Dynamic Mesh Handling in OpenFOAM

Hrvoje Jasak∗†

Director, Wikki Ltd, United Kingdom; Professor, University of Zagreb, Croatia

Extension of static mesh numerics in a CFD solver to cases with deforming boundaries considerably expands the scope of its use. Dynamic mesh handling includes deforming mesh cases, where the number and connectivity of mesh elements remains unchanged; and topological changes, where mesh size and connectivity varies during the simulation. Cases where the boundary deformation itself represents a part of the solution demand special attention: here, mesh handling needs to be fully automatic.

This paper describes dynamic mesh support in OpenFOAM, a C++ object-oriented library for numerical simulations in continuum mechanics. Unlike other tools, where dynamic mesh support is usually retro-fitted, object-oriented dynamic mesh engine has been built up from ground-up. Emphasis is given polyhedral cell support in mesh analysis and discretisation, vertex-based automatic mesh motion techniques and hierarchical design of topology morphing engine. The paper is completed with examples of solution-dependent motion with large boundary deformation.

Nomenclature

n  Unit normal vector
N  Neighbour cell centroid
P  Cell centroid
S  Surface
sφ  Volumetric source
u  Fluid velocity
us  Velocity of the boundary surface
V  Volume
α  Non-orthogonality angle
γφ  Diffusivity
Δmax  Cylinder displacement
φ  General tensorial property
ρ  Density

Subscript
f  Face, face value
N  Neighbour cell value
P  Cell value

I. Introduction

Examples of physical problems in aeronautical Computational Fluid Dynamics (CFD) which involve moving boundaries range from prescribed motion of wing actuators and similar flow control devices during take-off and landing to cases of missile release involving a 6 degrees-of-freedom (6-DOF) flow-dependent solid body motion and even more general cases of flow-induced boundary deformation. The latter also appear in contact stress analysis, fluid-structure interaction or other similar coupled problems. Beyond the basic
objective of supporting mesh motion in the flow solver, several objectives arise. Handling mesh deformation in a robust and reliable manner is of primary importance: without it, it is impossible to produce simulation results. For cases where domain shape is unknown a-priori, user intervention must be eliminated: a truly automatic solution-dependent motion is sought. Having in mind difficulties of problem setup, automatic dynamic mesh handling proves beneficial even for simple cases of prescribed motion.

One can recognise two fundamentally different dynamic mesh actions. Mesh deformation involves cases where boundary motion is accommodated simply by moving points that support the mesh, while in topological changes, the number or connectivity of points, faces and cells in the mesh changes within a time-step.

Discretisation support for deforming mesh cases may be considered complete: Arbitrary Lagrangian-Eulerian Finite Element Method (ALE-FEM) is a textbook subject, while moving mesh extensions to the Finite Volume (FV) discretisation have been present since the 1980s. In both cases, moving mesh support introduces no further discretisation error compared with static mesh techniques. In contrast, topological mesh changes, where mesh elements are added or removed between time-steps necessarily involve data mapping, with associated distribution and conservation errors. For this reason, deforming mesh techniques are easier to implement and numerically superior.

Some cases with significant boundary deformation make it impossible to accommodate the change using simple point motion: to preserve mesh quality and solution accuracy, faces or cells need to be added or removed from the mesh. Topological changes manipulate mesh resolution and connectivity in order to accommodate boundary deformation, typically using sliding interfaces, cell layering and similar techniques. In general, any local re-meshing or error-driven adaptive refinement may be considered as an example of topological change.

In this paper we shall address the subject of dynamic mesh handling and its implementation in OpenFOAM. In Section II, a moving mesh extension to the Finite Volume Method (FVM) on a polyhedral mesh is briefly summarised. Concentrating on the need for automatic mesh motion, a vertex-based solution technique using a mini-element Finite Element Method (FEM) with polyhedral cell support and variable diffusivity is presented in Section III. Layered implementation of topological change machinery in OpenFOAM is described in Section IV. The paper is completed with examples of dynamic mesh components working in unison on complex solution-dependent motion, Section V, and closed with a summary, Section VI.

II. FVM on a Moving Polyhedral Mesh

Moving mesh FVM is based on the integral form of the governing equation over an arbitrary moving volume \( V \) bounded by a closed surface \( S \). For a general tensorial property \( \phi \) it states:

\[
\frac{\partial}{\partial t} \int_{V} \rho \phi dV + \oint_{S} \rho \mathbf{n} \cdot (\mathbf{u} - \mathbf{u}_{s}) \phi dS - \oint_{S} \rho \gamma_{\phi} \mathbf{n} \nabla \phi \cdot dS = \int_{S} s_{\phi} dV, \tag{1}
\]

where \( \rho \) is the density, \( \mathbf{n} \) is the outward pointing unit normal vector on the boundary surface, \( \mathbf{u} \) is the fluid velocity, \( \mathbf{u}_{s} \) is the velocity of the boundary surface, \( \gamma_{\phi} \) is the diffusion coefficient and \( s_{\phi} \) the volume source/sink of \( \phi \). Relationship between the rate of change of the volume \( V \) and the velocity \( \mathbf{u}_{s} \) of the boundary surface \( S \) is defined by the space conservation law (SCL):

\[
\frac{\partial}{\partial t} \int_{V} \mathbf{V} - \oint_{S} \mathbf{n} \cdot \mathbf{u}_{s} dS = 0. \tag{2}
\]

Polyhedral FVM discretises the space by splitting it into convex polyhedra bounded by convex polygons. Temporal dimension is split into time-steps and equations are solved in a time-marching manner. Cell notation is shown in Fig. 1: a computational point \( P \) in cell centroid, a face \( f \), with area \( S_{f} \) and unit normal \( n_{f} \) with a neighbouring computational point \( N \).

Second-order discretisation of Eqn. (1) using a three time level scheme yields the following discretised form of Eqn. (1) for cell \( P \):

\[
\frac{3\rho_{P}^{n} \phi_{P}^{n} V_{P}^{n} \rho_{P}^{n+1} \phi_{P}^{n+1} V_{P}^{n+1}}{2\Delta t} + \sum_{f} \left( m_{f}^{n} - \rho_{f}^{n} V_{f} \right) \phi_{f}^{n} = \sum_{f} \left( \rho_{f}^{n} V_{f}^{n} \right) S_{f} n_{f} \cdot \left( \nabla \phi \right) + s_{\phi}^{n} V_{P}^{n}, \tag{3}
\]

where the subscript \( P \) represents the cell values, \( f \) the face values and superscripts \( n \) and \( o \) the "new" and "old" time level, \( \Delta t \) is the time step size, \( m_{f} = n_{f} \cdot \mathbf{u}_{f} S_{f} \) is the fluid mass flux and \( \dot{V}_{f} = n_{f} \cdot \mathbf{u}_{s} S_{f} \) is the

American Institute of Aeronautics and Astronautics
volumetric face flux. Cell volume $V_P^P$, $V_P^O$ and $V_O^O$ and volumetric face flux $V_f$ are calculated directly from geometric considerations and satisfy the discrete form of the SCL.$^2$

**III. Automatic Mesh Motion**

Objective of automatic mesh motion is to accommodate externally prescribed boundary deformation by changing positions of mesh points. During motion, the mesh must remain geometrically valid; this condition reduces to preservation of cell and face convexness and mesh non-orthogonality bounds.

**III.A. Problem Definition and Validity Constraints**

A general mesh deformation problem can be stated as follows. Let $D$ represent a domain configuration at a given time $t$ with its bounding surface $B$ and a valid computational mesh, Fig. 2. During a time interval $\Delta t$, $D$ changes shape into a new configuration $D'$. A mapping between $D$ and $D'$ is sought such that the mesh on $D$ forms a valid mesh on $D'$ with minimal distortion of control volumes.

Mesh validity constraints indicate that a domain could be considered as a linear elastic solid body under large deformation, governed by the Piola-Kirchoff stress-strain formulation. This is a non-linear equation and thus expensive to solve; in this study, a numerically cheaper Laplace equation with variable diffusivity is used instead.

A number of similar attempts has been reported in the past, ranging from cell-centred mesh motion solution (followed by cell-to-point interpolation), linear and non-linear spring analogy, torsional springs etc. In all cases, the critical component is control of discretisation error in the motion equation: it is this component that brings about deformation errors and algorithmic breakdown. A further complication is
related to polyhedral mesh support, implying that standard FEM discretisation may not be used: to the author’s knowledge, a second-order cell shape independent FEM has not been offered to date.

Calculation of polyhedral mesh metrics reveals a clue on control of discretisation error: for a polygon, face area is calculated by decomposition into triangles, while the cell volume calculation uses pyramids decomposed above the faces. Thus, if no face triangles and no cell pyramids are inverted during motion, initially valid mesh remains valid. Based on this idea, a mini-element approach is used. Here, each polyhedral cell is decomposed into tetrahedra on-the-fly, with the second-order shape function for a polyhedron assembled as a mini-element combination of second-order tetrahedral shape functions. Numerical properties of second-order tetrahedral shape functions controls inversion of individual tetrahedra by forcing the associated matrix coefficient to infinity.\(^3\)

A choice of mesh motion equation falls on the Laplace equation with variable diffusivity. Efficient diffusivity choices are based on distance from a moving boundary or measure cell distortion. For more details on variable organisation and motion-dependent diffusion the reader is referred to Jasak and Tuković.\(^3\)

III.B. Motion of a Cylinder

![Figure 3. Cylinder motion in 2-D: Initial polygonal mesh.](image)

The test case\(^3\) used to illustrate the performance of automatic mesh motion consists of a circle moving in a channel in 2-D\(^a\). Identical setup and a triangular mesh has been used by Baker\(^4\) with the pseudo-solid equation, and Helenbrook\(^5\) on the biharmonic equation.

Figure 3 shows the polygonal mesh used for the test, where \(D\) is the cylinder diameter, the height of the channel is \(2D\) and average mesh size is \(0.15D\). The polygonal mesh is generated using the algorithm proposed by Virag and Džijan.\(^6\)

The test consists determining the maximum displacement of the cylinder in one step without mesh inversion when the outside boundary remains fixed. Mesh quality is measured in terms of the non-orthogonality angle \(\alpha_f\) between \(\mathbf{d}_f\) and \(\mathbf{n}_f\), Fig. 1. Maximum single-step displacement \(\Delta_{\text{max}} = 1.39D\) is achieved using the diffusivity proportional to the distortion energy accumulated in motion.

IV. Topological Changes

In cases of extreme shape change, mesh motion alone is not sufficient to accommodate boundary deformation. Examples include a mixer vessel, where the internal part of the mesh rotates significantly past the stator, Fig. 5 and a case with two approaching boundaries, Fig. 6. For such cases, a mesh with fixed connectivity would quickly break down, unable to withstand additional twisting, or would introduce high discretisation error due to poor distribution of computational points.

In terms of software support, topological changes can be separated into clear functional levels. Our objective is to examine the levels and their interaction in a framework of object-oriented software like OpenFOAM. Components handling topological changes are separated into primitive mesh operations, topological modifiers and application-specific dynamic mesh objects.

IV.A. Primitive Mesh Changes

The lowest mesh manipulation layer specifies a topological change in terms of primitive operations: addition, removal or (connectivity) modification for a point, a face or a cell. Primitive mesh operations define a language for more complex mesh changes. A proposed set of nine mesh operations allows us to completely collapse an existing mesh or to build a mesh starting from empty space, thus proving generality of the interface.

\(^a\)In reality, the mesh is 3-D and consists of prismatic elements, as the software only operates on 3-D meshes.
(a) $\gamma = \text{const.}, \Delta_{\text{max}} = 0.65 D$, $\alpha_{f,\text{max}} = 83.4^\circ$, $\alpha_{f,\text{mean}} = 4.1^\circ$.

(b) $\gamma(l) = l^{-1}, \Delta_{\text{max}} = 1.1 D$, $\alpha_{f,\text{max}} = 80^\circ$, $\alpha_{f,\text{mean}} = 7.3^\circ$.

(c) $\gamma(l) = l^{-2}, \Delta_{\text{max}} = 0.95 D$, $\alpha_{f,\text{max}} = 82^\circ$, $\alpha_{f,\text{mean}} = 7^\circ$.

(d) $\gamma(l) = e^{-l}, \Delta_{\text{max}} = 0.94 D$, $\alpha_{f,\text{max}} = 85.1^\circ$, $\alpha_{f,\text{mean}} = 6.3^\circ$.

(e) $\gamma(U_d) = U_d, \Delta_{\text{max}} = 1.39 D$, $\alpha_{f,\text{max}} = 85^\circ$, $\alpha_{f,\text{mean}} = 10.4^\circ$.

(f) $\gamma(U_d) = U_d^2, \Delta_{\text{max}} = 0.67 D$, $\alpha_{f,\text{max}} = 68^\circ$, $\alpha_{f,\text{mean}} = 6.2^\circ$.

Figure 4. Deformed mesh for the limiting cylinder displacement with various non-constant diffusivity fields.

Figure 5. Mixer simulation: sliding interface in action.

Figure 6. Cell layering around a moving object.
The first functional level incorporates discretisation support, consisting of mesh and data renumbering. This functionality is built into the mesh object and is discretisation-independent.

Primitive mesh operations are sufficiently flexible, but impractical and tedious to use. For example, a single primitive operation may not lead to a valid mesh, e.g., removal of a single point. For this reason, primitive operations are executed in batches that make logical sense; complete mesh is rebuilt and checked for validity only when it is correctly re-assembled.

IV.B. Topology Modifiers

The second level of topological change machinery consists of higher-level objects called mesh modifiers. A mesh modifier holds a self-contained definition and a triggering mechanism for a topological change, executed in terms of primitive mesh operations. As an example, consider a layer addition/removal interface. When maximum layer thickness is achieved, a cell layer is added in front of a pre-defined mesh surface; when minimum thickness is breached, layer removal occurs.

Definition of a topology modifier relates only to a static mesh and come into action without user intervention. This is termed a “set-and-forget” strategy: a modifier present in a static mesh will be triggered automatically by mesh motion. OpenFOAM currently implements the following mesh modifier objects:

- Cell layer addition/removal, defined as a set of mesh faces which create an oriented surface, with minimum and maximum layer thickness, shown schematically in Fig. 7;

![Figure 7. Addition of an internal cell layer (yellow) during motion.](image)

- Attach-detach boundary, converting a set of internal faces into a boundary patch, thus attaching and detaching mesh components. Attach-detect action is triggered by either at times pre-defined by the user or in a solution-dependent manner;

- Sliding interface, defined as a pair of detached surfaces moving relative to each other, which will be attached in the overlapping region. Topological action removes the original interface faces and replaces them with facets to achieve one-to-one connectivity, Fig. 8;

![Figure 8. Coupling a sliding interface.](image)

Operation of a sliding interface is shown schematically in Fig. 9.

- Regular oct-tree refinement for hexahedral mesh regions.
More general mesh modifiers can also be defined: examples include error-driven adaptive mesh refinement or arbitrary crack propagation in structural analysis. Design of the topology engine allows simultaneous action of multiple non-interacting mesh modifiers. For cases where mesh modifiers interact or depend on each other, a further level of management is needed.

IV.C. Dynamic Mesh Objects

Mesh modifiers are considerably easier to use than primitive mesh changes but there exists room for further improvement, particularly when multiple modifiers are used in unison in a recognisable geometry-related manner, interacting with complex mesh motion.

To build on this, one may recognise typical cases of topological changes, using multiple mesh modifiers and prescribing motion in a user-friendly manner. Examples shown in Fig. 10 include a mixer vessel mesh object and a piston-and-valves configuration in internal combustion engine. For a mixer vessel mesh, motion is prescribed with axis of rotation and rotational speed in rpm, using a single sliding interface between a rotor and stator region (in red). It is clear how such mesh definition covers a range of topologically similar objects without reference to the geometry of a particular mixer.

A piston-and-valve configuration illustrates multiple interacting mesh modifiers: cell layering occurs at the piston as well as top and bottom valve surfaces (red and green), operating together with a sliding interface on a valve curtain (blue). During valve action, a sliding interface is decoupled first, followed by layering...
action and re-attachment of the sliding interface. For user convenience, motion is specified with reference to engine geometry, rotational speed and valve lift curves.

V. Results

Two example cases are given to illustrate dynamic mesh features in action. In the first, automatic mesh motion is used together to handle solution-dependent coupled motion-flow algorithm, while the second combines free surface flow CFD with 6-DOF solid body motion.

Consider a case\(^7\) of a free-rising bubble of air in a large water tank simulated using a surface tracking algorithm. Here, two fluids (inside and around the bubble) are handled by separate meshes and coupled on the free surface through boundary conditions. Surface motion is obtained as a part of the solution of the flow equation as a consequence of a double boundary condition.\(^7,8\)

Fig. 11 shows the mesh deformation and the pressure field around a 2-D air bubble of 1.5 mm diameter freely rising in water. After the initial transient, the bubble reaches terminal velocity and shape. The mesh in this simulation consists of 12,480 CVs in two disconnected regions and handles the interface using a coupled free surface boundary condition.

Here, mesh motion is dynamically re-calculated and bubble shape is a part of the solution. Variable diffusivity in the motion Laplacian is used to preserve boundary layers close to bubble surface. While bubble deformation is handled with ease, it is clear that mesh motion alone cannot accommodate bubble breakup, since this would involve a local and global solution dependent topology change.

![Figure 11. Free-rising air bubble in water in 2-D: pressure field and interface deformation with mesh quality indicators.](image)

The second example consists of a floating body in a free surface flow. Functionally, the implementation separates into two layers:

- A Volume-Of-Fluid (VOF) fluid flow solver\(^9\) formulated with support for mesh motion and topological changes. Note that a solver operates on a moving mesh without reference as to how mesh motion has been performed but only requires motion data as described in Section II;

- A floating body dynamic mesh class, which calculates forces on a given surface and solves the 6-DOF equation of motion for a solid body. Motion of the body acts as a boundary condition for the automatic mesh motion solver: as a result, the mesh deforms in response to forces given by the flow.
In this structure, a VOF solver is functionally independent from the way a mesh is moved and vice versa: a single mesh motion class can be used with multiple solvers. From the viewpoint of 6-DOF motion, it makes no difference whether the flow around a body is high-speed compressible (as in missile release) or incompressible free surface (as in naval hydrodynamics): this is simply a source of external forcing.

Fig. 12 shows 4 snapshots of the mesh in motion, where a barge is carried on an incoming wave using a 6-DOF motion solver in association with automatic mesh motion. The mesh quality is preserved even for large deformation, but motion alone does not allow for overturning.

In order to simulate the flow around an overturning body, computational mesh and 6-DOF motion will be decomposed into two components and assembled with the help of a sliding interface. Internal part of the mesh translates and rotates with the body, while the external part undergoes translation motion only. Connection between the two parts is handled with a sliding interface, compensating for rotation; in 3-D a sliding interface will be a sphere enveloping the body.

Fig. 12. 6-DOF motion in free surface flow: Floating body.

Fig. 13. 6-DOF motion in free surface flow: Overturning body.

Fig. 13 illustrates the free surface flow around an overturning body, given initial rotational velocity. Sliding between components can be clearly seen. As added benefit, internal part of the mesh moves as a...
solid body (attached to the barge) and undergoes no deformation. This is beneficial when fine near-wall mesh resolution is used: layered mesh structure next to the wall will remain intact. Note that the mesh does not match at the sliding interface, making it easy to increase mesh resolution close to the body without resorting to hanging node adaptation or propagating mesh refinement away from the region of interest.

VI. Conclusion

This paper describes the dynamic mesh capability for general polyhedral meshes implemented in OpenFOAM. Its main components are automatic mesh motion, topological modifiers and dynamic mesh classes.

Automatic mesh motion is deforms the mesh to respond to boundary motion prescribed in advance or in a solution-dependent manner. This is achieved by solving a motion equation formulated as a Laplacian with variable diffusivity, where boundary motion acts as a boundary condition. Polyhedral cell support is achieved using a vertex-based mini-element technique in the manner of a Finite Element Method.

For cases where simple mesh deformation will not suffice, topological mesh changes are used to vary the number and connectivity of primitive mesh components (points, faces, cells) during the simulation in a user-friendly manner. Topological changes are supported at the primitive level in a mesh object, including data mapping and renumbering. Two further levels combine primitive operations for ease of use: mesh modifiers and dynamic mesh classes. A mesh modifier handles a single self-contained operation, such as surface sliding or cell layering. At the top level, topology modification is married to mesh motion for specific classes of problems: examples include dynamic meshes for mixer vessels (motion + sliding interface) and solution-dependent motion in the case of a 6-DOF floating body.

Software architecture separates the implementation of flow solvers from dynamic mesh classes under generic interfaces, since mesh motion and flow solution are formally independent. Thus, a single dynamic mesh can be used with multiple flow solvers with no change.

A combination of an easy-to-use dynamic mesh engine and substantial physical modelling capabilities of OpenFOAM make a powerful tool, capable of handling complex physics in an environment where the domain shape varies in time- and solution-dependent manner.

Acknowledgments

Contribution of dr. Željko Tuković of the University of Zagreb, Croatia for the work on automatic mesh motion and example of moving bubble is gratefully acknowledged. Examples of complex topological action with piston-and-valves have been prepared in collaboration with drs. Tommaso Lucchini and Gianluca D’Errico of Politecnico di Milano, Italy.

References