

Polyhedral Mesh Handling in OpenFOAM

Hrvoje Jasak

`h.jasak@wikki.co.uk`

Wikki Ltd, United Kingdom

FSB, University of Zagreb, Croatia

18/Nov/2005

Objective

- Overview of mesh handling in OpenFOAM

Topics

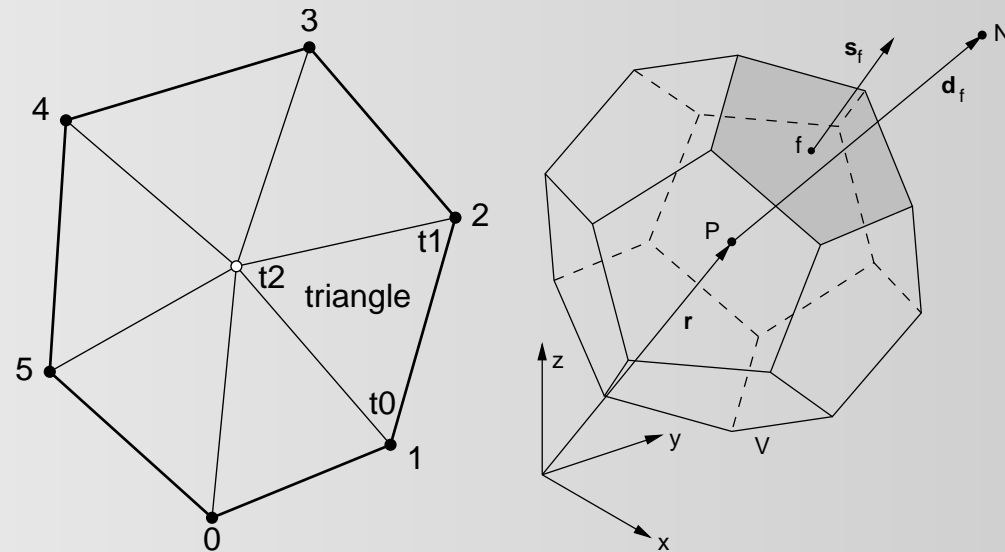
- Polyhedral cell and related nomenclature
- Mesh definition and organisation
- Zones and sets
- Moving mesh support
- Handling topological changes

Polyhedral Mesh Support

- To my knowledge, first CFD/CCM implementation capable of handling polyhedral cells throughout: working in 1995-1996
- Robust fully automatic mesh generator is a requirement in modern CFD
- Rationale
 - A polyhedron is a generic form covering all cell types: consistency in discretisation across the board
 - Finite Volume Method (FVM) naturally applies to polyhedral cells: shape (tet, hex, pyramid, prism) is irrelevant (unlike FEM)
 - Mesh generation is still a bottleneck: polyhedral support simplifies the problem
- Consequences
 - All algorithms must be fully unstructured. Structured mesh implementation possible where desired (e.g. aeroacoustics) but implies separate mesh classes and work on discretisation code re-use
 - In 1990s, fully unstructured support was a challenge: now resolved
 - No problems with imported mesh formats: polyhedral cell covers it all
 - Issues with “old-fashioned software” compatibility with no polyhedral support, e.g. post-processors. On-the-fly decomposition

Nomenclature

- A **polygonal face** is defined by a list of vertex labels. The ordering of vertex labels defines the face normal (orientation) using the right-hand rule
- A **polyhedral cell** is defined by a list of face labels that bound it
- Cell centre is marked by P , face centre and face interpolates by f , face normal s_f and neighbouring cell centre by N
- Face centre and cell volume are calculated using a decomposition into triangles or pyramids



Strong Ordering Requirement

- Polyhedral mesh definition
 - List of vertices. Vertex position in the list determines its label
 - List of faces, defined in terms of vertex labels
 - List of cells, defined in terms of face labels
 - List of boundary patches
- **All indices start from zero:** C-style numbering (no discussion please)
- OpenFOAM uniquely orders mesh faces for easier manipulation
 - All internal faces are first in the list, ordered by the cell index they belong to. Lower index defines **owner cell** (P); face normal points out of the owner cell
 - Faces of a single cell are ordered with increasing neighbour label, *i.e.* face between cell 5 and 7 comes before face between 5 and 12
 - Boundary faces are ordered in patch order. All face normals point outwards of the domain
- With the above ordering, patches are defined by their type, start face in the face list and number of faces
- Above ordering allows use of List slices for geometrical information

Checking Polyhedral Meshes

- Polyhedral definition introduces new kinds of errors
- Consistency checks
 - All points are used in at least one face
 - Boundary faces are used exactly one cell
 - Internal faces are used in at exactly two cells
- Topology checks
 - For each cell, every edge is used in exactly two faces
 - For complete boundary, every edge is used in exactly two faces
- Geometry checks
 - All volumes and all face areas magnitudes are positive
 - For each cell, sum of face area vectors is zero to machine tolerance. This takes into account the flipping of owner/neighbour face vectors are required
 - For complete boundary, sum of face area vectors is zero to machine tolerance
- Convexness condition: weak definition
 - In face decomposition, normals of all triangles point in the same direction
 - In cell decomposition, volumes of all tetrahedra is positive

Mesh Classes

- Base mesh classes: topology and geometry analysis engine
 - **polyMesh**: collection of all points, faces, cells and boundary patches in the simulation
 - **primitiveMesh**: active singly connected mesh as described above
- Discretisation types require specific support, re-using basic mesh info
 - **fvMesh**: supports the FVM discretisation
 - **tetFemMesh**: supports the FEM discretisation on tetrahedral decomposition
- Patch information separated along the same lines
 - PrimitivePatch: topology/geometry analysis
 - polyPatch: patch on the polyhedral mesh
 - fvPatch: support for the FVM
 - tetFemPatch: support for the FEM
- Patch mapping and access to base topology/geometry available for special cases

Operating on Subsets

- Zones and sets allow sub-setting of mesh elements
- Note: discretisation and matrix will always be associated with the complete mesh!
- Zones: points, faces and cells
 - Define partition of mesh elements. Each point/face/cell may belong to a maximum of one zone.
 - Fast two-directional query: what zone does this point belong to?
 - Used extensively in topological mesh changes
- Sets
 - Arbitrary grouping of points/faces/cells for manipulation
 - Single cell may belong to multiple sets
 - Sets used to create other sets: data manipulation with setSet tool
 - Examples

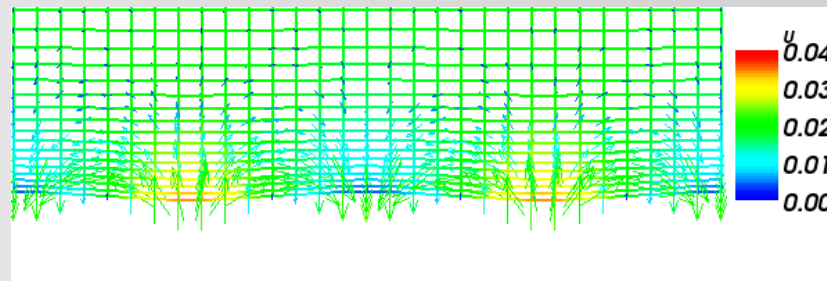
```
faceSet f0 new patchToFace movingWall
faceSet f0 add labelToFace (0 1 2)
pointSet p0 new faceToPoint f0 all
cellSet c0 new faceToCell f0 any
cellSet c0 add pointToCell p0 any
```

Moving Mesh Simulations

- Definition of a moving mesh problem: the number of points, faces and cells in the mesh and their connectivity remains the same but the point position changes
- Sufficient for most cases where shape of domain changes in time
- FVM naturally extends to moving meshes: need to re-calculate cell volume and area swept by a face in motion
- Moving mesh support built into mesh classes and discretisation operators
- In some places, algorithmic changes required in top-level solver code
- Problem: how to specify point motion for every point in every time-step?

Automatic Mesh Motion Solver

- Input for moving mesh cases is the point position field for every time step. This can be a pre-determined function of time (e.g. engine simulations) or a part of the solution (e.g. large deformation stress analysis, free surface tracking solver)
- Typically, only changes in boundary shape are of interest; internal points are moved to accommodate boundary motion and preserve mesh validity
- Automatic mesh motion solver
 - Internal point motion obtained by solving a “motion equation”
 - Prescribed boundary motion provides boundary conditions
 - Point-based (FEM) solver: no interpolation
 - Mesh validity criteria accounted for during discretisation
 - Easily handles solution-dependent mesh motion: solving one additional equation to obtain point positions



Topological Changes: Mesh Morphing

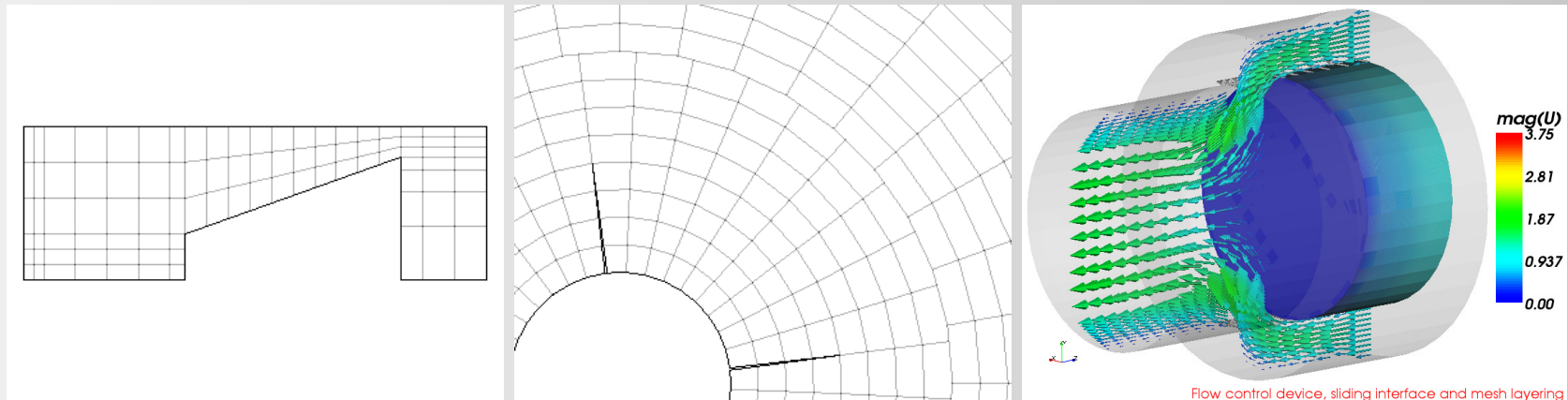
- Typically used with mesh motion; covers cases where the number of points/faces/cells and/or mesh connectivity changes during the simulation
- More complex and demanding than simple mesh motion
- Examples: attach-detach boundaries, cell layer addition/removal, sliding interfaces

Mesh Morphing Engine

1. Define primitive mesh operations: **add/modify/remove a point/face/cell**
2. Support primitive operations in polyMesh
3. Describe **mesh modifiers** in terms of primitive mesh operations
4. Create the topology instruction by setting up a combination of mesh modifiers on the original mesh
5. Collect changes from mesh modifiers into a **topology change request**
6. Execute topological change, including field mapping!

Mesh Morphing Engine

- Each mesh modifier is self-contained, including the triggering criterion
- Complex cases contain combinations of modifiers working together, mesh motion and multi-step topological changes
- Polyhedral mesh support makes topological changes easier to handle: solver is always presented with a valid mesh



- This is the current point of development
 - Intersecting mesh modifiers, parallelisation and bugs (sliding interface cutting algorithm), unified solver support (currently requires top-level code changes)
 - Issues with conservation of global properties after topological change. In some cases, this is algorithmic

Polyhedral Mesh Support

- OpenFOAM uses polyhedral mesh format. Mesh definition:
 - List of points
 - List of faces in terms of point labels
 - List of cells in terms of face labels
 - List of boundary patches
- Polyhedral mesh definition provides more freedom in mesh generation: bottleneck in modern CFD simulations
- Mesh analysis classes separated from discretisation support classes
- Built-in support for mesh motion and topological changes
- Simple handling of moving boundary problems: automatic mesh motion solver
- Topological changes support
 - Basic operations: add/modify/remove point/face/cell
 - Mesh modifier classes operate in terms of basic operations. Triggering of topological changes is automatic.
 - Pre-defined mesh modifiers available for standard operations
- Quest for a fully automatic polyhedral mesh generator continues!