PFEM-2

Towards Massively Parallel Simulations

J. M. Gimenez[1,2] N. Nigro[1,2] S. Idelsohn[2,3]

[1]Facultad de Ingeniería y Ciencias Hídricas
Universidad Nacional del Litoral

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

[3]International 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

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

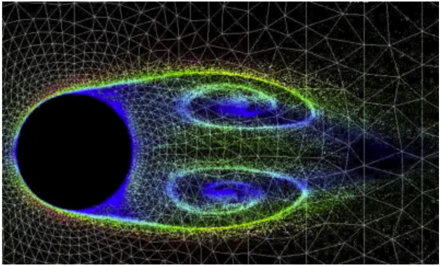
Presentation brief

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

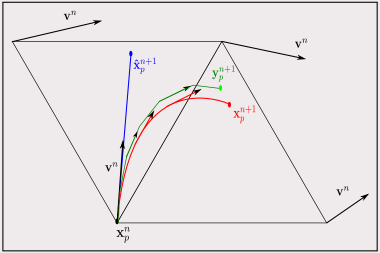
Particle/Mesh based method to solve transport equations.

- Enlarge Δt 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 Δt 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.



$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \int_n^{n+1} \mathbf{v}^{\alpha}(\mathbf{x}_p^{\tau})\, d\tau.$$

$$\hat{\mathbf{x}}_p^{n+1} = \mathbf{x}_p^n + \mathbf{v}^n(\mathbf{x}_p^n)\Delta t$$

$$\mathbf{y}_p^{n+1} = \mathbf{x}_p^n + \sum_{i=1}^{N} \mathbf{v}^n(\mathbf{y}_p^{n+\frac{i}{N}})\, \delta t$$

X-IVAS with locally adaptive time-step

for each particle:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \sum_{i=1}^{N} \mathbf{v}^n(\mathbf{x}_p^{n+\frac{i}{N}})\ \delta t_p$$

where

$$\delta t_p = \frac{\Delta t}{K \times CFL_h} = \frac{h}{K|\mathbf{v}|}$$

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 form 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

Need for large Simulations

- *Real* problems are in three dimensions.
- Supposing a 3d mesh of $5x10^6$ tetrahedra and around $1x10^6$ 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 $5x10^6$ tetrahedra and around $1x10^6$ 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

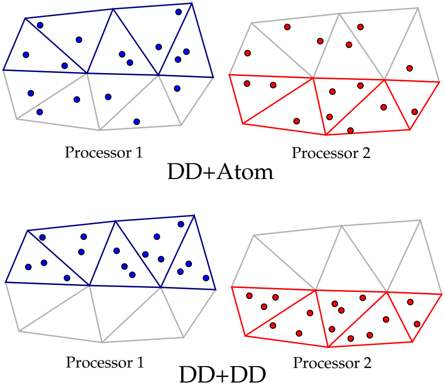
Based on the FEM library libMesh

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

What can we solve?

- Scalar Transport
- Navier Stokes $\Rightarrow$ laminar and turbulent
- Thermal Coupling (NS+ST) $\Rightarrow$ laminar and turbulent

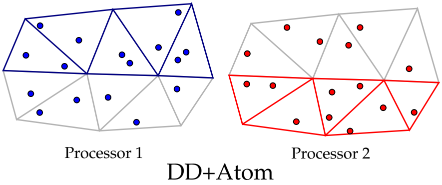
- Mesh-based methods use:
  - domain-decomposition (DD)
- Particle methods use:
  - atom-decomposition (ATOM)
  - domain-decomposition (DD)

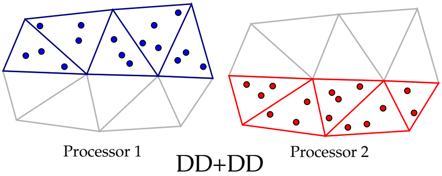


DD+Atom requires an updated copy of the entire mesh in each processor
$\Rightarrow$ DD+DD is chosen

Data Distribution

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

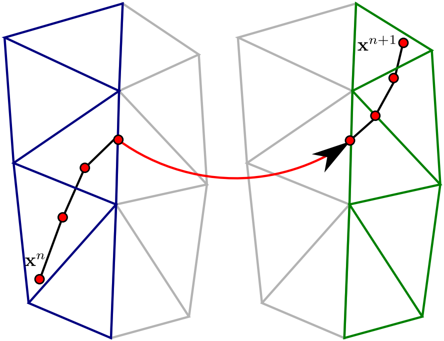




DD+Atom requires an updated copy of the entire mesh in each processor
$\Rightarrow$ DD+DD is chosen

Inter-Processors communication?

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

Work-load Balancing

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?

Presentation brief

1. PFEM-2 - Algorithm Revision

2. Distributed Memory Implementation

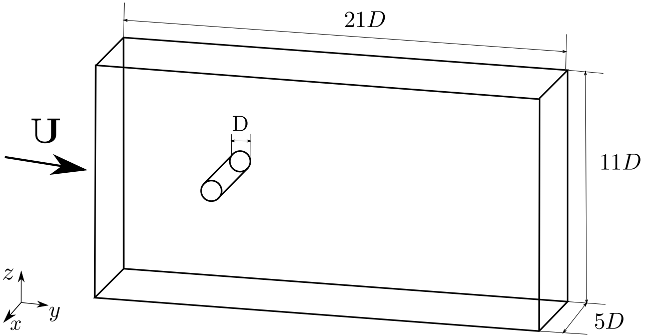
3. Tests
   - Flow Around a Cylinder 2d
   - Wall Mounted Cube

4. Conclusions and Future Work

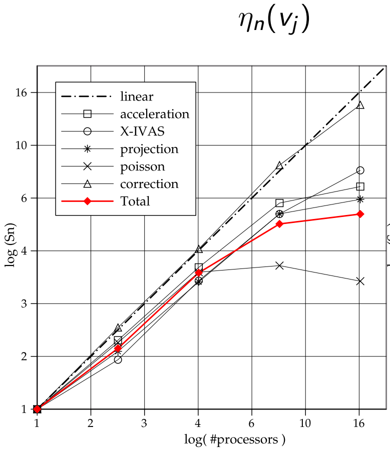
Presentation brief

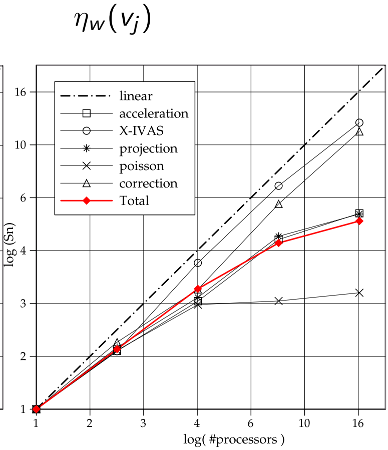
Case Description

- $Re = 1000$
- $D = 1$
- $CFL_{max} \approx 10 - 15$
- Beowulf Cluster server Intel i7-2600K 8Gb RAM and six nodes i7-3930K 16Gb RAM connected by Gigabit Ethernet



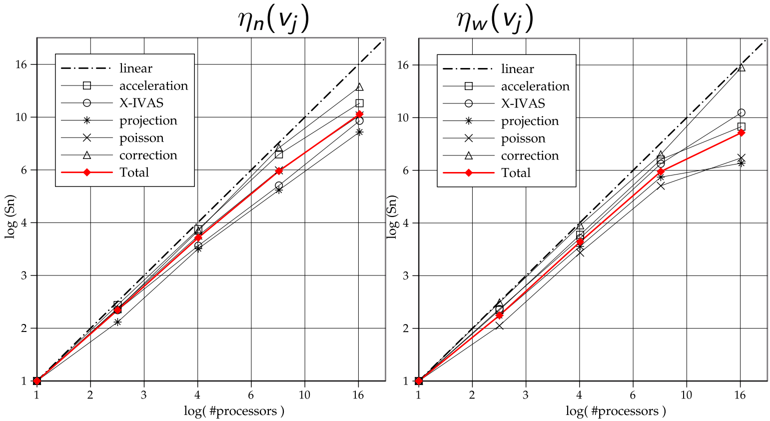
2D - 8.8x10^4 triangles - 4.3x10^4 nodes - 8x10^5 particles





PETSc scales well with at least $\sim 100,000$ dof per MPI process.

3D - 1.6×10^6 tetrahedra - 3.5×10^5 nodes - 16×10^6 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 Δt which preserves algorithmic stability and accuracy. Simulated $T = 1[s]$.

Solver	Δt	Co_{mean}	Co_{max}	S_{16}	CPU-time
PFEM-2 (3d)	$0.05[s]$	≈ 0.75	≈ 8	10.55x	197.66[s]
OpenFOAM® (3d)	$\approx 0.025[s]$	≈ 0.5	≈ 10	9.41x	613.98[s]

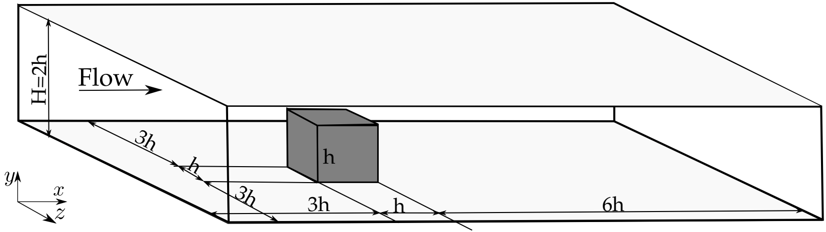
	Strouhal	$\overline{C_d}$	C_l amplitude
Experimental	0.21	1.02	-
Mittal (3d)	0.2	1.18	0.1 to 0.3
PFEM-2 (3d)	0.185	1.16	0.2 to 0.3
OpenFOAM® (3d)	0.195	1.22	0.5

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

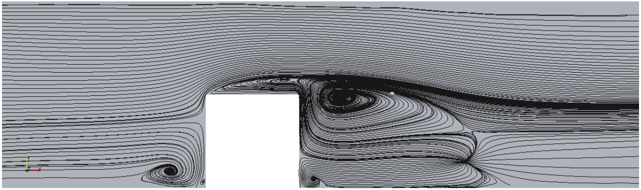
Presentation brief

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



Results - Vertical center plane



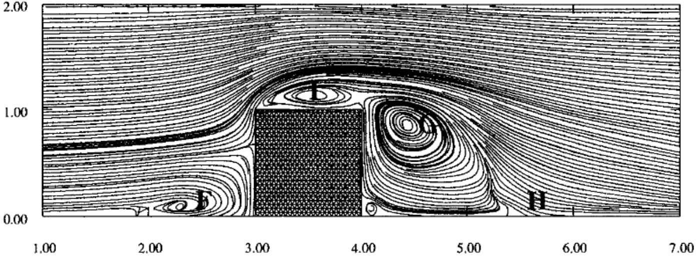
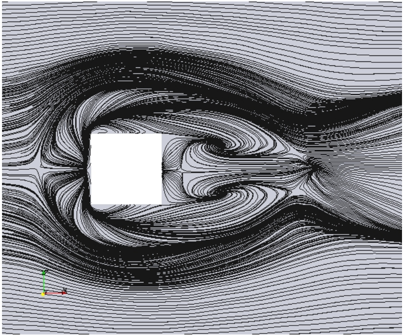


Fig. 9.6. The streamlines in the vertical center plane of the flow over a wall-mounted cube; from Shah and Ferziger (1997)



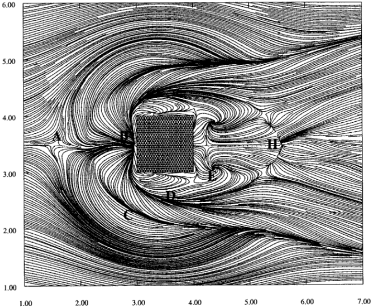


Fig. 9.5. The streamlines in the region close to the lower wall of the flow over a wall-mounted cube; from Shah and Ferziger (1997)

- Video ν^t
- Video Streamlines

Presentation brief

Conclusions

- Weighted Partitioning to Improve Scalability:
  - Laminar Flows - Diffusive dominant - Low $Re \propto \dfrac{CFL}{Fo_\nu} \Rightarrow \eta_n$
  - Laminar Flows - Convective Dominant - Medium $Re \propto \dfrac{CFL}{Fo_\nu} \Rightarrow \eta_w$
  - Turbulent Flows - Large $Re \propto \dfrac{CFL}{Fo}$ but appears $Fo_{\nu^t} \Rightarrow \eta_n$

  Reduced scope for $\eta_w \Rightarrow$ we only use η_n
- WE REACH $S_{16} \approx 10.5x$ AND COMPARING WITH OpenFOAM® WE ARE $3x$ 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