Parallel Dynamic Simplical Meshes in OpenFOAM

David P. Schmidt
and
Sandeep Menon

June, 2011
6th OpenFOAM® Workshop
State College, Pennsylvania
Goal

- To provide a very general adaptive-meshing approach to handle:
  - Large and complex domain deformation
  - Unstructured meshes
  - A priori unknown domain deformation
  - Variable length scales
  - Parallel execution

- Requires hands-off functionality
  - Provide inputs to control resolution and special boundary treatments
Methodology

- Simplical meshes: the simplest shapes

- **Phase 1: Mesh motion**
  - Optimization-based

- **Phase 2: Mesh reconnection**
  - Optimization-based
The enemies

Degenerate tetrahedra with reasonable edge lengths

(I) Sliver  (II) Cap  (III) Spade  (IV) Wedge

-Freitag and Knupp, 2002
Phase 1: Mesh motion

- Conforms to moving boundaries
- Is easy to include in conservation laws (relative vs. absolute fluxes)
- For small deformation, mesh motion is entirely sufficient
Mesh motion choices

- Simple Laplacian smoothing (as if points were connected by springs)
  - Not too bad for 2D; cheap
  - Fails frequently in 3D; not tied to cell quality
- FEM-based decomposition (Jasak and Tukovic)
  - Intrinsic OpenFOAM
  - Avoids inverted cells
- MESQUITE
  - Open-source mesh motion package from Sandia National Lab
  - Based on optimizing cell quality: LOTS of options
Calculate quality of each cell

Calculate a global norm of quality for the mesh

Optimize this norm

We parallelized the process using a halo approach
The halo approach
Surface smoothing

- Must work in parallel
- Handles curved surfaces
Phase 2: Mesh Reconnection

- For large deformation, mesh motion is not sufficient
  - Need evolves for more resolution
  - Connected vertices may move apart
  - Connected vertices may converge

- We can improve the mesh by changing how vertices are connected

- Can also drive refinement/coarsening
Adaptive Mesh Reconnection
We work on a “hull” of cells that share a common edge and use dynamic programming to find the optimal connectivity in polynomial time using dynamic programming.
Parallelization

- Based on a halo approach and cell migration
Length-Scale Calculation

- Know when pack finer resolution near surfaces or coarsen away from surfaces

- Can also be used for gaps between surfaces
Fast length scale algorithm

- Create a field that indicates the desired mesh resolution
- Works by advancing away from boundaries with fixed length scales
- As we move away from boundaries, uninitialized cells receive the average of cells with calculated length scales
- Length scale can be scaled by a growth factor
Remapping

- When the mesh moves, no interpolation is necessary
- When the mesh is coarsened, conservative quantities are easily conserved
- Refinement, or other forms of reconnection, can pose a challenge
- Developed a general 2nd order conservative remapping scheme
Second-Order **Conservative** Remapping

- Based on Taylor series and intersection calculations

- Use the concept (only conceptual!) of a supermesh

- Also works for general polyhedra (decomposing to tets facilitates intersection calculations)
Convergence Results for a General Polyhedral Mesh

- Requires changes to how OpenFOAM calculates centroids
- Current method is not consistent with centroids of tet decomposition
GDI Engine: Moving Valves and Piston
Demo with Incompressible Flow and 6-DOF
Droplet Collision

- Modified version of interTrackFoam
Cost of Incompressible Calculation + Adaptation

- Cost is dominated (~85%) by MESQUITE/mesh motion
- Costs a little bit more than the pressure equation
Baseline case was with 4 processors, so the theoretical slope here is $1/4$. 
Pros and Cons

- **Pros:**
  - High level of automation
  - Can be parallelized
  - Mesh generation is easy (can even fix skewed meshes)

- **Cons:**
  - Complex code
  - Significant CPU cost
  - Simplical meshes are not as accurate as higher-order meshes
Current Status

- Serial code pushed to OpenFOAM-extend

- Parallel code still being tested and debugged at UMass—alpha version on-line

- Documentation in progress: Dissertation, 1 journal paper on the remapping, more papers in progress, plus Kyle Mooney’s write-up and tutorial session
Thank you

Sponsors:

• Army Research Office
• General Motors Research Center

Send praise and elementary questions to:
schmidt@ecs.umass.edu

Send criticism, bug reports, and difficult questions to:
deepersterblue@gmail.com

Usage and documentation questions:
kyle.g.mooney@gmail.com
Subversion Coordinates

- “Vanilla” version: serial and no conservative remapping is already in 1.6-ext

- The parallel version (w/ conservative remap) in a separate branch called feature/parallelDynamicTopoFvMesh

- The parallel version of Mesquite (w/ Hex/Prism cell support) is in a separate branch called feature/mesquiteHexPrismSupport