PFEM-2
Towards Massively Parallel Simulations

J. M. Gimenez¹,² N. Nigro¹,² S. Idelsohn²,³

¹Facultad de Ingeniería y Ciencias Hídricas
Universidad Nacional del Litoral

²Research Center on Computational Mechanics (CIMEC)
UNL/CONICET, Predio Conicet Litoral Centro, Santa Fe, Argentina

³International Center for Numerical Methods in Engineering (CIMNE)
Edificio C1, Campus Norte UPC C/ Gran Capitán S/N 08034 Barcelona, Spain.

Mendoza - July 2013
Presentation brief

1. PFEM-2 - Algorithm Revision
2. Distributed Memory Implementation
3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube
4. Conclusions and Future Work
1 PFEM-2 - Algorithm Revision

2 Distributed Memory Implementation

3 Tests
   • Flow Around a Cylinder 2d
   • Wall Mounted Cube

4 Conclusions and Future Work
Presentation brief

1. PFEM-2 - Algorithm Revision
2. Distributed Memory Implementation
3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube
4. Conclusions and Future Work
Presentation brief

1. PFEM-2 - Algorithm Revision

2. Distributed Memory Implementation

3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube

4. Conclusions and Future Work
Presentation brief

1. PFEM-2 - Algorithm Revision

2. Distributed Memory Implementation

3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube

4. Conclusions and Future Work
PFEM-2 = PFEM + some new ideas

Particle/Mesh based method to solve transport equations.

- Enlarge $\Delta t$ as much as possible ($CFL \gg 1$ & $Fo \gg 1$) for stability reasons (robustness).
- Try to drastically reduce the CPU time against standard current CFD software available: from days to hours or from hours to minutes.
- Using first LARGE $\Delta t$ to select some possible solutions among many others and FINALLY adapt the time step for accuracy needs.
X-IVAS: eXplicit Integration of Velocity and Acceleration following the Streamlines

- A better particle trajectory integration (X-IVS) following streamlines.
- Resolving more difficult details of the flow with high accuracy.
- Extended to particle velocity integration (X-IVAS).
- Reducing drastically the time step restriction caused by the non-linearities.

\[
\begin{align*}
\mathbf{x}_{p}^{n+1} &= \mathbf{x}_{p}^{n} + \int_{n}^{n+1} \mathbf{v}^{\alpha}(\mathbf{x}_{p}^{\tau}) \, d\tau. \\
\hat{\mathbf{x}}_{p}^{n+1} &= \mathbf{x}_{p}^{n} + \mathbf{v}^{n}(\mathbf{x}_{p}^{n}) \Delta t \\
\mathbf{y}_{p}^{n+1} &= \mathbf{x}_{p}^{n} + \sum_{i=1}^{N} \mathbf{v}^{n}(\mathbf{y}_{p}^{n+\frac{i}{N}}) \delta t
\end{align*}
\]
X-IVAS with locally adaptive time-step

for each particle:

\[ x_{p}^{n+1} = x_{p}^{n} + \sum_{i=1}^{N} v^{n}(x_{p}^{n} + \frac{i}{N}) \delta t_{p} \]

where

\[ \delta t_{p} = \frac{\Delta t}{K \times CFL_{h}} = \frac{h}{K|v|} \]

\( CFL_{h} \) is the local value of the element which contains the particle
\( K \) is a parameter to adjust the minimal number of sub-steps required to cross an element
Screenshot of the method

1. **Acceleration Stage**: Calculate acceleration on the nodes - **Mesh**
2. **X-IVAS Stage**: Evaluate new particles position and state with X-IVAS - **Particles**
3. **Projection Stage**: Project state from particles to the mesh - **Particles**
4. **Implicit Diffusion Stage**: Implicit correction of the viscous diffusion - **Mesh**
5. **Poisson Stage**: Solving a Poisson equation system for pressure - **Mesh**
6. **Correction Stage**: Update states with corrections: - **Mesh and Particles**
Presentation brief

1. PFEM-2 - Algorithm Revision

2. Distributed Memory Implementation

3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube

4. Conclusions and Future Work
**Need for large Simulations**

- *Real* problems are in three dimensions.
- Supposing a 3d mesh of $5 \times 10^6$ tetrahedra and around $1 \times 10^6$ nodes:
  - FDM-FEM-FVM only stores grid/mesh data: 
    - 200 bytes per element 
    - 200 bytes per node 
    - then at least 2Gb of RAM memory.
  - PFEM-2 requires the same info + particle data: 
    - 10 particles per element 
    - 100 bytes per particle 
    - then at least 12Gb of RAM memory.

We need a distributed memory implementation
Need for large Simulations

- Real problems are in three dimensions.
- Supposing a 3d mesh of $5 \times 10^6$ tetrahedra and around $1 \times 10^6$ nodes:
  - FDM-FEM-FVM only stores grid/mesh data:
    - 200 bytes per element
    - 200 bytes per node
    - then at least 2Gb of RAM memory.
  - PFEM-2 requires the same info + particle data:
    - 10 particles per element
    - 100 bytes per particle
    - then at least 12Gb of RAM memory.

We need a distributed memory implementation
**Implementation Features**

Based on the FEM library **libMesh**

<table>
<thead>
<tr>
<th>PFEM-2 approach</th>
<th>Fixed-Mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh distribution</td>
<td>weighted domain decomposition (Metis)</td>
</tr>
<tr>
<td>Particles distribution</td>
<td>static</td>
</tr>
<tr>
<td>$Ax = b$ Solvers</td>
<td>Krylov solvers (PETSc)</td>
</tr>
<tr>
<td>$Ax = b$ Preconditioners</td>
<td>PETSc (allows user-defined)</td>
</tr>
<tr>
<td>100K elements - 1M particles</td>
<td>1 GB</td>
</tr>
<tr>
<td>Max problem size solved</td>
<td>6M elements - 60M particles</td>
</tr>
<tr>
<td>I/O formats</td>
<td>raw - UNV - VTK - Nemesis(parallel) ...</td>
</tr>
</tbody>
</table>

**What can we solve?**

- Scalar Transport
- Navier Stokes $\Rightarrow$ laminar and turbulent
- Thermal Coupling (NS+ST) $\Rightarrow$ laminar and turbulent
Mesh-based methods use:
- domain-decomposition (DD)

Particle methods use:
- atom-decomposition (ATOM)
- domain-decomposition (DD)

DD+Atom requires an updated copy of the entire mesh in each processor
⇒ DD+DD is chosen
Mesh-based methods use:
- domain-decomposition (DD)

Particle methods use:
- atom-decomposition (ATOM)
- domain-decomposition (DD)

DD+Atom requires an updated copy of the entire mesh in each processor
⇒ DD+DD is chosen
Inter-Processors communication?

- Mesh-based methods use:
  - Ghost layers
- Our particle method needs:
  - Several ghost layers .. how many?
  - Interchange particles

We are using **synchronous** transference: the particles are stored in a buffer and are sent at the end of the loop (requires external loop until all particles have completed their trajectories).
Domain-distribution target:

computations balanced and communication minimized

We can use weighting factors per element $\eta(v)$ in the partitioner to balance the work-load:

- In Mesh-based steps: $\eta_n(v_j) = \#dof_j$
- In X-IVAS step: $\eta_w(v_j) \approx K \times (CFL_h)_j \Rightarrow$ useful because CFL varies depending on the mesh refinement and flow state.

Some questions:

- Are the proposed weighting factors really good?
- What of them we have to use?
1. PFEM-2 - Algorithm Revision

2. Distributed Memory Implementation

3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube

4. Conclusions and Future Work
Presentation brief

1. PFEM-2 - Algorithm Revision

2. Distributed Memory Implementation

3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube

4. Conclusions and Future Work
Case Description

- $Re = 1000$
- $D = 1$
- $CFL_{max} \approx 10 - 15$
- Beowulf Cluster server Intel i7-2600K 8Gb RAM and six nodes i7-3930K 16Gb RAM connected by Gigabit Ethernet
PETSc scales well with at least $\sim 100,000$ dof per MPI process.
3D - $1.6 \times 10^6$ tetrahedra - $3.5 \times 10^5$ nodes - $16 \times 10^6$ particles

Total: $S_{16}(\eta_n(v)) = 10.45$ - only X-IVAS: $S_{16}(\eta_n(v)) = 9.69$

Total: $S_{16}(\eta_w(v)) = 8.77$ - only X-IVAS: $S_{16}(\eta_n(v)) = 11.19$

but X-IVAS represents the $\approx 25\%$ of the computational cost.
Comparisons

We use the largest $\Delta t$ which preserves algorithmic stability and accuracy. Simulated $T = 1[\text{s}]$.

<table>
<thead>
<tr>
<th>Solver</th>
<th>$\Delta t$</th>
<th>$C_{o_{mean}}$</th>
<th>$C_{o_{max}}$</th>
<th>$S_{16}$</th>
<th>CPU-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFEM-2 (3d)</td>
<td>0.05[\text{s}]</td>
<td>$\approx 0.75$</td>
<td>$\approx 8$</td>
<td>10.55x</td>
<td>197.66[s]</td>
</tr>
<tr>
<td>OpenFOAM® (3d)</td>
<td>$\approx 0.025[\text{s}]$</td>
<td>$\approx 0.5$</td>
<td>$\approx 10$</td>
<td>9.41x</td>
<td>613.98[s]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strouhal</th>
<th>$C_d$</th>
<th>$C_l$ amplitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental</td>
<td>0.21</td>
<td>1.02</td>
</tr>
<tr>
<td>Mittal (3d)</td>
<td>0.2</td>
<td>1.18</td>
</tr>
<tr>
<td>PFEM-2 (3d)</td>
<td>0.185</td>
<td>1.16</td>
</tr>
<tr>
<td>OpenFOAM® (3d)</td>
<td>0.195</td>
<td>1.22</td>
</tr>
</tbody>
</table>

Using AMG+PCG for Poisson solvers ($tol = 10^{-6}$)
Presentation brief

1. PFEM-2 - Algorithm Revision

2. Distributed Memory Implementation

3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube

4. Conclusions and Future Work
Case Description

- $Re = 3200$
- LES Turbulence Model (Static Smagorinsky)
- cube represented by 50x50x50 nodes with 2065115 elements against a 240x240x128 grid used in the reference
- first order in time against third order Runge-Kutta in the reference
Fig. 9.6. The streamlines in the vertical center plane of the flow over a wall-mounted cube; from Shah and Ferziger (1997)
Results - Close to lower wall

- Video $\nu^t$
- Video Streamlines

Fig. 9.5. The streamlines in the region close to the lower wall of the flow over a wall-mounted cube; from Shah and Ferziger (1997)
1. PFEM-2 - Algorithm Revision

2. Distributed Memory Implementation

3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube

4. Conclusions and Future Work
Conclusions

- Weighted Partitioning to Improve Scalability:
  - Laminar Flows - Diffusive dominant - Low \( \text{Re} \propto \frac{\text{CFL}}{\text{Fo}_\nu} \Rightarrow \eta_n \)
  - Laminar Flows - Convective Dominant - Medium \( \text{Re} \propto \frac{\text{CFL}}{\text{Fo}_\nu} \Rightarrow \eta_w \)
  - Turbulent Flows - Large \( \text{Re} \propto \frac{\text{CFL}}{\text{Fo}} \) but appears \( \text{Fo}_{\nu t} \Rightarrow \eta_n \)

Reduced scope for \( \eta_w \Rightarrow \) we only use \( \eta_n \)

- WE REACH \( S_{16} \approx 10.5 \times \) AND COMPARING WITH OpenFOAM® WE ARE 3\( \times \) ABOVE WITH SIMILAR SCALABILITY \Rightarrow \) Gigabit Ethernet Limitations on the scalability?

- Numerical method issues: X-IVAS with higher order in time?, Development of better projection operators, more testing on turbulence modeling and coupled problems in terms of accuracy & efficiency.

\( \Rightarrow \) A promissory beginning towards massively parallel simulations
Thanks for your attention, questions are welcome