FEM Techniques and High Performance Computing Approaches for Incompressible Flow Simulation

Sven H.M. Buijssen
Sven.Buijssen@math.uni-dortmund.de

Supervisors: Prof. Dr. S. Turek / Prof. Dr. H. Blum

Graduate School for Production Engineering and Logistics;
Institute for Applied Mathematics,
University of Dortmund
Motivation for CFD

What is Computational Fluid Dynamics?

What is its purpose in production engineering?
Motivation for CFD

What is Computational Fluid Dynamics?
Numerical simulation of flowfields by solving governing partial differential equations on computers

What is its purpose in production engineering?
Applications in aerodynamics, chemical engineering, material sciences, medicine, heat transfer ...

virtual windtunnel  bubble column reactor  combustion chamber


Existing CFD software

Already available CFD programs:
FLUENT®, CFX®, FEMLAB®, STARCD®, FeatFlow, UG, ...

Do we really need yet another CFD package?
For industrial problems in 3-D which are nonstationary and have complex geometries – yes

So, what is needed for really fast and accurate code?
Needs for state-of-the-art CFD

Modern mathematics has proven that CFD packages must provide:

• robust and accurate discretisations (capable of handling incompressibility, dominant convection, anisotropic meshes, singular perturbations ...)

• adaptive grid refinement/coarsening and a posteriori error control

• fast domain decomposition / multigrid solvers

None of the available CFD packages meets all of these demands.
Needs for state-of-the-art CFD

Modern mathematics has proven that CFD packages must provide:

- robust and accurate discretisations (capable of handling incompressibility, dominant convection, anisotropic meshes, singular perturbations ...)
- adaptive grid refinement/coarsening and a posteriori error control
- fast domain decomposition / multigrid solvers

None of the available CFD packages meets all of these demands.

Additionally:
Known deficiencies of existing software

• Commonly used: unstructured grids. Very flexible, but:
  • indirect addressing for matrices and vectors
    ⇒ No chance to exploit modern computer hardware: caching in, prefetching, pipelining
    ⇒ low MFlop/s-Rate even on Gigaflop processors
• Parallel implementations employing state-of-the-art multi grid or domain decomposition methods have known deficiencies:
  * sensitive to anisotropies
  * deterioration of convergence rates with increasing number of processors
    ⇒ low parallel efficiency due to communication and numerics
Main component in CFD software is a matrix–vector–multiplication (often 60-90% of CPU time).

<table>
<thead>
<tr>
<th>Computer</th>
<th>#Unknowns</th>
<th>MFLOP/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC 21264</td>
<td>33,280</td>
<td>100</td>
</tr>
<tr>
<td>(667 MHz)</td>
<td>133,120</td>
<td>58</td>
</tr>
<tr>
<td>‘EV67’</td>
<td>532,480</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>2,129,920</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>8,519,680</td>
<td>10</td>
</tr>
</tbody>
</table>

Peak performance: 1500 MFLOP/s
Known deficiencies of existing software

- Commonly used: unstructured grids. Very flexible, but:
  - indirect addressing for matrices and vectors
  - No chance to exploit modern computer hardware: caching in, prefetching, pipelining
  - Low MFlop/s-Rate even on Gigaflop processors

- Parallel implementations employing state-of-the-art multi grid or domain decomposition methods have known deficiencies:
  - sensitive to anisotropies
  - deterioration of convergence rates with increasing number of processors
  - Low parallel efficiency due to both communication and numerics
We can conclude that existing CFD software has either

• too poor numerical components

or

• too poor MFLOP/s rates per processor

or

• too bad parallel efficiency
to be as fast as it possibly could be.
The idea behind FEAST

HPC meets modern Numerics: Marry modern numerical mathematics, algorithmic approaches and implementational efficiency with hardware aspects to create really fast code.

Principles: (Recursive) Divide and Conquer strategy
- Find and exploit locally structured parts
  ⇒ gives high MFLOP/s rates
- Find and hide locally anisotropic parts
  ⇒ ensures robustness
Realisation

Realisation through an ensemble of **generalised tensor product meshes** plus so-called **ScaRC** as a generalised solver concept.

Use different solver methods for different kind of subgrids and couple them in a hierarchy.
Examples for Adaptive Meshing

‘Semi-unstructured grids consisting of many different (local) generalized tensorproduct meshes’
Examples for Adaptive Meshing

’Semi-unstructured grids consisting of many different (local) generalized tensorproduct meshes’

STEP 2
Examples for Adaptive Meshing

‘Semi-unstructured grids consisting of many different (local) generalized tensorproduct meshes’
Examples for Adaptive Meshing

‘Semi-unstructured grids consisting of many different (local) generalized tensorproduct meshes’

STEP 4
Realisation

Realisation through an ensemble of generalised tensor product meshes plus so-called ScaRC as a generalised solver concept.

Use different solver methods for different kind of subgrids and couple them in a hierarchy.
Example for ScaRC

For a typical grid with increasing distortions (unit: AR) ScaRC gives (parallel) convergence $\rho$ and (local) MFLOP/s rates as follows

<table>
<thead>
<tr>
<th>#NEQ-global</th>
<th>$AR$</th>
<th>$\rho$ (#IT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,704</td>
<td>3</td>
<td>0.03 (5)</td>
</tr>
<tr>
<td>431,317</td>
<td>10</td>
<td>0.09 (6)</td>
</tr>
<tr>
<td>9,344,533</td>
<td>10</td>
<td>0.09 (6)</td>
</tr>
<tr>
<td>37,366,805</td>
<td>$10^7$</td>
<td>0.09 (6)</td>
</tr>
</tbody>
</table>

Global (parallel) convergence rates

<table>
<thead>
<tr>
<th>#NEQ-local</th>
<th>MV-V/C</th>
<th>MGTRI-V/C</th>
</tr>
</thead>
<tbody>
<tr>
<td>4,225</td>
<td>245/1275</td>
<td>299/706</td>
</tr>
<tr>
<td>66,049</td>
<td>193/638</td>
<td>173/382</td>
</tr>
<tr>
<td>1,050,625</td>
<td>168/614</td>
<td>146/288</td>
</tr>
</tbody>
</table>

Local MFLOP/s rates (AMD ‘XP 1500+’)

University of Dortmund – Institute of Applied Mathematics – Graduate School Talk, 2004/04/15
FEAST - Conclusion

With FEAST there is now a very efficient solver engine for scalar problems available.
With FEAST there is now a very efficient solver engine for scalar problems available.

But FEAST is only a starting point, a basic module for scalar problems.
Aim of my project

My project will cover the following aspects:

• Create a complete CFD solver for solving the incompressible nonstationary Navier–Stokes equations

\[ u_t - \nu \Delta u + (u \cdot \nabla) u + \nabla p = f, \quad \nabla \cdot u = 0 \]

based on a discrete projection method which brings the system of equations down to a sequence of scalar problems for velocity and pressure (Burgers equations, Pressure-Poisson equation)

• Deal with adaptivity
  ⇒ see next talk of M. Grajewski

• Improve the load balancing of FEAST and, therewith, of my CFD application
Conclusion

Thank you for your attention.