Aerodynamic simulation of flow through porous media based on Lattice Boltzmann Method

Jiaxing Qi
15.12.2014
Abstract

• We have developed an efficient and parallel incompressible Navier-Stokes equation solver - MUSUBI, to investigate the problem of sound generation through porous media.
• An interpolation-based grid refinement technique for LBM has been proposed to handle this multi-scale aero acoustic problem.
• To illustrate the efficiency of our code, we present its performance on the latest HLRS parallel system - Hornet.
• At last we will present preliminary results of flow through a porous media.
Contents

• Sound generation by flow through silencer
• Lattice Boltzmann Solver - MUSUBI
• Optimization design
• Grid refinement technique
• Performance analysis
• Preliminary results of flow through porous silencer
• Conclusion and outlook
Motivation

- Silencer is used to reduce noises emitted from technical devices.
- Movement of fluid (air) induce sound generation.
- CFD is utilized as a tool to predict aerodynamics noise from outflow of a silencer made of a porous medium.
Challenges

1. To handle very large data sets efficiently at reasonable cost on supercomputer.
2. To simulate flows in complex geometry
3. To resolve multi-scale areas:
   - Flow through porous medium \( \sim O(\mu m) \)
   - Sound propagating to far field \( \sim O(cm) \)

Highly parallel octree-based Lattice Boltzmann solver - MUSUBI
MUSUBI in the APES tool-chain

- Goals: integrated approach for large-scale mesh-based simulations.
- Efficient and Parallel tools:
  - mesh generation → solvers → post-processing
- TreELM: linearized octree data structure.
Lattice Boltzmann Method

• It describes incompressible fluid flows at kinetic level.
  – Cartesian uniform grids (lattice).
  – Each element has 19(3D) or 9(2D) state values.

• Algorithm is simple, suitable for parallel:

  ![Lattice Boltzmann Method Diagram]

• Easy to handle complex geometry.
• LBM has been applied by many people to solve aerodynamics problems.
Optimization of compute kernel

- **Memory**: Only fluid elements are stored.
- **Locality**: Elements arranged according to space-filling curve.
- **Vectorization**: Data stored in 1D array with connectivity list.
- **Combined stream-collision**.
- **Implicit solid boundary treatment**.

**Distribution links (D3Q19):**

```
real(kind=rk):: state(19*nElems)
```

**Connectivity List**

```
integer :: neigh(19*nElems)
```

**Lattice with obstacles**
Grid refinement

- Introduce ghost elements at level interface to perform steam-collision.
- Ghost are interpolated (linear or quadratic) by fluid elements on other level
  - From coarse to fine
  - From fine to coarse
- Goals of interpolation: to maintain the continuity of pressure, velocity and shear stress over level interface.
- Two ways of scaling:
  - “Acoustic scaling”: $dt \sim dx$
  - “Diffusive scaling”: $dt \sim dx^2$
Performance analysis
How to measure performance

- Absolute performance in terms of products:
  - $MLUPS$ (= million lattice updates per second)
- Relative performance in parallel:
  - $MLUPSpP = \frac{MLUPS}{\# \text{ processes}}$
  - $MLUPSpN = \frac{MLUPS}{\# \text{ nodes}}$
- Shown in performance map:
  - X axis: number of lattices (elements) per node
  - Y axis: $MLUPSpN$
- Ideal scaling: serial performance replicated in parallel
  - $MLUPSpN = \text{constant}$
MUSUBI Performance on Hornet XC40

- Intel Haswell processors with 24 cores per node.
- Intel Fortran compiler 15
- Uniform mesh without any boundary
- Intra-node performance:
  - 1 node with 1-48 processes.
  - How many processes in one node can give best performance?
- Inter-node performance:
  - Up to 2048 nodes
- Performance with grid refinement
Intra-node performance

Hornet Intranode performance (intel)

- MLUPS/node vs nElems/proc

Graph showing performance metrics for different node counts and element counts.
Inter-node performance
Performance on locally refined grids

Porous medium in a channel

- Total computing effort:
  - time * nNodes

Memory: 75 GB → 4.6 GB
Preliminary results of flow through porous silencer
Mesh generation using Seeder

From geometry definition to the final simulation mesh

* Picture from Manuel’s thesis
Simulation target domain

* Picture from Manuel’s thesis
## Configuration

<table>
<thead>
<tr>
<th></th>
<th>Final target</th>
<th>Preliminary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh level range</td>
<td>5 levels (L8 - L12)</td>
<td>4 levels (L9 - L12)</td>
</tr>
<tr>
<td>Inlet pressure</td>
<td>0.3125 bar</td>
<td>0.3125 bar</td>
</tr>
<tr>
<td>Outlet</td>
<td>Free flow</td>
<td>Free flow</td>
</tr>
<tr>
<td>Physical time</td>
<td>33 ms or even longer</td>
<td>5 ms</td>
</tr>
<tr>
<td>Number of elements</td>
<td>130 M</td>
<td>119 M</td>
</tr>
<tr>
<td>Machine</td>
<td>Hornet / Hermit</td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>64 nodes, &lt;1h / 1ms (Hornet)</td>
<td>128 nodes, 3h / 1ms (Hermit)</td>
</tr>
</tbody>
</table>
Diffusive linear: pressure
Diffusive linear: velocity
Diffusive quadratic: pressure
Diffusive quadratic: velocity
Conclusions and outlook

- MUSUBI enables efficient computation on complex geometries.
- CPU and memory savings by locally refined grids.
- Large multi-scale simulations
- Future work:
  - Adaptive mesh refinement
  - Dynamic load balancing
the End

Thank you!