} &= \left[ x_{1,j,k}, x_{2,j,k}, \cdots, x_{N_x,j,k} \right]^{t} \\
\end{align*}
\]
Fourier Diagonalization (2/8)

Second-order discretization

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

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

- \(A_{j,k}^n\), \(A_{j,k}^s\), \(A_{j,k}^t\) 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. 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.

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

where \(\alpha = a_{j,k}^w = a_{j,k}^c\) and \(\beta = a_{j,k}^p\)
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 $A_{j,k}^P$

The product $x = Q^{-1}x$ inverse Fourier transform:

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

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

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

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

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

$$\bar{x}_N x = \sum_{i=1}^{N_x} x_i (-1)^i$$
Second-order discretization

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:

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

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

$$A^b_{j,k} \bar{x}_{j,k-1} + A^s_{j,k} \bar{x}_{j-1,k} + \lambda_{j,k} \bar{x}_{j,k} + A^n_{j,k} \bar{x}_{j+1,k} + A^t_{j,k} \bar{x}_{j,k+1} = \bar{b}_{j,k}$$

As the non-diagonal entries of matrices $A^p_{j,k}$ have been eliminated, unknown $\bar{x}_{i,j,k}$ is only coupled with unknowns in the same plane $i$.
Selecting of the $i$ component of each of the $N_yN_z$ block equations, we obtain a penta-diagonal scalar equation system in $\overline{x}$

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

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

1. **Direct FFT.** Calculate the $N_yN_z$ 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 $N_yN_z$ 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**.

$A_{p,j,k}^p$ are **heptadiagonal** $N_x \times N_x$ circulant matrices,

\[
A_{p,j,k}^p = \begin{bmatrix}
\beta & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_3 & \alpha_2 & \alpha_1 \\
\alpha_1 & \beta & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_3 & \alpha_2 \\
\alpha_2 & \alpha_1 & \beta & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_3 \\
\alpha_3 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_1 & \beta & \alpha_3 \\
\alpha_2 & \alpha_3 & \alpha_3 & \alpha_2 & \alpha_1 & \beta & \alpha_1 \\
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_3 & \alpha_2 & \alpha_1 & \beta \\
\end{bmatrix}
\]

where

\[
\begin{align*}
\alpha_1 &= a_{w1}^{p,j,k} = a_{e1}^{p,j,k} \\
\alpha_2 &= a_{w2}^{p,j,k} = a_{e2}^{p,j,k} \\
\alpha_3 &= a_{w3}^{p,j,k} = a_{e3}^{p,j,k} \\
\beta &= a_{p}^{p,j,k}
\end{align*}
\]
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 $\bar{x}$.

\[ \sum_{d=1}^{3} \left( a_{j,k}^{b_d} \bar{x}_{i,j,k-d} + a_{j,k}^{s_d} \bar{x}_{i,j-d,k} + a_{j,k}^{n_d} \bar{x}_{i,j+d,k} + a_{j,k}^{t_d} \bar{x}_{i,j,k+d} \right) + \bar{a}_{i,j,k}^p \bar{x}_{i,j,k} = \bar{b}_{i,j,k} \]
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 $A^{-1}b$
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:

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

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

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

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

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

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

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

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

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

   a) Solve \( t \) from \( A_{p,p} t = b_p \)
   b) \( \tilde{b}^p_s \leftarrow A_{s,p} t \)
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 ($10^5\cdots7$ 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 $x_p$
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 $\tilde{A}_{s,s}$ 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.
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 (2/9) - 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
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

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

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:**

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

- **Fourth-order:**

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

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 \( A_i \)
Stationary Iterative Approach for Schur Problems (3/3)

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.