V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

DYNAMIC MESH HANDLING IN OPENFOAM APPLIED TO FLUID-STRUCTURE INTERACTION SIMULATIONS

Hrvoje Jasak[*,†] **and Željko Tuković**[†]

[*]Wikki Ltd,
31 Dolben Court, Montaigne Close, London SW1P 4BB, United Kingdom
e-mail: h.jasak@wikki.co.uk
[†]Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb
Ivana Lučića 5, 10 000 Zagreb, Croatia
e-mail: {hrvoje.jasak,zeljko.tukovic}@fsb.hr

Key words: OpenFOAM, Dynamic Mesh, Finite Volume, CFD, Fluid-Structure

Abstract. *The power of OpenFOAM [1] in physical modelling stems from mimicking of partial differential equations in software, thus allowing rapid and reliable implementation of complex physical models. State-of-the art complex geometry handling and dynamic mesh features are essential for practical engineering use. This paper describes features of dynamic mesh support in OpenFOAM.*

Polyhedral mesh handling implemented in OpenFOAM is a flexible basis for its dynamic mesh features, at several levels. For simple cases of linear deformation or solid body motion, algebraic expressions suffice. For complex cases of time-varying geometry [2] or solution dependent motion, mesh deformation is obtained by solving a mesh motion equation. For extreme deformation, automatic motion is combined with topological changes, where the number or points, faces or cells in the mesh or its connectivity changes during the simulation. Finally, in cases of extreme and arbitrary mesh motion tetrahedral re-meshing based on edge swapping may be used.

Examples of dynamic mesh handling with progressively complex requirements shall be used to show how object-oriented programming simplifies the use of dynamic mesh features with various physics solvers.

A self-contained Fluid-Structure Interaction (FSI) solver in OpenFOAM acts as an illustration of dynamic meshing features and parallelised surface data transfer tools. It combines a second-order Finite Volume fluid flow solver with moving mesh support coupled to a large deformation formulation of the structural mechanics equations in an updated Lagrangian form [3].

1 INTRODUCTION

There exists a number of physical phenomena in continuum mechanics where the solution couples with additional equations influencing the shape of the domain on which the solution is being sought or a position of an internal interface. Examples of such cases include prescribed boundary motion in mixers, pumps and internal combustion engines; free-surface flows, where the interface between the phases is a part of the solution; fluid-structure interaction, where the deformation of a solid changes the shape of the fluid domain *etc.* Among several solution frameworks, a *deforming mesh* method is attractive for the clarity of formulation and accuracy it offers. Here, the points of the computational mesh is moved to follow the changing shape of the boundary. The main difficulty in this case is maintaining mesh validity and quality without user interaction.

In cases of extreme shape change, mesh motion alone is not sufficient to accommodate boundary deformation. Mesh topology, connectivity and resolution needs to be adapted, or the mesh needs to be locally regenerated. Each approach to the dynamic mesh problem carries its own advantages and drawbacks; when used n combination, they significantly enhance the power of numerical simulation software.

The power of object orientated software design in scientific computing stems from software organisation, where separate units – in our case, physics solvers and dynamic mesh handling – are separated and developed in isolation. Clear interfaces between the two allows the user to pick the appropriate physics solver and dynamic mesh handling technique without further coding. This also guarantees that the components tested in isolation will work without problem when used together.

In this paper we shall review the dynamic mesh handling techniques implemented in OpenFOAM by the authors and recent Open Source community contributions. This includes algebraic motion, Laplacian smoothing [4], Radial Basis Function (RBF) mesh deformation developed by Bos and co-workers [5] and field re-meshing techniques implemented by Menon *et al.* [6]. This is followed by a review of functionality and layout of the topological change machinery. At the top-level, physics solvers and dynamic mesh object provide a clear and simple interface, the structure of which will be reviewed below. The paper is completed with a set of relevant dynamic mesh examples, including Fluid-Structure interaction, and a brief summary.

2 DYNAMIC MESH HANDLING

The defining feature of a moving mesh simulation is temporal variation of the external shape of the domain. Thus, one can distinguish between *boundary motion* and *internal point motion*. Boundary motion can be considered as given, either prescribed by external factors or a part of the solution.

The role of internal point motion is to accommodate boundary motion and preserve the validity and quality of the mesh. It influences the solution only through mesh-induced discretisation errors [7] and is detached from the remainder of the problem, owing to the

Arbitrary Lagrangian-Eulerian (ALE) formulation of the conservation equations. Consequently, internal point motion can be specified in a number of ways, ideally without user interaction.

2.1 Moving Deforming Mesh

The mesh deformation problem can be stated as follows. Let $\mathcal{D}$ represent a domain configuration at a given time t with its bounding surface $\mathcal{B}$ and a valid computational mesh, figure 1. During a time interval Δt, $\mathcal{D}$ changes shape into a new configuration $\mathcal{D}'$. A mapping between $\mathcal{D}$ and $\mathcal{D}'$ is sought such that the mesh on $\mathcal{D}$ forms a valid mesh on $\mathcal{D}'$ with minimal distortion of control volumes.

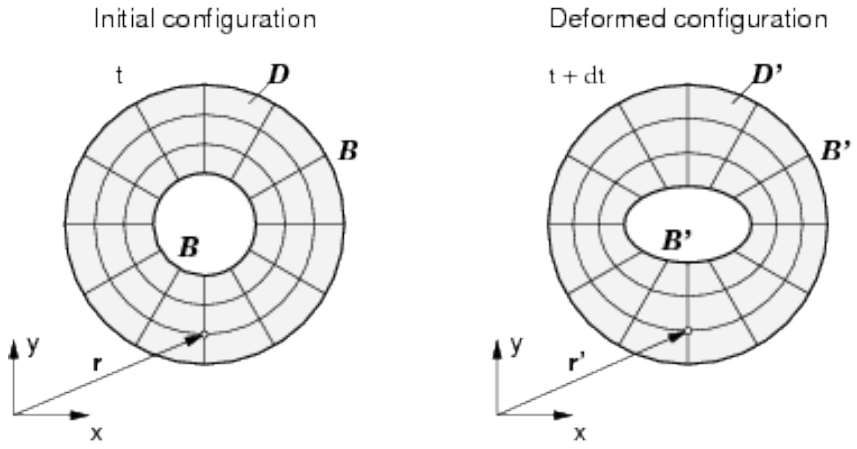

Figure 1: Mesh deformation problem.

2.2 Moving Mesh Discretisation Support: Finite Volume Method

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 ϕ it states:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho\phi \ \mathrm{d}V + \oint_S \rho \mathbf{n}\bullet(\mathbf{v}-\mathbf{v}_s)\phi \ \mathrm{d}S - \oint_S \rho\gamma_\phi \mathbf{n}\bullet\nabla\phi \ \mathrm{d}S = \int_V s_\phi \ \mathrm{d}V, \tag{1}$$

where ρ is the density, $\mathbf{n}$ is the outward pointing unit normal vector on the boundary surface, $\mathbf{v}$ is the fluid velocity, $\mathbf{v}_s$ is the velocity of the boundary surface, γ_ϕ is the diffusion coefficient and s_ϕ the volume source/sink of ϕ. Relationship between the rate of change of the volume V and the velocity $\mathbf{v}_s$ of the boundary surface S is defined by the *space conservation law* (SCL)[8]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathrm{d}V - \oint_S \mathbf{n}\bullet\mathbf{v}_s \ \mathrm{d}S = 0. \tag{2}$$

Polyhedral FVM discretises the space by splitting it into convex polyhedra bounded by convex polygons, [7]. Temporal dimension is split into time-steps and equations are solved in a time-marching manner. Cell notation is shown in figure 2: a computational point P in cell centroid, a face f, with area S_f and unit normal $\mathbf{n}_f$ with a neighbouring computational point N.

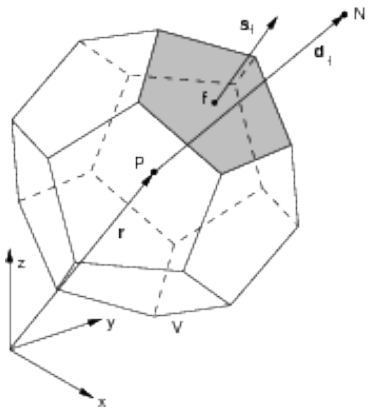

Figure 2: Polyhedral control volume (cell).

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

$$\frac{3\rho_P^n \phi_P^n V_P^n - 4\rho_P^o \phi_P^o V_P^o + \rho_P^{oo} \phi_P^{oo} V_P^{oo}}{2\Delta t} + \sum_f (\dot{m}_f^n - \rho_f^n \dot{V}_f^n) \phi_f^n = \sum_f (\rho\gamma_\phi)_f^n S_f^n \, \mathbf{n}_f^n \bullet (\nabla\phi)_f^n + s_\phi^n V_P^n, \tag{3}$$

where the subscript P represents the cell values, f the face values and superscripts n and o the "new" and "old" time level, Δt is the time step size, $\dot{m}_f = \mathbf{n}_f \bullet \mathbf{v}_f S_f$ is the fluid mass flux and $\dot{V}_f = \mathbf{n}_f \bullet \mathbf{v}_{sf} S_f$ is the volumetric face flux. Cell volume V_P^n, V_P^o and V_P^{oo} and volumetric face flux $\dot{V}_f$ are calculated directly from geometric considerations and satisfy the discrete form of the SCL[8].

2.3 Topological Mesh 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, figure 3 and a case with two approaching boundaries, figure 4. 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 discretisation support, standard topological change algorithms involve mesh-to-mesh mapping of data. This is typically associated with mapping errors, either in the sense of non-smooth local field values or in a loss of global conservation. A

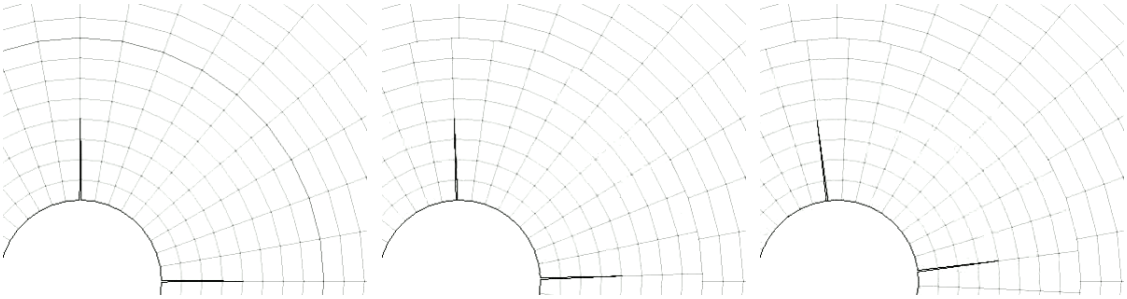

Figure 3: Mixer simulation: sliding interface in action.

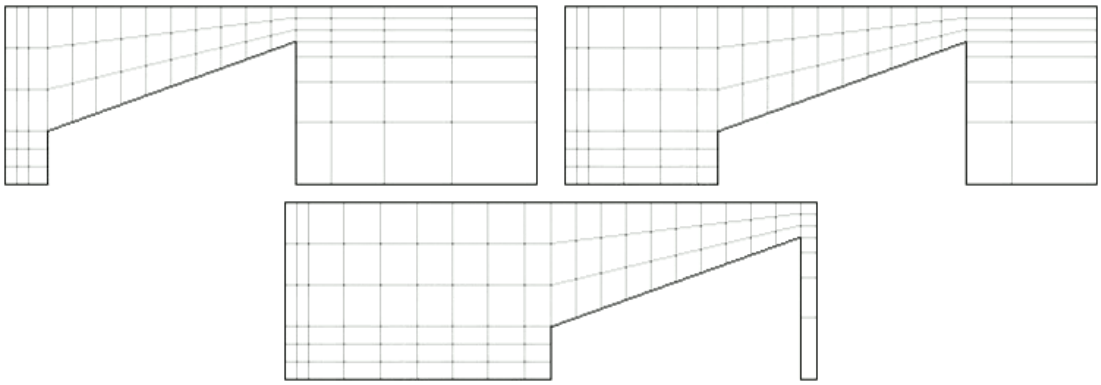

Figure 4: Cell layering around a moving object.

simple – and inaccurate – cure for this is global scaling. However, it quickly becomes clear that the moving mesh FVM provides an alternative route, removing the mapping errors. If a topological change occurs in conjunction with mesh motion, it is possible to first collapse the cells and faces to zero volume and area using standard mesh motion techniques and then remove them from the mesh. In case of cell or face addition, new elements are introduced with zero metrics and then inflated through mesh motion.

The desired side-effect of no mapping is thus achieved: a FVM solution for a zero-volume cell is indeterminate and does not affect the surrounding field. In practice, the "mapping step" is performed implicitly by the mesh motion algorithm.

3 MESH DEFORMATION SOLVER

Having resolved the issues of discretisation support and data mapping, it remains to determine the motion of mesh points in response to prescribed boundary motion, ideally in an automated manner.

The overriding criterion for the success of automatic mesh motion is mesh validity: an initially valid mesh must remain valid after deformation. In terms of FVM metrics, [4], this condition reduces to positivity of cell volumes and face areas, preservation of cell and face convexness and mesh non-orthogonality bounds. In layman's terms, the cells and

faces in the mesh should not be "flipped" while the mesh is in motion.

At a higher level, mesh motion validity criterion may be written in terms of a motion function, specifying the vertex deformation in a form consistent with the continuum representation. Here, we shall assume that the function is continuous in space (as opposed to existing only on mesh points). On this basis, the motion validity condition can be seen in terms of motion function smoothness in space: a monotonically smooth and regular motion function will not cause cell or face flipping.

Depending on complexity of boundary motion, mesh deformation cases may be handled either by simple algebraic expression or by more complex functional forms, as shown below.

Algebraic Mesh Motion. In algebraic mesh motion, position and velocity of points is calculated directly from a globally known motion laws. Good examples involve solid body motion or linear deformation of the mesh within its bounding box. Based on the skill of the user, this may extend to quite complex cases of multiple bodies regularly oscillating in the flow field. The motion technique is exceptionally efficient and accurate but somewhat limited in scope: it is typically defined for a small subset of geometries. Examples include prescribed solid body motion, regularly oscillating geometries, eg. liquid reservoirs in sloshing simulations and arbitrarily complex linear deformation.

Laplacian and Pseudo-Solid Smoothing. For cases where boundary motion is irregular or solution-dependent, algebraic mesh motion is not sufficiently flexible. An alternative way of looking at the mesh motion problem is to consider prescribed boundary motion as a "boundary condition" on an unknown "mesh motion equation". It follows that internal point motion may be determined by solving the motion equation, [4].

Mesh validity constraints indicate that a domain could be considered as a solid body under large deformation, governed by the Piola-Kirchoff stress-strain formulation. This is a non-linear equation and thus expensive to solve; as stresses are of no interest, a similar and numerically cheaper approach along the same lines is sought. Two obvious choices are the pseudo-solid equation [9] and the Laplace equation [10].

When the Laplace equation governs mesh motion, the prescribed boundary deformation is not uniformly distributed through the domain. The nature of the equation is such that point movement is largest adjacent to the moving boundary, potentially leading to local deterioration in mesh quality. Ideally, largest deformation should be confined to the internal part of the mesh, where it causes less distortion. This can be achieved by prescribing variable diffusivity in the Laplacian, as explored in [4].

Motion using Radial Basis Functions. In his work, Bos [5] recognises the fact that smooth interpolation criteria may be formulated in purely algebraic terms rather than coded into a form of a partial differential equation. Such a formulation would lead to a faster and more robust mesh motion technique.

Radial Basic Function (RBF) interpolation uses a small number of data-carrying points to create a global and smooth interpolation of available data in space. The smoothness criteria is encoded as a condition on positive interpolation coefficients and global stencil support. This is achieved by solving a system of linear equations for interpolation parameters: this is the most expensive operation in the assembly of RBF interpolation.

The RBF interpolation formula [5] is defined as:

$$s(\mathbf{x}) = \sum_{j=1}^{N_b} \gamma_j \phi(|\mathbf{x} - \mathbf{x}_{b,j}|) + q(\mathbf{x}), \tag{4}$$

where $\mathbf{x}$ is the interpolant location, $\mathbf{x}_b$ is the set of N_b locations carrying the data, $\phi(x)$ is the basis function, dependent on point distance between the target point and data carriers and $q(\mathbf{x})$ is the (usually linear) polynomial function, depending on choice of basis function and γ_j. Consistency of interpolation is achieved by requiring that all polynomials of the order lower than q disappear at data points:

$$\sum_{j=1}^{N_b} \gamma_j p(\mathbf{x}_{b,j}) = 0. \tag{5}$$

Upon choosing the basis function, coefficients of q,

$$q = b_0 + b_1 x + b_2 y + b_3 z \tag{6}$$

$b_0 - b_3$ and γ_j are determined by solving the system:

$$\begin{bmatrix} s(\mathbf{x}_{b,j}) \\ 0 \end{bmatrix} = \begin{bmatrix} \Phi_{bb} & Q_b \\ Q_b^T & 0 \end{bmatrix} \begin{bmatrix} \gamma \\ \beta \end{bmatrix}, \tag{7}$$

where $s(\mathbf{x}_{b,j})$ is the function value at interpolant locations (source data), γ carries all γ_i coefficients and β carries $b_0 - b_3$ and Φ_{bb} carries the evaluation of the basis function for pairs of interpolation points $(\mathbf{x}_{b,i}, \mathbf{x}_{b,j})$ and acts as a dense connectivity matrix:

$$\Phi_{(i,j)} = s(|\mathbf{x}_{b,i} - \mathbf{x}_{b,j}|). \tag{8}$$

Q_b is the rectangular matrix with $[1\ \mathbf{x}_b]$ in each row.

The system is a dense matrix and needs to be solved from γ and β using QR decomposition (direct solver), thus defining the interpolation.

The choice of radial basis functions is described in detail in [5] and can have two forms. Basis functions with local support disappear beyond the radius r and are typically polynomial. This eliminates some entries in Φ_{bb}, making the system easier to solve. In contrast, basis functions with global support cover the whole interpolation space, and usually require a smoothing function to make the system in Eqn. 7 easier to solve.

In terms of mesh motion, RBF interpolation shall be established [5] on a small set of control points on the moving surface, whose motion shall be used as "known motion data". The RBF formula, Eqn. 4, is then used to calculate the motion of all other mesh points.

RBF-based mesh motion has proven extremely efficient and robust. It works especially well in cases where the number of control points may be small: it is the number of control points that defines the size of dense matrix for inversion in Eqn. 7 and with it the bulk of interpolation cost. Examples of RBF-based mesh motion will be shown below and originate from the work of Bos [5].

4 IMPLEMENTATION OF TOPOLOGICAL MESH CHANGES

Topological changes in OpenFOAM are implemented in an object-oriented and hierarchical manner. Primitive mesh actions can add, modify or remove a single point, face or a cell in the mesh. For ease of interaction, topological modifiers such as cell layer addition/removal, sliding interfaces or attach/detach boundaries form the second level of topological machinery. Each mesh modifier is a self-contained unit, including an automatic triggering criterion, such as min/max cell layer thickness for layer addition/removal.

4.1 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.

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. Discretisation-specific mesh derived classes such as `fvMesh` for the FVM are responsible for corresponding field data mapping; they also collect discretisation-specific mesh motion data in a form suitable for use with discretisation. In the case of moving mesh FVM, this included mesh motion fluxes, appearing in Eqn. 3.

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, eg. 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.

4.2 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;
- Attach-detach boundary, converting a set of internal faces into a boundary patch, thus attaching and detaching mesh components. Attach-detect action is triggered 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. Uncovered faces remain grouped in boundary patches, ready to support boundary-type discretisation;
- Dynamic crack propagation in non-linear structural analysis, where a crack damage model is used to indicate which internal faces of the mesh should be converted into boundary faces. In this way, crack propagation and mesh motion is used to capture the real dynamic shape of the cracking geometry and the associated stress state;
- Regular octree mesh refinement for hexahedral mesh regions.

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.

4.3 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. Calculation of dynamic boundary motion and triggering of complex topological changes is encapsulated in the dynamic mesh object itself: as a result, its interface to the remainder of the code can be a simple update function.

There obviously exists a large variety of dynamic mesh objects: their main role is to facilitate case setup and user interaction. In most cases, combinations of user-friendly

dynamic mesh motion and specific topology modifiers are used to form mesh templates for a class of motion cases. Examples of dynamic mesh objects in action shell be presented below.

The ultimate flexibility of a dynamic mesh object is the one where boundary motion may be prescribed in an arbitrary manner and topological changes are triggered simply by mesh quality constraints. Menon *et al.* [6] implement a tetrahedral re-meshing class which combines mesh motion based on Laplacian smoothing and a cell quality indicator. When a cell or a cluster of cells is considered too distorted for further use, a local re-meshing step is introduced. Here, a cluster of cells in the problematic region is analysed and improved in quality through triangular/tetrahedral edge swapping, cell splitting or merging. Such clusters are typically isolated and local re-meshing occurs only once in several time-steps of a dynamic mesh run and the computational effort is limited. The result is a highly efficient and flexible dynamic mesh algorithm operating without user intervention. A combination of automatic detection of distorted cells and local re-meshing leaves an impression of a smoothly-changing mesh in motion. Curiously, the dynamic tetrahedral re-meshing answers to the interface of a dynamic mesh class and requires no further interaction in the code.

4.4 Interfacing with the Physics Solver

A dynamic mesh object, `dynamicFvMesh`, is a derived form of an FV mesh class (`fvMesh`, which supports the FVM discretisation and allows for the possibility of changing during the run. The simplest example would be a `staticFvMesh`, which remains unchanged during the run: its `update()` function does no work. More complex meshes may operate in their own specific way; in complex cases like 6-Degree-of-Freedom (6-DOF) object motion, this may involve a calculation of forces from the volumetric flow and solution of an Ordinary Differential Equation (ODE), or dynamic re-meshing calls in mesh refinement or tetrahedral edge swapping. Solution information needed on the mesh side (eg. pressure, velocity and turbulence fields) may be extracted from the solver database, without intervention in the top-level code.

The impact of dynamic mesh actions in a physics solver code may be handled in a generic manner. The FVM physics solver operates on a field level and is intrinsically independent of the mesh (discretisation of space). Provided that all mesh-to-mesh mapping actions happen behind the scenes, the physics solver simply needs to be equipped with moving mesh terms shown in Eqn. 3. Therefore, the solver-level interventions related to dynamic mesh changes boil down to the following:

- Create a mesh object that conforms to the dynamic mesh interface and performs the necessary mesh motion and data mapping information. In OpenFOAM, this is done using run-time selection tables, without exposing the source code of a derived dynamic mesh class to the physics solver at compile-time;

- Write the physics equations in a form that supports moving deforming mesh dis-

cretisation, as shown in Eqn. 3;

- Call the `mesh.update()` function at the appropriate place in the solver.

5 DEFORMING MESH SIMULATIONS

In what follows, we shall present the examples of various dynamic mesh objects in action, each representing a class of motion problems.

5.1 Prescribed Solid Body Motion

Among dynamic mesh cases, prescribed solid body motion is the easiest to deal with: the complete domain is moving with uniform displacement for each time step. Coupled with a nonlinear flow model, even such simple motion produces interesting results.

Figure 5 shows a snapshot of a laminar free surface flow in a swirling container. The motion is specified as a superposition of multiple sinusoidal loops, aimed at improving the mixing in the system. Free surface is coloured by fluid velocity.

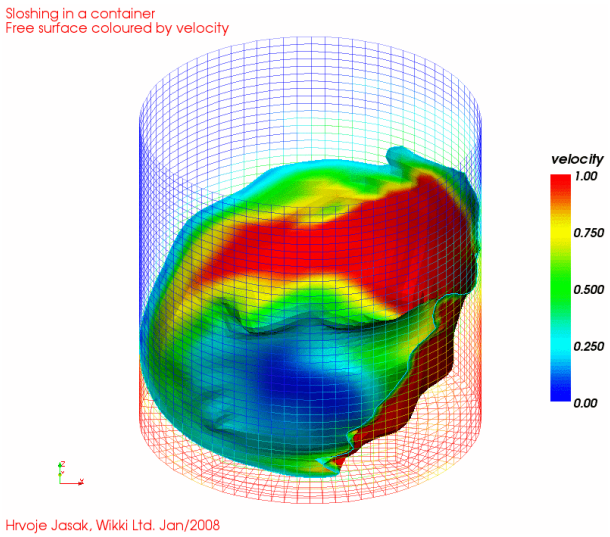

Figure 5: Solid body motion with free surface flows: swirling flow in a moving container.

Similar cases of solid body motion regularly appear in naval hydrodynamics CFD, specifically in sloshing and slamming simulations. In many cases, the effect of moving mesh on the flow field may be reformulated in terms of volumetric body force, calculated as a derivative of motion velocity, either analytically or from user-prescribed motion data. Experience shows that a moving mesh simulation runs are more robust and accurate, at a price of a low computational overhead.

5.2 Mixers and Turbomachinery

A `mixerFvMesh` is a perfect example of user-friendly definition of mesh motion and topological changes. The mesh consists of a static part and rotational part, separated by a sliding interface. Motion of the rotor is simple to define: constant rotational speed around a prescribed axis. It is surprising to see how many cases comply to this definition: from mixers and pumps, various types of turbomachinery to propellers and shrouded propulsors.

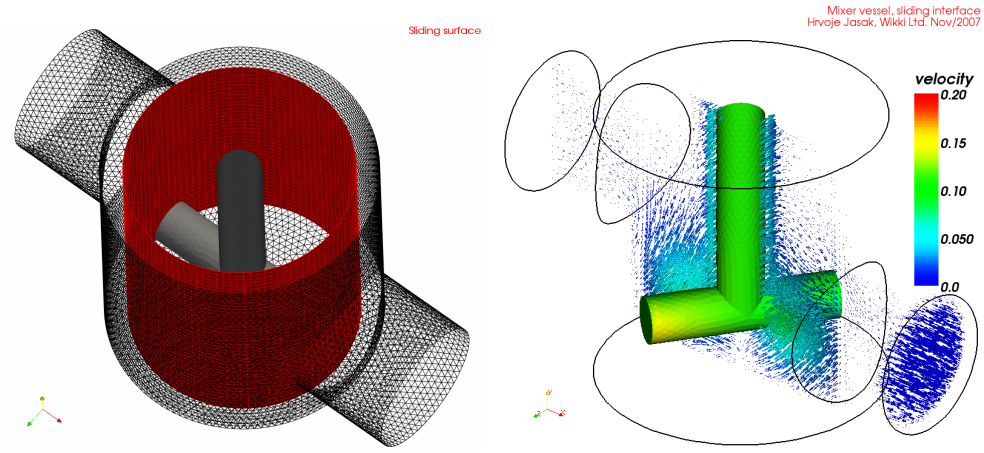

Figure 6: Mixer vessel simulations using a sliding interface technique. Left, sliding surfaces; right, flow solution.

Figure 6 shows a simplified mixer geometry used in material processing, with the sliding surface coloured in red. The sliding interface between two disconnected regions allows the user to build the rotor and stator mesh components separately, thus simplifying parametric studies. Attach/detach action of a sliding interface is executed every time-step, with the rotational point motion prescribed analytically.

From the point of view of the flow solver, no modifications are required. The mesh is presented to the solver as a singly-connected component with a "perfectly matched" interface: polyhedral cell definition is perfect for the action of a sliding interface. In terms of motion, the mesh motion fluxes are present in faces for the rotational component and equal to zero in the stator.

5.3 Naval Hydrodynamics: Floating Object with 6-DOF Force Balance

Simulation of floating objects in the flow formally involves solid body motion, but is considerably more complex than cases above. Firstly, motion of the object is unknown and a part of the solution: it involves a solution of the motion equation, with forces acting on the body calculated from the flow field. Access to the pressure, velocity and turbulence fields is done using database access and forces are calculated within a helper object: this will allow us to couple the same dynamic mesh object to various types of volumetric

flow solvers. Examples would include a manoeuvring submarine (incompressible flow), aircraft (compressible transonic flow) or a floating object (volume-of-fluid free surface flow). Secondly, while the object moves as a solid body, the flow domain around it does not: prescribed surface motion needs to be accommodated by mesh deformation.

The main component of the `sixDofMotion` dynamic mesh class is a list of `floatingBody` objects, which store the patch identification for each floating object, its inertial properties (coordinates of centroid, mass, moment of inertia) and motion ODE. Optionally, it is possible to add the propulsion force acting off centroid, like a sail force in sailing yacht simulations. The `sixDofMotion` class collects the boundary motion from all floating object and uses an automatic mesh motion solver to deform the global mesh.

Figure 7 left, shows a snapshot of a free surface flow around two simplified barges, with the wake from the leading barge influencing the motion of the trailing.

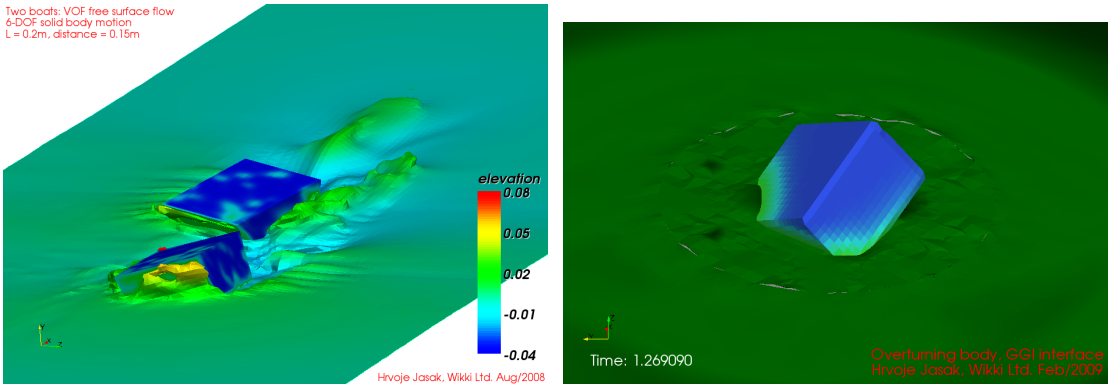

Figure 7: 6-DOF floating body simulations.

For cases of overturning bodies, simple mesh motion will not suffice: substantial rotation would destroy the mesh. To deal with this, a `floatingBody` class optionally supports a two-part mesh, where the internal part is attached to the floating body and moves in unison with it. The external part of the mesh captures translational motion only; between the two, a sliding or GGI interface similar to the one in mixers is used to accommodate relative rotation between components. An example of this kind is shown in Figure 7, right. An interesting side-effect of this mesh setup is that the mesh close to the body remains undisturbed and near-wall mesh layers are protected from deformation.

5.4 RBF Mesh Motion

The work of Bos [5] concentrates on simulation of flapping motion of insect wings in flight. Early studies have quickly shown that Laplacian and pseudo-solid mesh motion is insufficiently robust to handle the motion of this amplitude and rotation. RBF motion has been implemented as a more robust alternative, using the interpolation machinery described above.

Formally, all points on the moving boundary and stationary far-field points should be used as data carriers: their large number would make RBF interpolation impractical and expensive. Two improvements are introduced:

- In cases of interest, the surface of the wing (or other moving object) moves in a regular manner, nearly as a rigid object. It is therefore possible to coarsen the set of data carriers by skipping points on the moving surface without loss of accuracy in volumetric motion. Surface points will be moved according to user prescription to ensure accuracy;
- Stationary points are typically located in far field. It is therefore practical to remove them from interpolation support and replace their data with a smoothing function, which extinguishes the motion in far field.

Combination of the above significantly improves the performance of RBF motion, without significant loss of accuracy.

Figure 8 shows RBF mesh motion in action on a case of translating and rotating box (red) with a stationary far field. In comparison with Laplacian smoothing, RBF motion meshes show high quality in motion [5]. A careful look at three positions shows how the smoothing function modifies the motion in the far field.

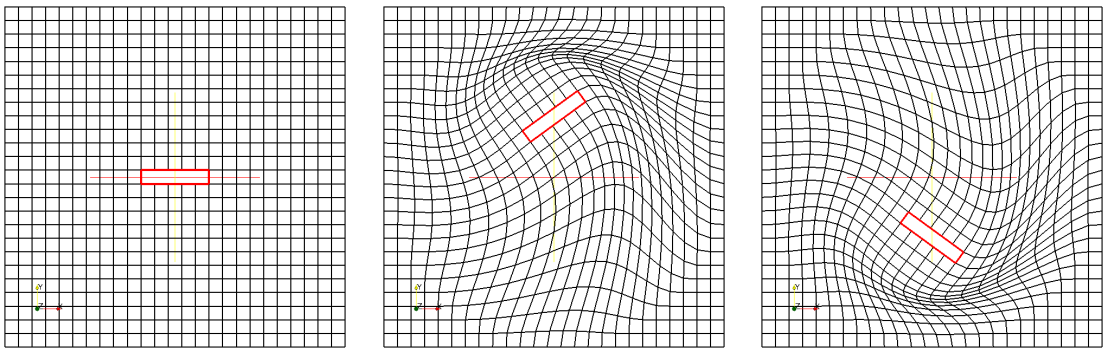

Figure 8: Radial Basis Function mesh motion, reproduced from Bos [5].

5.5 Re-meshing with Tetrahedral Edge Swapping

Menon *et al.* [6] performed a detailed numerical study of viscoelastic droplet collision. Free surface is presented as a moving and deforming mesh interface or outer boundary, undergoing topological change on impact and breakup. In collision, illustrated in Figure 9, two mesh components merge into one and potentially separate into several parts. This is the ultimate challenge for dynamic mesh handling: surface motion, surface breakup and number of droplets post impact are a part of the solution. Tetrahedral edge swapping algorithm implemented by Menon is perfect for this kind of study: motion and topological changes are completely automatic.

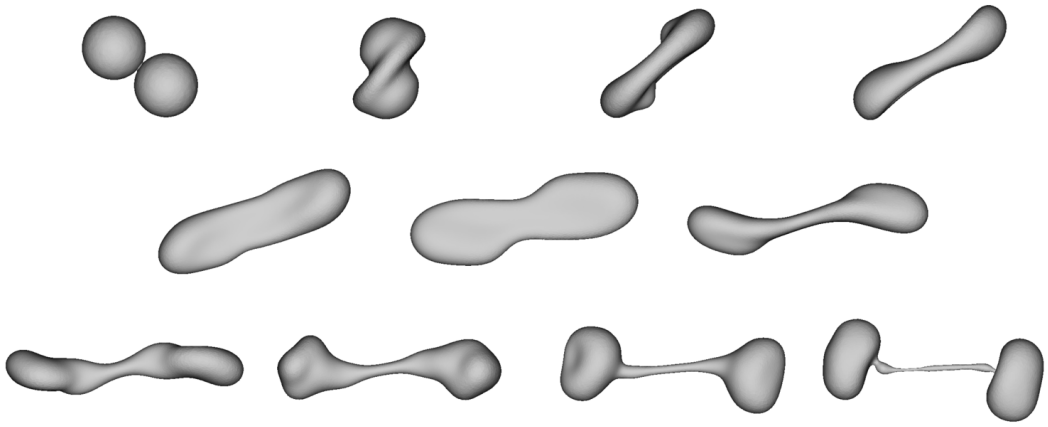

Figure 9: Surface tracking with tetrahedral edge swapping: collision of viscoelastic droplets, reproduced from the work of Menon *et al.*

Interestingly, tetrahedral edge swapping complies to the interface of a `dynamicFvMesh` and is attached to a fluid flow solver without intervention. The dynamic mesh algorithm is quite general and can be used in other simulations without change. Figure 10 shows the mesh action for a flow simulation in an internal combustion engine with moving piston and valves. Traditionally, this class of problems is handled by hand-built meshes, sliding interfaces and layer addition/removal action and involve substantial effort in setting up the case. Tetrahedral edge swapping provides a completely automatic alternative.

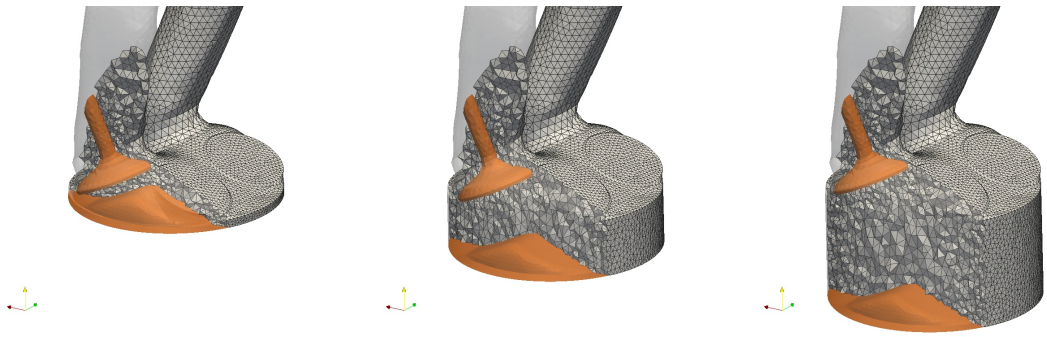

Figure 10: Tetrahedral edge swapping technique for in-cylinder flows, reproduced from the work of Menon *et al.*

5.6 Fluid-Structure Interaction

OpenFOAM is ideally suited for coupled multi-physics simulations, for two reasons. In many cases, physics solvers to be used in a coupled manner are already available and accessible in full source in the library. Using them together is therefore a relatively simple task, involving data exchange and coupling algorithms within a single executable. If one

of physics solvers is not available, coupling OpenFOAM to external tools is again straightforward, due to full access to the source code. For example, fluid-structure coupling for sail simulations in racing yachts uses a specialised structural analysis solver with a thin plate formulation and features specific for sail design (stiffeners, rope and pulley systems and similar). Coupling the fluid and mesh motion capabilities of OpenFOAM with such an external solver is achieved through force and motion boundary conditions, exchanging information between the codes and using surface-to-surface data mapping tools already available in OpenFOAM.

Surface Mapping Tools. The basic functionality of data mapping between two surfaces has been available in OpenFOAM for a number of years. It combines fast surface search algorithms and inverse-distance weighting, with escapes for cases of "direct hit" where the data is transferred without interpolation (for cases of matching meshes on the interface). Recent development of advanced data interpolation tools, be it the General Grid Interface (GGI) interpolation [11] or RBF interpolation [5] provides further options. GGI interpolation is of particular interest: it simultaneously provides smooth and force/flux-conservative mapping for face-based data.

For cases where either the structural or the fluid solver runs on a massively parallel computer in a domain decomposition mode, the task of data mapping needs to account for the distributed layout of two meshes. The task is of considerable complexity, as the domain decomposition pattern is not know in advance. In this case, parallelisation of surface mapping tools is built directly into the interpolator. Each processor holds its own part of the surface data (governed by the parallel decomposition of the volumetric mesh) and the surface-to-surface mapping step is preceded by processor communications. In this way, domain decomposition used by two solvers is independent of each other, albeit at the price of additional parallel communications.

Level of Coupling. In terms of level of coupling, open source implementation provides substantial flexibility. The most commonly used algorithm uses Picard iterations within each time-step, where the fluid and structure solver are used consecutively and the coupling terms are explicit. Fixed or adaptive (Aitken) relaxation may be used to achieve closed coupling between the solutions.

For strongly coupled FSI problems, further avenues are available. Discretisation machinery in the structural and fluid flow solver are built on a common support of mesh, matrix and linear algebra support. It is therefore possible to achieve component coupling at matrix level by combining the solution of the discretised fluid and structural model. To achieve this, surface-to-surface data mapping needed for force and displacement transfer should also be presented in matrix form. Substantial cost increase is involved: fluid flow, mesh motion and structural analysis would need to be solved in a single linear solver call, with appropriate linearisation.

Similar effect may be achieved at a lower price using an Arnoldi-type algorithm directly in the top-level code. Here, separate solution for the flow and stress analysis is used as a preconditioning step and the main convergence loop within a time-step is replaced by a matrix-free non-linear Arnoldi solver. This is a subject of current research.

Numerical Example. In this example, a combination of the laminar fluid flow solver and a non-linear structural analysis solver in the updated Lagrangian formulation [3] is used. Flexible structure undergoes a substantial deformation, which in turn considerably impacts the fluid flow.

On the solid side, the mesh deformation is already available as a part of the solution (in updated Lagrangian formulation), while the fluid solver uses a Laplace-based automatic mesh motion solver.

Figure 11 shows snapshots of the fluid flow and structural deformation in a case experimentally studied by Hron and Turek. Substantial deformation of the structure is clear to see, as is the reason for using a large deformation formulation in the stress analysis solver. Parallel data mapping tools allow us to decompose the fluid and solid domain independently: in most cases, the fluid mesh carries substantially more computational effort and independent decomposition of components is helpful in achieving load balance.

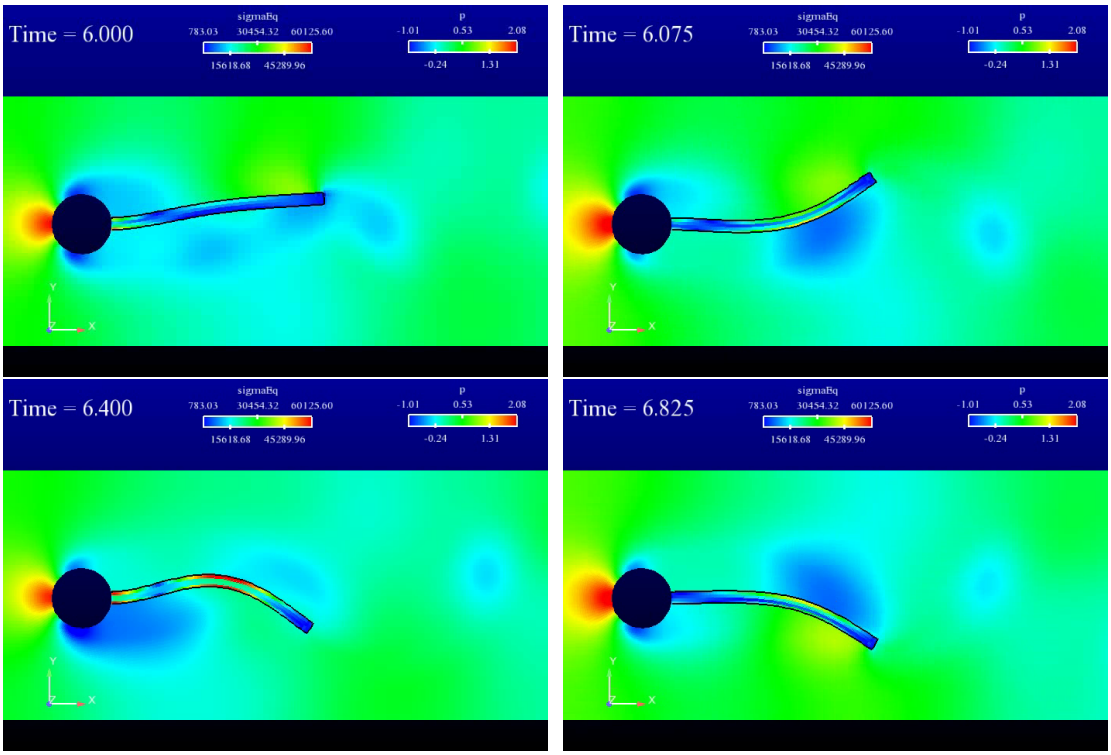

Figure 11: Fluid-structure simulation with large deformations in the solid.

6 SUMMARY

This paper describes features of dynamic mesh handling and recent contributions implemented in OpenFOAM. At the top-level, flow solvers are equipped to deal with mesh deformation during the run, by inclusion of mesh motion terms. Interfacing with the dynamic mesh classes is straightforward: all that is required is a mesh update call in the time loop. On the other side, dynamic mesh classes are independent from top-level solver physics, can be implemented in isolation and re-used without change.

Two new dynamic mesh techniques have been presented. RBF mesh motion, implemented by Bos show performance superior to Laplacian smoothing, especially for cases of large deformation. For cases where simple mesh motion is insufficient, fully automatic tetrahedral edge swapping technique implemented by Menon may be used. While it is limited to tetrahedral meshes, it is efficient and robust, making it an ideal choice for cases involving topological change of external boundary.

Fluid-structure interaction case, combining laminar fluid flow and large deformation of a solid is used as a demonstration of solver-to-solver coupling, automatic mesh motion and data mapping tools, all of which are implemented in OpenFOAM available in open source.

7 ACKNOWLEDGEMENT

Authors would like to thank dr. Frank Bos and Mr. Sandeep Menon and their research colleagues at TU Delft and UMass Amherst for kind permission to present their work. We are particularly grateful for making the code they developed in OpenFOAM available to the Open Source Community.

REFERENCES

[1] Weller, H.G. Tabor, G. Jasak, H. and Fureby, C., A tensorial approach to computational continuum mechanics using object orientated techniques, *Computers in Physics*, **12**, pp. 620-63, (1998)

[2] Jasak, H, Dynamic Mesh Handling in OpenFOAM, *48th AIAA Aerospace Sciences Meeting, Orlando, Florida*, (2009)

[3] Tuković, Ž. and Jasak, H., Updated Lagrangian Finite Volume Solver for Large Deformation Dynamic Response of Elastic Body, *Transactions of FAMENA*, **31**, pp. 55-70 (2007)

[4] Jasak, H. and Tuković, Ž., Automatic mesh motion for unstructured finite volume method, *Transactions of FAMENA*, **30**, pp. 1-18 (2007)

[5] Frank Bos, Numerical simulation of flapping foil and wind aerodynamics: Mesh deformation using radial basis functions, PhD Thesis, Technical University Delft (2009)

[6] Mooney, K., Menon, S. and Schmidt, D., A Computational Study of Viscoelastic Droplet Collisions, *22nd Annual Conference on Liquid Atomization and Spray Systems*, ILASS-Americas, Cincinnati, OH, USA (2010)

[7] Jasak, H., Error analysis and estimation in the Finite Volume method with applications to fluid flows, PhD Thesis, Imperial College, University of London (1996)

[8] Demirdžić, I. and Perić, M., Space conservation law in finite volume calculations of fluid flow, *Int. J. Num. Meth. Fluids*, **8**, pp. 1037-1050 (1988)

[9] Johnson, A. A. and Tezduyar, T. E., Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces *Computer Methods in Applied Mechanics and Engineering*, **7**, pp. 73-94 (1994)

[10] Löhner, R. and Yang, C., Improved ALE mesh velocities for moving bodies, *Communications in Numerical Methods in Engineering*, **12** pp. 599-608, (1996)

[11] Beaudoin, M. and Jasak. H., Development of an Arbitrary Mesh Interface for Turbomachinery simulations with OpenFOAM, *Open Source CFD International Conference, Berlin*, (2008)