Which strategy to move the mesh in the Computational Fluid Dynamic code OpenFOAM

Christophe Kassiotis

April 12, 2008

École Normale Supérieure de Cachan,
Laboratoire de Mécanique et Technologies (LMT)
Secteur Génie Civil,
61, avenue du président Wilson, 94 235 Cachan Cedex, France

TU-Braunschweig Braunschweig
Institut for Scientific Computing
D-38092 Braunschweig , Germany

Abstract

In this paper we compair some of the moving mesh strategies provided by OpenFOAM.

1 Introduction

Fluid-Structure interaction problems usually lead to unsteady moving domain for the fluid part. Traditional Computational Fluid Dynamic programs solve the fluid equations on a fixed (eulerian) grid. A classical approach to overcome this difficulty is to consider the so-called Arbitrary Lagrangian Eulerian (ALE) method where the grid is moved arbitrary inside the fluid domain, following the movement of the boundary.

However, this lead to new difficulties: knowing the new shape of the boundary, how can one conserve the quality and the validity of the inner mesh? Several approaches are traditionally proposed with mechanical analogy (*spring* or *pseudo-solid* analogy) or Laplacian smoothing.

By the way, a fluid-structure interaction problem become not only a two fields (fluid and solid) but a three fields coupling problem (fluid, solid and mesh). We have also to add that the global solution must not depend of the mesh motion, and this is naturally verified when the quality of the mesh is preserved.

The outline of the present paper is the following. In Section 2 we will show that the use of default mesh solvers can lead to problems for fluid-structure computations. The first two parts of the section can be quickly read for a someone who is not interested in fluid-structure interaction. That make us briefly introduce and recall the main solving strategy for the mesh motion (in OpenFOAM) in Section 3. Then, we will solve some examples in Section 4 to make a choice between some of this strategies. We conclude in Section 5.

2 Bring to light a mesh motion problem

2.1 Oscillating flexible structure in a flow

Developing fluid-structure interaction solvers by a partitioned strategy[1], we were trying to validate the following benchmark first introduced by Wall and Ramm [8, 7]. This benchmark (see sketch in Fig. 1) aims to solve the motion of a thin appendage behind a bluff body in a fluid flow at a Reynold number around 100.

For the fluid, the incompressible Navier-Stockes equations are solved, and compute on their discrete form on around 6000 Finite Volume (Q1) Element. For the solid, we consider a Saint-Venant-Kirchoff type material, based upon a finite deformation measure and a plane stress description. We discretize it on 40 9-nodes Finite Elements (Q9). An example of mesh associated to the discrete form of the equations can be seen for the fluid in Fig. 5.

The material properties are the following:

- for the fluid: dynamic viscosity $\nu_f = 1.51 \cdot 10^{-1}$[2] as $\rho_f = 1.18 \cdot 10^{-3}$ (like air) and $\mu_f = 1.82 \cdot 10^{-4}$ ($10\times$ more than air).
- for the solid: Young's modulus $E_s = 2.5 \cdot 10^6$, Poisson's ratio $\nu_s = 0.35$ and density $\rho_s = 0.1$.

[1] we are coupling CTL components based on Feap for the structural part and OpenFOAM for the fluid part

[2] like in [7, 8] the value are given without units. A careful reader can see that the unit for dimensions is $10^{-1}m$, the other units are the international system one (kg, $s\ldots$)

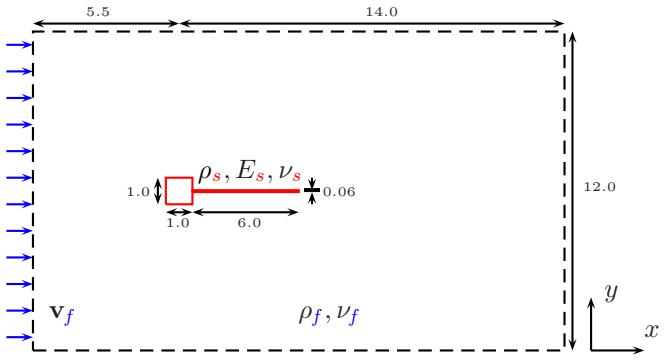


Figure 1: The benchmark used for FSI problems

The boundary conditions are imposed as follow:

- for the fluid: imposed velocity $v_f = [51.3, 0, 0]^T$ in input, slip condition for the upper and lower parts of the mesh, "do-nothing" output condition.
- for the solid: imposed null displacement at the origin of the appendage.
- for the interface: kinematic continuity ($\mathbf{v}_s = \mathbf{v}_f$) and dynamic equilibrium ($p_f = \boldsymbol{\sigma}_s \mathbf{n}$) are imposed for strong coupling or approximate for weak coupling.

The initial conditions are computed from a stationary flow regime with fixed structure for the fluid; they are null for the solid.

The fluid-structure interaction computation strategy used allows different time steps for the fluid and the structure. We consider here $\Delta t_f = 0.0002$ and $\Delta t_s = 0.005$ for the computation time interval $t \in [0, 5]$.

The solving of the fluid-structure interaction problem is carried out by the so-called partitioned strategy. The coupling can be weak (this is often the case for partitioned strategy) but strong too. For more details on the subject, the reader is invited to see [4]. In the *Institut for Scientific Computing* is developed a middleware called *Component Template Library* (CTL)[3] , developed by Rainer Niekamp [5]. The CTL allows to make components from codes, and to make them communicate easily. For the fluid-structure interaction framework, we consider mainly two components:

- ofoam developed by Martin Krosche [3], a component based on OpenFOAM[4].
- coFeap developed by Martin Hautefeuille and Christophe Kassiotis [2], a component based on Feap[5], the Finite Element Analysis Program from R. L. Taylor [6].

[3] http://www.wire.tu-bs.de/forschung/projekte/ctl/e_ctl.html
[4] http://www.opencfd.co.uk/openfoam
[5] http://www.ce.berkeley.edu/~rlt/feap

2.2 A problem with moving mesh

The first implementations of the fluid component Ofoam were carried out for OpenFOAM 1.2 and 1.3. The last release (OpenFOAM 1.4.1) was introduced in summer 2007 and Ofoam was adapted to support this new version. This new version gives me first the following sentiments: the `icoDyMFoam` solver seems to be really faster than before. As we will see in the following Section 3 this can be explain by the use of a Finite Volume based strategy to move the mesh.

Unfortunately, testing of this new solver gives sometime bad results. To be more precise, let's consider the Wall and Ramm benchmark describe above and in [8]; On the one hand applying material properties proposed herein on the seems to give relatively accurate results. On the other hand, applying for instance just a smaller stiffness to the structure than the value given above leads to computation crashes after a little more than 1s. The material properties above seems to be the one allowing maximum motion for the structure with the default solver in OpenFOAM 1.4.1.

The computation crash occurs during the increasing motion phase, when the displacement of the extremity is around $y_{\max} = 1.1$. The mesh shows clearly at this time non-convex elements: it becomes clear that this crash is mainly due to bad (or unadapted) mesh motion computations. In Figure 2), one can see the output for the fluid-structure interaction problem just before the crashing. Even if the velocity field seems correct, the mesh deformation exhibits non-convex elements. That leads to bad computation of the pressure field and crahsing computations.

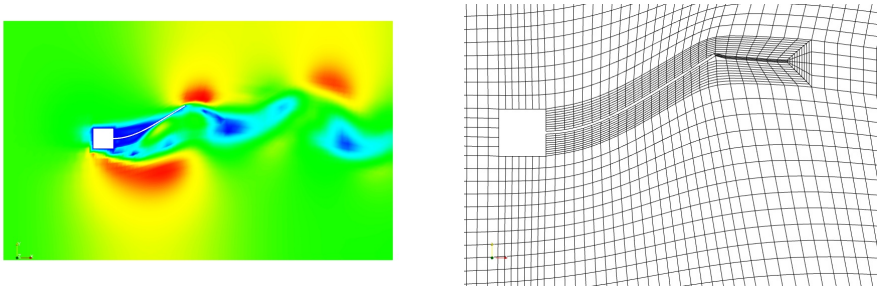


Figure 2: Velocity field in the fluid-structure interaction problem and its mesh motion just before crashing.

In the following section we will see how can one handle mesh motion in a CFD code like OpenFOAM.

3 Strategies to move the mesh

In this section, we briefly recalls the main explanations given by Jasak and Tuković about handling mesh motions in [1].

3.1 Mesh motion equations

Let us consider a domain $\mathcal{D}_{\text{old}}$ which represent the mesh configuration at a time t_{old}. This domain is moving through imposed condition at the boundary for

instance imposed displacement or velocity. The question is to build a new valid mesh $\mathcal{D}_{\text{new}}$ at $t_{\text{new}} = t_{\text{old}} + \Delta t$ knowing the initial valid mesh $\mathcal{D}_{\text{old}}$ and imposed boundary conditions.

It can be proved that the mesh motion problem is formally equivalent to a solid body under large deformations. The coast to solve this non-linear equation is tremendous and this strategy can't be applied in an efficient 3D CFD code. A traditional way to overcome this difficulty is to consider simplified solid equations for the mesh motion:

- The *spring analogy* aims to links each point of the mesh by fictitious spring. In [1], Jasak and Tuković show this leads failures modes. This failures modes can be avoided introducing non-linear and torsional springs. However, the cost induced by the improvement of this non perfect system by nature can be considered as too huge.
- The *pseudo-solid* equations can be considered as simplification of the 3D non-linear problem to the linear one; the analogy is a mechanical problem with small deformations.

A totally other strategy is to consider the Laplace smoothing equation:

$$\nabla \cdot (\gamma \nabla \mathbf{u}) = 0 \qquad \text{in} \quad \mathcal{D} \tag{1}$$

Where $\mathbf{u}$ is the mesh deformation velocity such that:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \Delta t \ \mathbf{u} \tag{2}$$

The Laplace smoothing equation does not allow to take account coupling of the components of the motion vector due to rotation. This coupled motion are handle by the *pseudo-solid* solver, but the computational cost can't be neglected.

In OpenFOAM, mesh motion solver based on a *pseudo-solid* and a Laplace smoothing equation are implemented. In the following, as the rotational coupling is not the main interest of this paper, we will concentrate one Laplace system.

3.2 Methods for mesh motion equations

The solve of the Laplace equation is insured by a numerical strategy. The simplest idea is too in a CFD code in Finite Volume framework is to use a Finite Volume Method. This method leads to many difficulties extensively described in [1], Section 4.3. The problem of mesh motion based on such a strategy will be shown in the following Section 4.

To overcome this difficulties, a Finite Element Method based strategy has been implemented in OpenFOAM [1]. Among all the ideas developing in this papers, one of great interest is the cell decomposition strategy. Each polyhedral cell is decomposed in a sum of linear tetrahedron where the expression of the Laplacian operator can be expressed without the use of shape functions.

4 Numerical Experiment

4.1 A rigid structure moving at imposed velocity in a flow

In this section we will not consider a fluid-structure interaction problem. To compare the different mesh motion solver considered, we will solve the following

problem: a flow at a Reynold number around 100 is surrounding an appendage fixed behind a square bluff body. The boundary and initial conditions are describe in Section 2 and in [8].

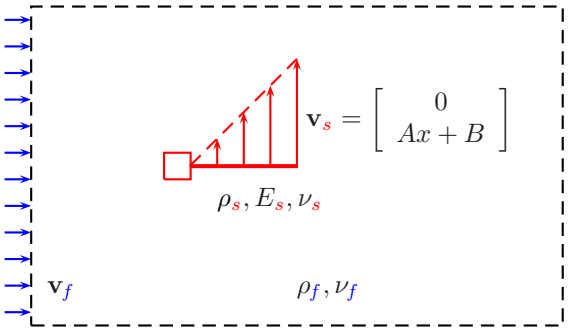


Figure 3: The benchmark used for moving mesh testing

However, as sketched in Fig. 3, the thin appendage is not here considered as an elastic body, but submitted to a bi-dimensional velocity field with the form $\mathbf{v}_s = [0, Ax + B]^T$ such that the velocity is null at the origin of the appendage ($\mathbf{v}_s(0) = [0, 0]^T$) and at the tip of the appendage L, its value is $\mathbf{v}_s(L) = [0, 4.0]^T$.

4.2 Qualitative comparison for different mesh motion solvers

The solver for the considered problem, have at least to solve the incompressible Navier-Stokes Equation in a moving shape domain. In OpenFOAM, it is natural to use the so-called `icoDyMFoam` solver, which fulfilled this minimum requirement. This program propose many options, and we will focus on the change of moving mesh options which are set in the `dynamicMeshDict` file (see Fig. 4).

First, we consider the default proposal in OpenFOAM-1.4.1 wich is the `velocityLaplacian` solver with a uniform diffusivity coefficient. This is the in the following way in the `dynamicMeshDict` file:

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.4.1                                 |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version         2.0;
    format          ascii;

    root            "";
    case            "";
    instance        "";
    local           "";

    class           dictionary;
```

```
20      object          motionProperties;
21  }
22
23  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
24
25  dynamicFvMesh          dynamicMotionSolverFvMesh;
26  twoDMotion             yes;
27  solver                 velocityLaplacian;
28  diffusivity            uniform;
29  frozenDiffusion        off;
30
31  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

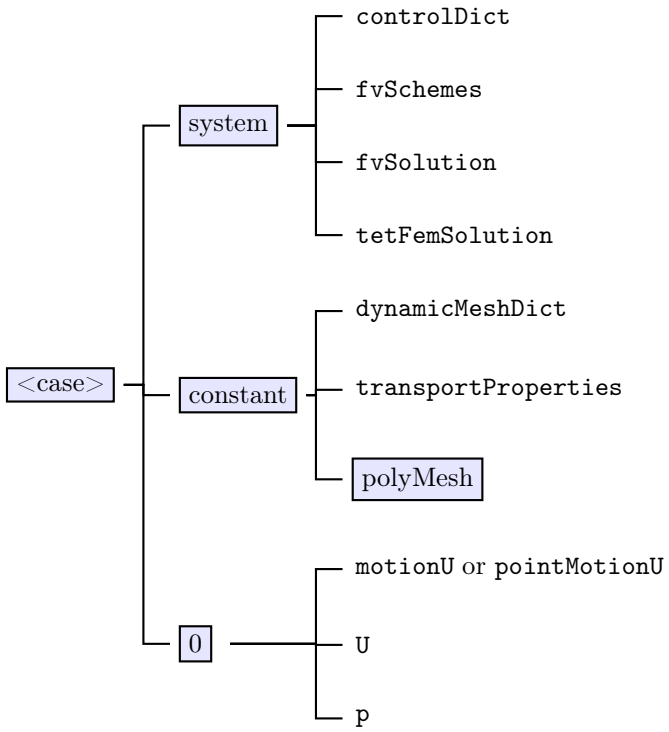
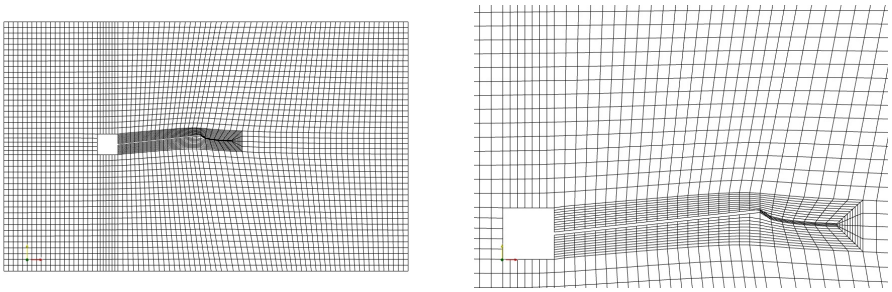


Figure 4: OpenFOAM architecture case

The velocities at the boundary have to be given in a `pointMotionU` file (see Fig. 4) under the form of `pointVectorField` so that the non-uniform values are set *only* at each point of the mesh's boundaries.

As seen in Fig. 5, this quickly lead to bad mesh deformations, and at $t = 0.11s$, when the maximum displacement at the tip of the appendage is 0.44, one can observe computation crashing due to non-convex element.

The simplest way to overcome this difficulty is to modify the diffusivity coefficient. OpenFOAM proposes many options, but it seems that whatever the case considered, the most robust results are given by quadratic inverse dependency. With this option, the diffusivity coefficient is somehow proportional to $\sim 1/l^2$ where l is the distance to a specified boundary. So, the `blockMeshDict` file take the following form:



Time $0.11s$

Figure 5: Mesh motion solving uniform Laplace equation.

```
23  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
24
25  dynamicFvMesh          dynamicMotionSolverFvMesh;
26  twoDMotion             yes;
27  solver                 velocityLaplacian;
28  diffusivity            quadratic inverseDistance 1(movingWall);
29  frozenDiffusion        off;
30
31  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

As seen in Fig. 6, the inverse quadratic option allows a better conservation of mesh quality in the close narrows of the moving boundary. However, at $t = 0.66$, when the maximum displacement is 2.68 some of the elements become non-convex.

The next step is to change the kind of solver used. With the previous solver `velocityLaplacian` we were solving the Laplace smoothing equation by FVM. With `laplaceFaceDecomposition`, not only the Laplace smoothing equation is solved for a given mesh by the Finite Volume Method, but the mesh is rebuild after a decomposition of all cells and faces, and the Laplace smoothing equation is solved by the Finite Element Method (see Sec. 3 for more details). The tremendous gain in robustness observed are paid by a computational coast (see the following Sec. 4.3 for more details).

The `laplaceFaceDecomposition` is first applied with uniform diffusivity. The used of this solver is specified in the `blockMeshDict` file by the following entrances:

```
23  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
24
25  dynamicFvMesh          dynamicMotionSolverFvMesh;
26  twoDMotion             yes;
27  solver                 laplaceFaceDecomposition;
28  diffusivity            uniform;
29  frozenDiffusion        off;
30
31  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

For the FE based solvers, the velocities at the boundary have to be given in a `motionU` file (see Fig. 4) under the form of `tetPointVectorField` so that

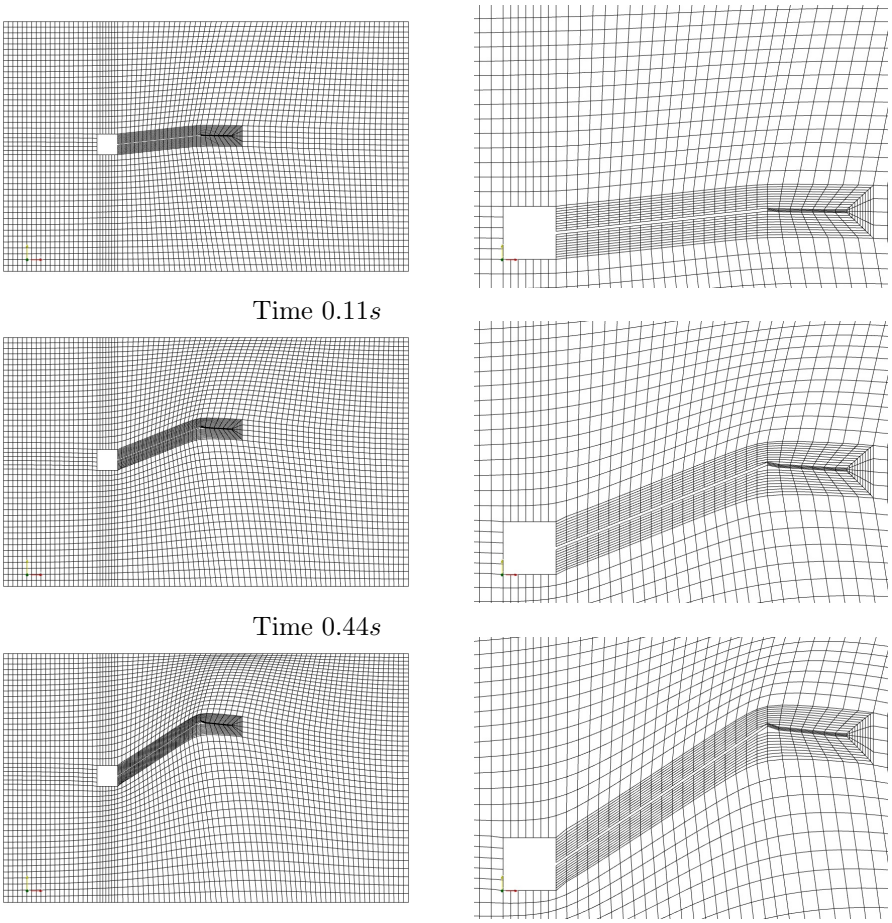


Figure 6: Mesh motion solving a non uniform Laplace equation with quadratic inverse distance to the moving boundary dependancy.

the non-uniform values are set at *each point and face center* of the mesh's boundaries.

Even if the deformation of the meshes observed in Fig. 7 are far from being really smooth and beautiful, the first non-convex element that leads to crashing the computation is observed only at $t = 0.84s$, when the maximum displacement is $3,36$.

The last improvement is to consider a quadratic inverse distance moving mesh specified as follow in the `blockMeshDict` file:

```
23 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
24
25 dynamicFvMesh          dynamicMotionSolverFvMesh;
26 twoDMotion             yes;
27 solver                 laplaceFaceDecomposition;
28 diffusivity            quadratic;
```

```
distancePatches         (movingWall);
frozenDiffusion         off;

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

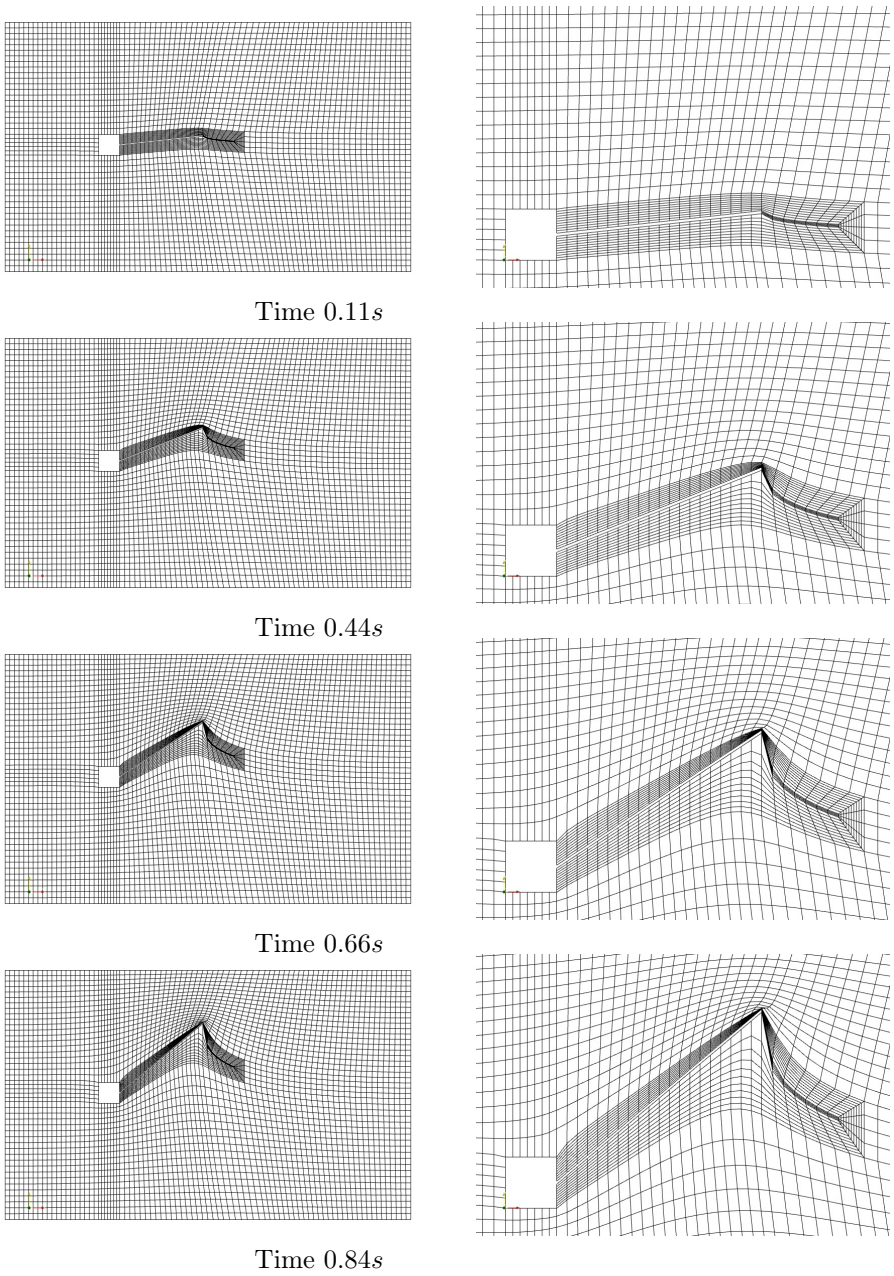


Figure 7: Mesh motion with a face decomposition strategy solving a uniform Laplace equation.

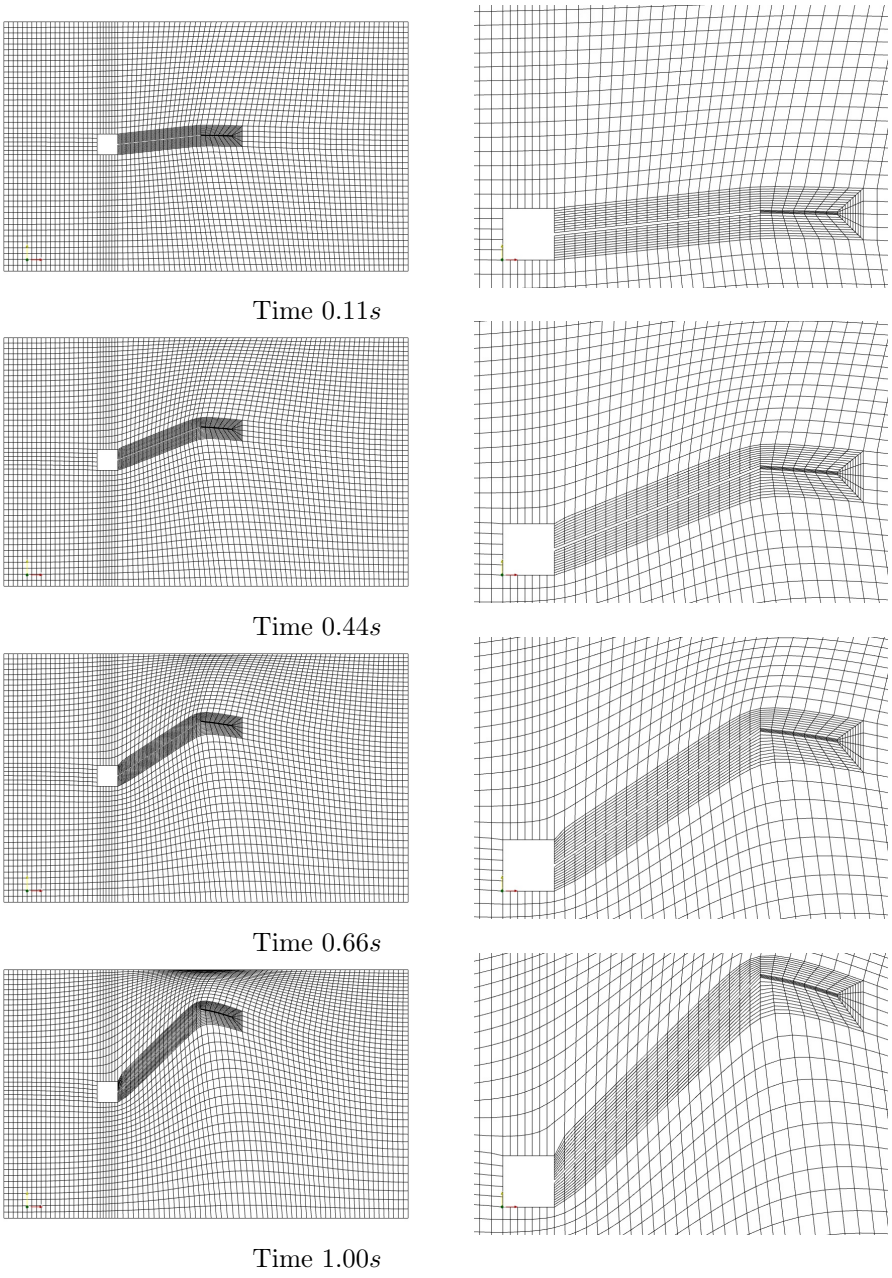


Figure 8: Mesh motion with a face decomposition strategy solving a non uniform Laplace equation with quadratic inverse distance to the moving boundary dependency.

4.3 Quantitative comparison for different mesh motion solvers

In this section we will denote by $\tilde{t}$ all computational time, and by t the *physical* time for the computation. In the considered simulations, $t \in [0, 1]$ so that the maximum displacement awaited is $y_{\max} = 4.0$. The reference value are taken for the simulation with the finite element method and non uniform Laplace smoothing. For the quantitative comparison, we consider two non-dimensional variables:

- The computational coast c: is the computational time for one time step divided by the time step size. The value is taken as the mean for all the time of the computation before the crash (if it occurs). This not on the advantage of the FEM method that allows to compute much more disordered meshes with higher max. Courant number. This value is then divided by the found for the `LaplaceFaceDecomposition` solver:

$$c = \frac{\tilde{t}(t_{\max}}{\min(t_{\max}, 1s)} \times \frac{1s}{\tilde{t}_{\text{ref}(1s)}} \tag{3}$$

- The robustness r: is taken as the time when the computation crash divided by the reference value taken for the Finite Element Method with *cell and face* decomposition solver and a quadratic diffusivity, value of $1s$ (we consider that a tip displacement of more thant 4.0 will necessitate also to change the mechanical model).

$$r = \frac{\min(t_{\max}, 1s)}{1s} \tag{4}$$

The results can be found in Tab. 1.

case	robustness	computational coast
1	0.11	0.428
$2a$	0.66	0.453
$2b$	0.66	0.575
3	0.67	1.747
4	1.00	1.000

Table 1: robustness and computational coast for four moving mesh methods. 1 - uniform Laplace equation solved by FV. 2 - non-uniform Laplace equation solved by FV (a for OpenFOAM-1.4.1-dev, and b for OpenFOAM-1.4.1). 3 - uniform Laplace equation solved by FE *cell and face* decomposition. 4 - non uniform Laplace equation solved by FE *cell and face* decomposition.

5 Conclusion

As a conclusion, for fluid-structure interaction problem coupling `icoDyMFoam` solver from OpenFOAM, we advice:

- to use the solver `velocityLaplacian` each time an order of magnitude of the maximum displacement is known to be not too big.

- to use the solver `laplaceFaceDecomposition` when the order of magnitude of the maximum displacement is not known, or when it is known to be big.

Whatever is the case, one shall use an inverse quadratic diffusivity coefficient.

Other solver, and among them *pseudo-solid* and only *cell* decomposition solvers were not extensively tested. For the first one, the computational coast seems to be extremely high for no gain in most of the fluid-structure interaction cases. For the second one, the result in term of computational coast and reliability can be compared to *cell and face* decomposition presented herein.

Acknowledgment

The author wants to acknowledge Hrvoje Jasak for help, support and quick answering in the OpenFOAM forum[6].

Martin Krosche is also acknowledged for constant sharing office and great (scientific) conversations.

References

[1] H. Jasak and Z. Tuković. Automatic mesh motion for the unstructured finite volume method. *Transactions of FAMENA*, 30(2):1–18, 2007.

[2] C. Kassiotis and M. Hautefeuille. *coFeap's Manual*. École Normale Supérieure de Cachan, Laboratoire de Mécanique et Technologies, Secteur Génie Civil, 2008. http://www.lmt.ens-cachan.fr/cofeap.

[3] M. Krosche. Ofoam's manual. Informatikbericht, Institute for Scientific Computing, 2008. (In Preparation).

[4] H. G. Matthies, R. Niekamp, and J. Steindorf. Algorithms for strong coupling procedures. *Computer Methods in Applied Mechanics and Engineering*, 195:2028–2049, 2006.

[5] T. Srisupattarawanit, R. Niekamp, and H. G. Matthies. Simulation of nonlinear random finite depth waves coupled with an elastic structure. *Computer Methods in Applied Mechanics and Engineering*, 195:3072–3086, 2006.

[6] R. L. Taylor. *FEAP – a Finite Element Analysis Programm – User Manual*. University of California at Berkeley, Departement of Civil and Environmental Engineering, 2008. http://www.ce.berkeley.edu/~rlt/feap/manual.pdf.

[7] W. A. Wall. *Fluid-Struktur Interaktion mit stabilisierten Finiten Elementen*. Phd Thesis, Institut für Baustatik, Universität Stuttgart, 1999.

[8] W. A. Wall and E. Ramm. Fluid-structure interaction based upon a stabilized (ALE) finite element method. Sonderforschungsbereich 404, Institute of Structural Mechanics, 1998.

[6] see particularly the thread about the question described in this paper http://openfoam.cfd-online.com/cgi-bin/forum/show.cgi?1/7079