Moving mesh for complex geometries

Wafa Daldoul

ECOLE NATIONALE SUPERIEURE DES MINES DE PARIS

CENTRE DE MISE EN FORME DES MATERIAUX

Moving mesh for complex geometries

Wafa DALDOUL

2 février 2017

The three-dimensional simulation of physical phenomena involving moving bodies undergoing large displacements represents a real challenge. These simulations, combine the issues related to instability, meshing and Fluid-Structure Interaction, are generally difficult to perform and very costly in terms of computation time. We present in this research report an overview on moving mesh methods used in the literature. Several mesh deformation methods are proposed, local remeshing operations are explained and finally different approaches that uses PDE are detailed.

Contents

Introduction

In this project we are interested in specific metal manufacturing processes. The studied processes involve several heat transfer mechanisms. Studying these heat transfer mechanisms is very important since they condition the temperature distribution of the final part and its quality. The numerical simulation has proved to be a powerful tool for understanding and controlling these phenomena.

The software THOST is dedicated to the simulation of heat transfer phenomena occurring in furnaces and quenching tanks[18, 17]. In order to make the simulation of complex processes more realistic, THOST software has to be able to simulate the overall process as a single problem and to control both heating and cooling as a unique continuous and integrated process. This is the objective of HECO project: the control of the history of industrial parts along their heat treatment, from HEating to COoling.

The variety of the industrial equipment (turbines, fans,..) has to be taken into consideration but also the displacement of the parts from one equipment to another.

The three-dimensional simulation of these physical phenomena involving moving bodies undergoing large displacements represents a real challenge. These simulations, which combine the difficulties related to instability[16, 22], meshing and Fluid-Structure Interaction, are generally difficult to perform and very costly in terms of computation time.

In such applications, as the body moves, the mesh may require adaptation at each time step to ensure good representation of the fluid-solid interface or the existing mesh has to be allowed to deform to track the moving geometry. The former solution is computationally expensive, especially for 3D problems but also requires an interpolation step to map the solution into the newly generated mesh. This interpolation step is itself time-consuming but also introduces additional errors. The latter option introduces the concept of a "moving" and "deforming" mesh.

In the case of a deforming mesh, the mesh deformation has to be accomplished efficiently and reliably in order to robustly perform the FSI simulations. Solver stability and accuracy is a primary concern in mesh moving algorithms. As the mesh deforms, elements can become inverted or highly skewed which can result in solver stability issues and errors degrading the accuracy of the model. For this, the algorithm used for moving the mesh has to be efficient and robust and able to handle large deformations. Usually, moving mesh algorithms are coupled to some mesh optimization operations that aim at maintaining the best possible the mesh quality while deforming it using either several local remeshing operations such as vertex addition, vertex collapsing, connectivity change or some mesh smoothing methods based only on vertex displacements. In addition to that, the efficiency of the computations in terms of CPU time represents also a great challenge.

The aim of this thesis is to develop a robust moving mesh algorithm able to handle large displacements and efficient in terms of CPU time. In our context, a finite elements scheme and unstructured meshes are used.

A brief bibliographic overview of the moving mesh methods used in the literature will be given. An outline of this report is given as follows: In section.1 the different mesh deformation methods used in the literature will be given. In section.2 some local remeshing operations will be explained and finally in section.3 the different approaches used for solving the governing equations are presented.

1 Mesh deformation methods

In moving mesh simulations, the whole mesh has to be allowed to deform in order to follow the displacement of the geometry while preserving the validity of the computational mesh. The whole problem can be formulated as follows: knowing the displacement of the vertices located on the moving body, what displacement must be applied to the rest of the domain vertices in order to respect the above description.

In general, a moving mesh algorithm has to be efficient in terms of CPU time as it is called at each solver time step. It has also to be able to keep a valid mesh so that remeshing operations remain very occasional in order to avoid the induced errors due to the interpolation steps required after any remeshing. It has to be able to preserve a good quality of the mesh so that the solution accuracy is not degraded. The main mesh deformation methods usually used in the literature will be given in this section.

1.1 Mechanical Analogy

In the case of the mechanical analogy based methods, two alternatives are generally considered: the spring analogy methods and linear elasticity methods.

1.1.1 Spring analogy method

The method views the fluid mesh as a network of springs. In the spring analogy first presented in [41], a fictitious tension/compression spring is attached to each edge connecting two vertices i and j, the stiffness of the spring k_{ij} is chosen to be inversely proportional to the length of the supporting edge l_{ij} :

$$K_{ij} \propto \frac{1}{l_{ij}}$$

The above definition of K_{ij} is driven by the fact that if during the mesh movement two vertices tend to get closer, the spring attached to the edge they belong to becomes stiffer and therefore prevents them from colliding.

This method often results in invalid meshes due to mesh lines crossovers; this is due to the fact that the tension/compression springs prevent two vertices from colliding but does not prevent a vertex from crossing an edge that faces it. This method has been improved by Farhat in [43] that introduced torsional springs at the mesh vertices to prevent neighboring triangles from interpenetrating each other; for a triangle whose vertices are designed by i, j and k, let θ_i^{ijk} denote the angle between two edges ij and ik of this triangle, the stiffness of a torsional spring is given by:

$$C_i^{ijk} = \frac{1}{sin^2\theta_i^{ijk}} = \frac{l_{ki}^2 l_{kj}^2}{4{A^{ijk}}^2}$$

where, A^{ijk} is the area of the considered triangle, l_{ki} is as before the length of the edge connecting k and i, and $l_{kj}$the length of the edge connecting k and j.

The above stiffness coefficient is meant to prevent both vertex-to-vertex and vertex-to-edge collisions. Actually, whether vertex k moves towards vertex i or j, or whether it moves towards edge $i-j$, this leads the area A^{ijk} of the triangle to decrease, which in turn results in the increase of the stiffness coefficient C_i^{ijk} .

This method despite of the attempts to improve it has shown law robustness and tends to deteriorate the quality of the mesh when large displacements are considered [39].

1.1.2 Linear-elasticity method

In the case of the linear elasticity analogy [1], the motion of the internal nodes is governed by the equations of elasticity.

the movement of the vertices is obtained by solving an elasticity-like equation:

$$div(\sigma(\varepsilon)) = 0,$$

with

$$\varepsilon = \frac{\nabla d + (\nabla d)^T}{2},$$

where σ and ε are respectively the Cauchy stress and strain tensors, and d is the Lagrangian displacement of the vertices. The Cauchy stress tensor follows the Hooke's law for isotropic homogenous medium, where ν is the Poisson ratio, E the Young modulus of the material, and λ and μ are the lame coefficients and I_d the identity tensor:

$\sigma(\varepsilon) = \lambda trace(\varepsilon) I_d + 2\mu\varepsilon$, and $\varepsilon(\sigma) = \frac{1+\nu}{E}\sigma - \frac{\nu}{E} trace(\sigma) I_d$

As boundary condition, the motion of the nodes at the interfaces is specified to match the normal velocity of the fluid at the interface.

One advantage of this method is that it offers the possibility to introduce a special treatment based on element size, actually, it is possible to adapt the local stiffness according to the size of each element so that smaller elements are stiffened more than the large ones [52]. This can be realized by modifying the way the Jacobian transformation from the element domain to the physical domain is accounted for in the FEM matrix assembly [35].

Classically, the P^1Finite Element formulation of the linear elasticity matrix leads to the evaluation of quantities of the form:

$\int_K s \frac{\partial \varphi_J}{\partial x_k} \frac{\partial \varphi_I}{\partial x_l} dx = \frac{s}{36|K|} (\bar{\eta_J})_k (\bar{\eta}_I)_l$

where s is either λ, μ or $\lambda + 2\mu$ and $\bar{\eta_I} = ((\bar{\eta_I})_x, (\bar{\eta_I})_y, (\bar{\eta_I})_z)$ is the inward non-normalized normal opposite to vertex i in tetrahedron K .In [35], the above quatity is replaced by :

$\int_K s \frac{\partial \varphi_J}{\partial x_k} \frac{\partial \varphi_I}{\partial x_l} \left(\frac{|\hat{K}|}{|K|}\right)^{\chi} dx = \left(\frac{|\hat{K}|}{|K|}\right)^{\chi} \frac{s}{36|K|} (\bar{\eta_J})_k (\bar{\eta}_I)_l$

where $\hat{K}$ is the reference element. This method stiffens each element by a factor proportional to $|K|^{-\chi}$and $\chi > 0$ is the stiffening power that determines the degree by which the smaller elements are rendered stiffer than the larger ones.

But, this method can only handle small incremental deformations and may require much subiteration to complete the full deformation of the mechanical time step. It is usually coupled to regular optimization phases to maintain a quality compatible with the desired accuracy.

Both spring analogy and linear elasticity methods need mesh connectivity information to construct a system of equations of the size of the number of flow nodes n_i in order to calculate the displacement of the mesh. Solving this $n_i \times n_i$ matrix repeatedly is computationally intensive.

For the case of the linear elasticity method, in [40], the author tried to find a way to reduce the CPU time dedicated to this method. Actually, instead of solving the elasticity system at each solver time step, its solution is done once for a large time step Δt_{max}. After solving the elasticity system, the mesh motion is analyzed. If the mesh is not valid, then the elasticity time step is reduced until validity is achieved: $\Delta t = \frac{\Delta t_{max}}{2^n}$. Some results obtained by this method are shown in the Figure1.

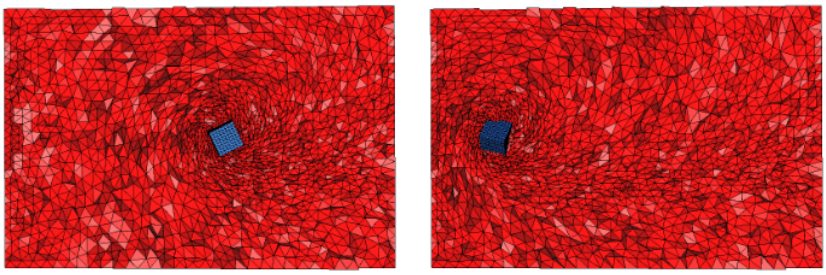


Figure 1: Snapshots of a moving rotating cube using the method in [40]

1.2 Adaptive MMPDE-based methods

Mesh adaptation plays an indispensable role in the efficient numerical simulation of many physical phenomena since it helps reducing the dedicated CPU time to these simulations while keeping high accuracy[28, 27, 29]. Some authors view the theory of moving meshes as part of an r-adaptive strategy for solving partial differential equations with moving bodies.

The idea is to move the mesh points and concentrate them in regions where the solution has a "special" behavior, usually a rapid variation of either the solution or its derivatives[10, 26]. The objective is to get the smallest error possible for the number N of mesh nodes. A vector or a scalar monitor function is used in order to control the size, shape and orientation of the elements of the mesh to be generated. The monitor function is usually designed to give an estimate of some measure of the solution error which is then equidistributed over each mesh cell.

The mesh adaptation is viewed as a mathematical equivalent of the determination of a coordinate transformation. The adaptive mesh is viewed as the image of a uniform reference mesh under a coordinate transformation (mapping) $x = x(\xi)$ from a regular domain (the computational domain) $\Omega_c = \{\xi_j, j = 1, ..., N_v\}$ to an irregularly-shaped domain (the physical domain) $\Omega = \{x_j, j = 1, ..., N_v\}$. $J = \frac{dx}{d\xi}$ is used to denote the Jacobian matrix of the mapping $x(\xi)$.

Several techniques have been developed in order to generate moving adaptive meshes. Most of these approaches can be classified into two groups. The first group is called the location-based method (or the coordinate-based method) because it controls directly the location of the mesh nodes. The second group is called the velocity-based approach since it controls the time derivative of the mapping or the mesh velocity. In other words, the mesh equation is formulated for the mesh velocity and the mesh point location is obtained by integrating the velocity field. In the following paragraphs, good representatives of each group will be given.

1.2.1 Velocity-based approaches:

1.2.1.1 The classical Lagrangian method In fluid dynamics, Lagrangian coordinates constitute a moving coordinate system used to follow fluid particles. If $u(x, t)$ is the velocity of the fluid, ξ the reference coordinate of a fluid particle and $x(\xi, t)$ the position of the particle at time t, then the particle and consequently the Lagrangian coordinates envolve with:

$$\frac{\partial x}{\partial t} = u \tag{1}$$

An interesting property of Lagrangian coordinates is that convection terms are eliminated from the governing equations. However, a purely Lagrangian formulation is usually avoided especially in problems involving large deformations since they tend to produce highly skewed meshes resulting in the failure of the numerical calculation.

1.2.1.2 The geometric conservation law method The so-called geometric conservation law (GCL) was used for many years in the engineering community to develop cell-volume-preserving finite volume schemes. In [54] for example, the GCL was used to eliminate oscillations and preserve physical conservation laws for solutions on moving meshes.

In [4, 53] it is used to transform the algebraic expression specifying the Jacobian into equivalent differential equation which is the key for the moving mesh strategy. The GCL is given by the following definition:

For any coordinate transformation $x(\xi, t)$ from Ω_c to Ω, the mesh speed x_t and the time derivative of the Jacobian matrix for the mapping $x(\xi)$ from Ω_c to Ω satisfy:

$$\nabla . x_t = -\frac{1}{J}\frac{DJ}{Dt} \tag{2}$$

Let's consider the determination of the mapping $x(\xi, t)$ through its time derivative. For a given monitor function $\rho(x, t) > 0$, we want the cell area to be inversely proportional to $\rho(x, t)$ at each time t. To do so, the mapping $x(\xi, t)$ is required to satisfy:

$$\nabla . x_t = -\frac{1}{\rho}\frac{D\rho}{Dt} \tag{3}$$

By comparing equations (2) and (3) we have :

$$\frac{D}{Dt}(\rho J) = 0 \tag{4}$$

Which implies that:

$$\rho J = constant \tag{5}$$

,

(3) can be written as:

$$\nabla . (\rho x_t) + \frac{\partial \rho}{\partial t} = 0 \tag{6}$$

Using only this equation is not sufficient to determine the mesh speed x_t. Using the Helmholtz Decomposition Theorem for vectors:" *A continuous and differentiable vector field can be resolved into the orthogonal sum of the gradient of a scalar field and the curl of a vector field*" a supplementary equation can be provided. Actually, x_t can be calculated using its divergence through (6) and its curl. For this $x(\xi, t)$ is required to satisfy :

$$\nabla \times \omega(x_t - u) = 0 \tag{7}$$

where ω and u are, respectively, a weight function and a background velocity field to be specified. Different choices of ω lead to different curl conditions.

Equations (2) and (7) can be formulated as a minimization of the functional:

$$I_{GCL} = \frac{1}{2}\int_{\Omega} |\nabla.(\rho x_t) + \frac{\partial \rho}{\partial t}|^2 + \left(\frac{\rho}{\omega}\right)^2 |\nabla \times \omega(x_t - u)|^2 dx \tag{8}$$

The following boundary condition is considered:

$$x_t.n = 0 \tag{9}$$

Equations (8) and (9) constitute an elliptic system for the mesh velocity x_t, its solution exists and is smooth. By satisfying the relation (5) the Jacobian of the mapping is determined by the monitor function $\rho(x, t)$ and thus, the Jacobian stays positive and the mapping stays itself locally non-singular. However, this method usually tend to produce highly skewed meshes.

1.2.1.3 The moving finite element method The moving finite element method (MFE) developed in [31, 30] is also a velocity-based method. Given a time dependent physical problem:

$\frac{\partial u}{\partial t} = \mathcal{L}u$

where $\mathcal{L}$ is a spatial differential operator. Determining x_t using the MFE method, consists in minimizing the L2-norm of the residual involving u and x_t over the entire space:

$I_{mfe}\left[x_t, \frac{Du}{Dt}\right] = \int\limits_{\Omega} \left(\frac{Du}{Dt} - \nabla u.x_t - \mathcal{L}u\right)^2 \omega dx$

In the classical version of MFE [31, 30] ω is taken equal to 1 and in the gradient weighted MFE (GWMFE) [5, 6] ω is equal to $1/(1+|\nabla u|^2)$.

One positive point of the MFE method is that the resulting mesh has the tendency to follow a path corresponding to the smallest L2-norm of the residual of the discrete equations. The problem is that the mesh resulting from only the minimization of this error can become degenerate or nearly degenerate and requires special regularizing terms (i.e., penalty functions).

1.2.2 Location-based approaches

1.2.2.1 Winslow's variable diffusion functional Winslow's formulation [55] requires solving a non-linear Poisson-like equation in order to generate the mapping. Suppose that higher mesh concentrations are desired in regions with large values of a given function $\omega(x) > 0$. With the approach of Winslow, the mapping is defined as the minimizer of :

$$I_{Win}[\xi] = \int_{\Omega} \frac{1}{\omega} \sum_i (\nabla \xi_i)^T (\nabla \xi_i) dx$$

1.2.2.2 The functional of Brackbill and Saltzman Brackbill and Saltzman developed a global coordinate approach based on modifying the Winslow algorithm. In [2] they tried to combine mesh smoothness, concentration and orthogonality. The functional associated with mesh concentration for example is given by the integral:

$$I_{BS}[\xi] = \int_{\Omega} \omega(x) J dx$$

One difficulty that arises is that there is no theory to guide the relative weights of each integral and they are left to be specified by the user. In addition to that, the various terms are not dimensionally homogenous and must be rescaled according to each specific application. One more thing is that one cannot guarantee the existence of a solution to the previous equation (unless it is a pure smoothness functional).

1.2.2.3 Harmonic mapping In [13] Dvinsky developed a method for generating adaptive grids based on harmonic maps. Given an $n \times n$ symmetric positive definite matrix $G(x)$ with $g = det(G)$, the mapping is defined as a harmonic mapping between $\Omega \subset \mathcal{R}^n$equipped with metric G and $\Omega_c \subset \mathcal{R}^n$equipped with the Euclidean metric. This is equivalent to finding the minimizer of:

$$I_{hrm}[\xi] = \int_{\Omega} \sqrt{g} \sum_i \left(\nabla \xi_i\right)^T G^{-1} \left(\nabla \xi_i\right) dx$$

In [48], the authors used a moving mesh strategy similar to Dvinsly's scheme except the fact that a finite elements approach was used rather than a finite difference discretization and the fact that the harmonic mapping in this case was constructed by an iterative procedure.

1.2.2.4 Brackbill's direction control functional Based on Winslow method and the harmonic mapping method, Brackbill [42] developed a functional which takes into consideration both mesh concentration and mesh alignment. The functional can be written as follows:

$$I_{brb}[\xi] = \int_{\Omega} \sum_i \left(\nabla \xi_i\right)^T G_i^{-1} \left(\nabla \xi_i\right) dx$$

for user-defined functions G_i. Depending on the desired properties of the mesh, they can be chosen in several ways. For example, in the case $G_i = \frac{1}{\sqrt{g}} G$, the functional reduces to a harmonic map. If, for instance, the mesh concentration is of primary concern, the natural choice is $G = \omega(x) I$, which leads to Winslow's functional. If the mesh is supposed to align with a certain prescribed field(v_1, v_2, v_3), then G can be chosen such that $\left(\nabla \xi_i\right)^T G^{-1} \left(\nabla \xi_i\right) = \left| v_i \times \nabla \xi_i \right|^2$.

1.2.2.5 Huang's approach The previous methods succeed in concentrating mesh nodes in regions where the monitor function is large. However, the precise relation between the mesh and the monitor function is unclear. One may observe an improvement of the solution accuracy using the previous types of mesh generation but it is not clear whether the obtained mesh is optimal in terms of minimization of the actual error or not. In order to deal with this problem, Huang in [19] formulates a functional based more directly on error distribution.

In [49] the mapping is formulated by minimizing an energy based on the so-called equidistribution (1) and alignment (2) conditions [20] and the mesh adaptation is controlled through a matrix-valued function.

Equidistribution condition:

$|K|\sqrt{det(G}_K) = \frac{\sigma_h|K_c|}{|\Omega_c|}, \forall K \in \Omega$

Alignment condition:

$\frac{1}{d}trace((J)G^{-1}(J)^{-T}) = det((J)G^{-1}(J)^{-T})^{\frac{1}{d}}, \forall K \in \Omega$

where, $|K|$ is the volume of K, G is the metric tensor associated with the adaptive mesh, G_K is the average of G over K, $det(.)$ and $trace(.)$ denote the determinant and trace of a matrix, respectively, $|K_c|$ is the volume of the element $K_c \in \Omega_c$ corresponding to K, and

$\sigma_h = \sum_{K\in\Omega} |K|\sqrt{det(G}_K), \ |\Omega_c| = \sum_{K_c\in\Omega_c} |K_c|$

The meshes that closely satisfy these conditions can be obtained by minimizing the energy function:

$I_h = \theta \sum_{K\in\Omega} |K|\sqrt{det(G}_K) trace((J)G_K^{-1}(J)^{-T})^{\frac{dp}{2}} + (1-2\theta)d^{\frac{dp}{2}} \sum_{K\in\Omega} |K|\sqrt{det(G}_K) \ (\frac{|K_c|}{|K|\sqrt{det(G}_K)})^p$

The above functional is used for generating an adaptive mesh for a specified monitor function. When one is solving a time dependent problem, and in order to move the mesh in a smooth manner the MMPDE approach is used to define the moving mesh equation as a gradient system of I_h [21, 46]

$\frac{d\xi_j}{dt} = -\frac{P_j}{\tau} \left[\frac{\partial I_h}{\partial \xi_j}\right]^T, \ j = 1, ..., N_v$

where the derivative of $I_h$with respect to ξ_j, $^{\partial I_h}/_{\partial \xi_j}$, is considered as a row vector, $\tau > 0$ is a parameter used to control the response time of the mesh movement to the change in the metric tensor.

In [36] a specific map is obtained by minimizing a mesh adaptation functional of the following form:

$E(\xi, \eta) = \frac{1}{2} \int_\Omega (\nabla \xi^T G^{-1} \nabla \xi)$

This functional is minimized by solving the corresponding Euler-Lagrange equation :

$\nabla.(G^{-1}\nabla\xi) = 0,$

This system is solved using Gauss-Seidel iteration. The iteration is continued until there is no significant change in calculating new grids from one iteration to the next.

1.2.3 Numerical comparison

In [44], the authors tried to present a short numerical comparison of four adaptive moving mesh methods : the GCL method, the winslow's approach, the harmonic map approach and Huang's

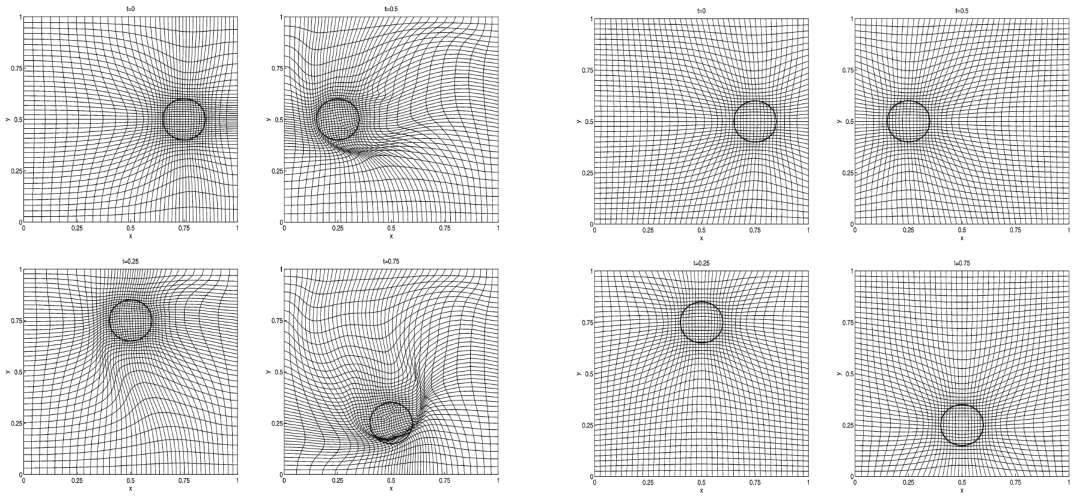
approach.

The monitor function used is defined as follow:

$$\rho(x,y,t) = 1 + 10exp\left(-50\left|\left(x - \tfrac{1}{2} - \tfrac{1}{4}cos(2\pi t)\right)^2 + \left(y - \tfrac{1}{2} - \tfrac{1}{4}sin(2\pi t)\right)^2 - \left(\tfrac{1}{10}\right)^2\right|\right)$$

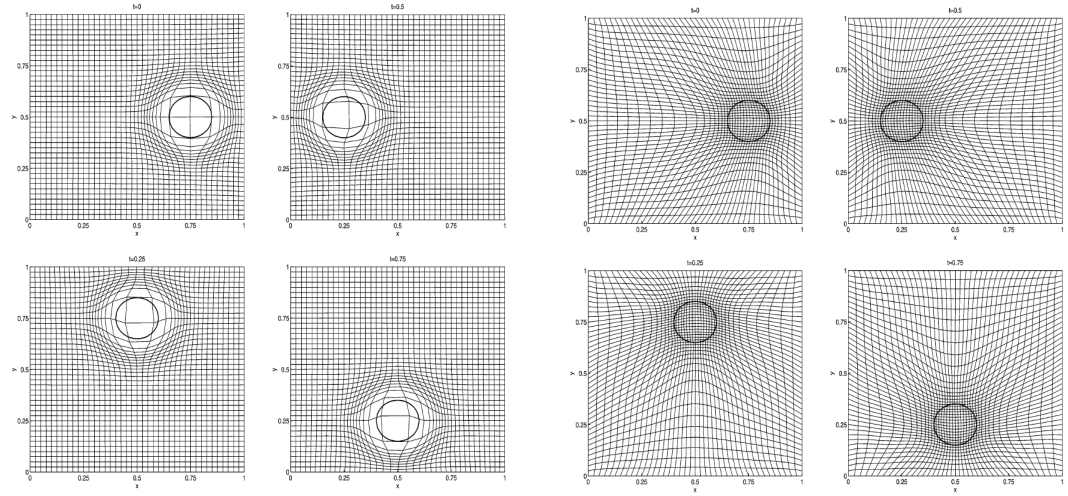
This equation needs the generation of a moving adaptive mesh from the time-dependent monitor function.

The figures below show the adaptive meshes at different times for the four methods.



(a) The adaptive meshes obtained by the GCL method at t=0,0.25,0.5,0.75[44]

(b) The adaptive meshes obtained by Winslow's method at t=0,0.25,0.5,0.75[44]



(c) The adaptive meshes obtained by the harmonic map method at t=0,0.25,0.5,0.75[44]

(d) The adaptive meshes obtained by Huang's method at t=0,0.25,0.5,0.75[44]

Figure 2: Numerical Comparison

For winslow's and Huang's methods, the mesh concentration follows closely the evolution of the function ρ. For the harmonic mapping, the cercle with the highest mesh concentration is slightly outside the cercle with the maximum ρ. Moreover, the mesh in the central part of the cercle is much coarser than in the outer region although ρ takes approximately the same valye in both places. This

was also noticed in [3]. In the case of the GCL, the mesh becomes highly skewed and the mesh concentration slowly misplaced.

1.3 Level-set moving mesh partial differential equation (LMPDE)

The Level-set deformation method [25, 32] moves the nodes with a proper velocity so that the nodal mapping has the desired Jacobian determinant. The grid in this method is represented in an implicit manner; the intersection points of the level-sets will form the grid nodes at each time.

For example in [32], for 2D simulations, the authors used two level-sets of the following form to represent the grid:

$\psi_1(x, y, 0) = x, \;\; \psi_2(x, y, 0) = y,$

The nodes of the mesh are after moved using the usual level-set evolution equation. The velocity vector used in the evolution equation is constructed by solving a Poisson equation determined by the monitor function.

The problem of this method is that it is able to produce only structured grids.

1.4 Optimization-Based Rezone strategy

The reference Jacobian optimization-based rezone strategy developed in [50] aims at producing a mesh of high quality while keeping it as close as possible to the Lagrangian mesh. The essential idea of the RJM method is the recognition that the Lagrangian solution contains sufficient information about the flow to constrain our measure of mesh smoothness.

In [50] the author considered three grids. At the beginning of the computational cycle, the grid of the previous cycle is considered from the previous step and used as an initial grid. This grid is assumed to be of "good" quality which means that it is unfolded and also well adapted to the "flow" at this time step. The second grid is the Lagrangian grid; this grid is obtained by simply moving each node of the initial grid with the Lagrangian fluid velocity. This Lagrangian grid is not necessarily optimal and may require "rezoning". The rezoning strategy aims at improving the quality of the Lagrangian grid while preserving the Lagrangian grid as much as possible. By requiring the rezoned grid to remain as close as possible to the Lagrangian grid, employing locals remappers is justified. Local remappers are computationally much more efficient than global remappers [12]. One more reason to maintain the rezoned grid close to the Lagrangian one is that, the Lagrangian grid contains important information about the flow; it follows for example the interface between two materials.

The rezoning strategy consists of two components: A sequence of local optimizations and a global optimization. The sequence of local optimizations aims at defining a set of "reference" Jacobians which incorporate the definition of mesh quality at each grid point. Actually, at each node a local patch is formed from the adjacent cells of the Lagrangian grid and a local realization of the Winslow smoothness functional [38] on this patch is constructed. The minimization of this functional with respect to the position of the central node defines its "virtual" location. By connecting this "virtually" moved node to its stationary neighbors, a reference Jacobian (RJM) that represents the best locally achievable geometric grid quality is defined.

A local realization of the Winslow function is used to determine the new virtual position of the central node. The optimized position is obtained by minimizing the following functional [2] with two variables while keeping all other coordinates fixed at their Lagragian positions.

$$F(x, y) = \int_0^1 \int_0^1 \frac{[(x_\xi)^2 + (y_\xi)^2] + [(x_\eta)^2 + (y_\eta)^2]}{|J|} d\xi d\eta$$

where

$$J = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix}$$

is the Jacobian matrix of map $x(\xi, \eta)$, $y(\xi, \eta)$, and $|J|$is its determinant.

It can be noticed that this functional has a barrier function, which means that the value of the functional approaches inifinity when the unknown map approaches any degenerate map where $|J| = 0$.

The RJMs are constructed by connecting these virtual positions.

It is not possible to use directly the RJMs themselves to determine the new positions of the vertices. Actually, each vertex belongs to four cells: a simple rezone strategy applied individually to each cell will lead in general to four incompatible specifications of the rezoned position of any vertex. For this, a "global" optimization step is performed in order to resolve incompatibilities of the locally defined RJMs. Actually, the final rezoned grid results from a minimization of a global objective function that measures the distance between the RJMs and the Jacobian of the rezoned grid (the solution of the global optimization step). This function does not guarantee that the rezoned grid is unfolded, for this, a "barrier" function that penalizes grids whose cells are inverted is included.

$$F(x(\xi,\eta), y(\xi,\eta)) = \int_0^1 \int_0^1 \frac{\left\|J - J_{ref}\right\|_F^2}{|J|/\left|J_{ref}\right|} d\xi d\eta$$

where $\|.\|_F$ is the Euclidean norm of the matrix, $|.|$the determinant of the matrix, J the Jacobian of the rezoned grid and J_{ref} the Jacobian of the rezoned grid. The term $\frac{|J|}{\left|J_{ref}\right|}$ in the denominator is the barrier function used to penalize degenerated solutions.

This function is minimized using a line search procedure coupled with a conjugate gradient algorithm. The optimization procedure is iterative and every time that it results in a folded mesh, the mesh is rejected and the objective function is set to a large number.In [50] the method was used for structured grids but it can also be used for unstructured grids.

1.5 Interpolation methods

Interpolation methods treat mesh deformation as a problem of interpolating deformations between the moving body and the boundaries of the domain of calculation. These methods are able to handle large deformations and arbitrary mesh topologies and can be easily implemented in parallel.

1.5.1 Radial Basis Function (RBF)

One main interpolating method is the Radial Basis Function (RBF).The RBF method [51, 14]consists in interpolating the displacement of the moving body to all the flow mesh using an interpolation function:

$$s(x) = \sum_{j=1}^{n_b} \alpha_j \phi(||X - X_{b_j}||) + p(X) \tag{10}$$

where $X_{b_j} = [x_{b_j}, y_{b_j}, z_{b_j}]$ are the centers in which the values are known, in this case the boundary nodes of the moving body, p a polynomial, n_b the number of boundary nodes and ϕ is a given basis function with respect to the Euclidean distance $||X||$.

The coefficients $\{\alpha\}$ and the polynomial $p(X)$ are chosen to satisfy the n_b fitting conditions:

$s(X_{b_j}) = d_{b_j}$

with d_b containing the discrete known values of the displacements at the boundary nodes and the constraints :

$$\sum_{j=1}^{n_b} \alpha_j q(X_{b_j}) = 0$$

for all polynomials q with a degree less or equal than that of polynomial p.
The values of the coefficients and the linear polynomial can be obtained by solving the system:

$$\begin{bmatrix} d_b \\ 0 \end{bmatrix} = \begin{bmatrix} M_{b,b} & P_b \\ P_b^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

with α the coefficients α_j, β the coefficients of the linear polynomial p, $M_{b,b}$ an $n_b \times n_b$ matrix containing the evaluation of the basis fucntion $\phi_{b_i b_j} = \phi(\|X_{b_i} - X_{bj_i}\|)$.

Solving this system can be done in an iterative manner. RBF is an efficient method to achieve high quality mesh and to handle large mesh deformations caused by translations and rotations for both 2D and 3D meshes. However, the performance depends on the used RBF[37].

The main drawback of this method is that the $n_b \times n_b$ matrix is often dense and ill-conditioned and solving it especially for large meshes is expensive due to the large number of surface points.

This method can be optimized using specialized preconditioners, however, they add more complexity to the implementation of the method.

Some other methods were developed to vastly reduce the number of surface points used, such as the Greedy point selection used in [33]. The method uses a chosen error function on the surface mesh to select a reduced subset of surface points; this subset contains a sufficiently small number of points so as to make the volume deformation fast.

1.5.2 Inverse Distance Weighting interpolation method (IDW)

One other interesting method is based on the Inverse Distance Weighting interpolation method (IDW) . The IDW [45] is a weighted average interpolation technique for multivariate interpolation of scattered data points.

The interpolated value is an average of the known values at the data points weighted by the inverse of the distance to the unsampled point. For mesh deformation, that means that the influence of boundary node displacements on the displacement of an internal mesh node is inversely proportional to the distance between the two points.

The moving boundaries nodal displacement field is defined as $s(x_{b_i})$, where x_{b_i} is the position vector of the boundary node i. Then, the displacement field s in the volume mesh is then described through a weighted average of all moving body boundary nodes displacement field as given by:

$$s(x) = \frac{\sum_{i=1}^{m} w_i(x) s(x_{b_i})}{\sum_{i=1}^{m} w_i(x)} \tag{11}$$

where m is the total number of boundary nodes and $w_i(x)$ is a two-exponent weighting function of the reciprocal distance

$$w_i(x) = A_i \left[\left(\frac{L_{ref}}{\|x - x_{b_i}\|} \right)^a + \left(\frac{\alpha L_{ref}}{\|x - x_{b_i}\|} \right)^b \right],$$

where A_i is the average area of all moving boundary faces containing the node i, this is used so that the mesh refinement of a region does not increase its influence in the interpolation method, L_{ref} is the distance between the mesh centroid and the farthest point of the domain, a is the basic IDW exponent, α is the maximal distance that nodes near to the moving body have to a boundary, and b is the IDW exponent which rigidifies nodes in a near-body region. α is given by :

$\alpha = \frac{5}{L_{ref}} \max_{i=1}^{m} \|s(x_{b_i}) - \overline{s}\|$, where $\overline{s} = \sum_{i=1}^{m} .s(x_{b_i})$ and $\overline{A_i} = \frac{A_i}{\sum_{j=1}^{m} A_j}$

The weighting function is designed such that it preserves a rigid body motion of nodes close to the boundaries, in addition to ensuring a smooth deformation transition through the mesh.

In order to increase the robustness and lower computational cost, this method was coupled in [47] to the so-called Moving Submesh Approach (MSA) proposed in [24]. It consists in interpolating the displacement to the fine computational mesh from the solved motion of a coarse mesh which is obtained by the IDW method.

In contrast to RBF interpolation, IDW method is an explicit mesh deformation method which does not require solving a system of equations for deforming the volume mesh. In fact, it results in an algebraic expression for the nodes displacements as function of the boundary deformation. This explicit evaluation reduces the computational costs and simplifies the implementation and the parallelization of the mesh deformation routines and can handle translations, rotations and arbitrary mesh topologies.

2 Post-treatment using local remeshing operations

While moving, the quality of the mesh decreases. Rather than generating a new mesh, some local remeshing operations can be performed in order to maintain a good mesh quality.

The standard mesh motion algorithms are able to move the mesh points. Then some kind of mesh smoothing can be applied to the mesh in order to maintain a reasonable mesh quality. Using this kind of operations, the topology of the mesh is not modified, which means that the underlying graph of the mesh remains unchanged. However, this approach fails for even simple motions such as large rigid body translation and rotation. Moreover, although the mesh topology may be preserved for simple mesh motions, such a procedure gives little control of the mesh size, hence, some cells may be compressed or stretched undesirably due to the mesh motion and smoothing, which results in a large error in the solution.

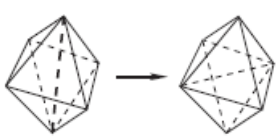
H-adaptation methods can be used to improve the quality of the mesh by modifying its topology[8].

They consist in locally coarsening or refining the mesh by the inclusion or deletion of mesh nodes or in changing the connectivities between the existing nodes. But, since these operations change the topology of the mesh, once performed they have to be followed by an interpolation step.

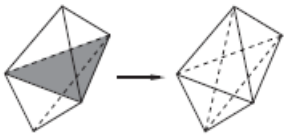
If no attention is paid, local remeshing operations may invalidate the mesh. Consequently, adequate checks of the mesh quality have to be performed. In [39], after moving the mesh swap operations are performed to optimize the mesh.

Edge swap operation (Figure3a) plays an important role in improving the mesh quality and acts only on the connectivities of the mesh leaving the vertices' positions unchanged. In 2D, edge

swapping is a simple operation which consists in flipping an edge shared by two triangles. In 2D, this operation changes neither the number of triangles nor the number of edges of the mesh. Face swapping (Figure3b) consists in suppressing face F of a tetrahedron K.

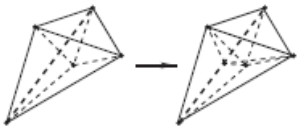


(a) edge swap operation [7]

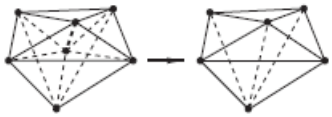


(b) face swap operation [7]

In [9] the authors used, in addition to the swapping operations, edge collapse operation (Figure3d) which is used to remove one vertex from the mesh. Collapsing an edge pq consists in merging its two end points to a single one. They also used edge split (Figure3c) that consists in splitting one edge pq by introducting a new point m in the mesh.



(c) edge split operation [7]



(d) edge collapse operation [7]

Usually, swap operators (edge swaps and face swaps) aim at improving locally the quality of the elements. Splits and collapses are there to control the mesh size: long edges are split whereas one of the two vertices of short edges is collapsed.

These operations are more efficient than the fixed-topology optimization methods in maintaining good mesh quality but, once used, local interpolations steps have to be performed. These interpolations are time consuming and induce extra errors.

3 Resolution of equations in moving mesh framework

A fundamentally important consideration when developing a computer code for simulating time-dependent physical or mechanical problems occurring in evolving domains is the choice of an appropriate kinematical description of the continuum. Actually, this choice determines the relationship between the deforming continuum and the mesh. In the case of FSI problems two main classes of methods are usually used depending on whether an implicit or explicit representation of the considered evolving domain is used.

As parts of the first class, Eulerian methods use a fixed mesh of a computational box. A Level-Set function is used according to which the evolving domain is defined in the sense of implicit function.

The so-called level set approach enables to represent the interface between two domains as a zero level of a smooth function [15]. In practice, a signed distance function is used to localize the interface. Let Ω_{i1}, Ω_{i2} and Γ_i represent, respectively, the first domain, the second domain and the interface. They verify:

$$\Omega_{i1} \cup \Omega_{i2} = \Omega \quad \text{and } \Omega_{i1} \cap \Omega_{i2} = \Gamma_i$$

For each node of the computational domain Ω, the level set function α which is the signed distance from the interface reads:

$$\alpha(x) = \begin{cases} > 0 & x \in \Omega_{i1} \\ 0 & x \in \Gamma_i \\ < 0 & x \in \Omega_{i2} \end{cases}$$

Thanks to this implicit representation, the domain evolution can easily be accounted using the following advection equation[23]:

$$\frac{\partial \alpha}{\partial t} + v.\nabla \alpha = 0$$

$$\alpha(t = 0, x) = \alpha_0(x)$$

In this case, any modification of the mesh that aims at following the movement of the moving body has to be followed by an interpolation step.

In our case, we use the so-called "immersed Volume method". This approach is of a great flexibility since it allows us to solve one set of equations in the whole domain. It consists in generating one unique volume mesh in which the different parts of the complex geometry are immersed and treated as different materials using Level-Set function and an anisotropic mesh adaptation at the interface between two materials. In order to ensure a continuous variation of the material properties across an interface, the so-called mixing laws are used [34].

On the opposite, Arbitrary-Lagrangian-Eulerian (ALE) methods [11] use an explicit discretization of the considered evolving domain. So, an exact and conformal mesh to the domain has to be generated.

In the ALE framework the nodes of the domain can move with the continuum and their velocity is directly integrated in the conservation equations. In fact, when the mesh is deformed while keeping its topology fixed, the equations are solved in a fully ALE manner so no interpolation step is needed. But, due to the fixed-topology constraint imposed by the classical Arbitrary-Lagrangian-Eulerian formulation, once the connectivity of the mesh changes between two time steps, local or global interpolation operations have to be performed in order to get the solution on the new mesh. In order to get rid of these interpolation operations, some attempts to get a true changing topology ALE scheme have been done, but only in 2D [39].

In [9] the authors tried to combine the advantages of both methods; on the one hand they have considered a conformal mesh to the interface (so an explicit representation of the interface) when it comes to solving the mechanical problem and on the other hand, they considered an implicit representation of the evolving geometry when it comes to compute its evolution using the Level-set function described above.

Conclusion and perspectives

We tried in this report to give a brief bibliographic overview of the main existing moving mesh methods.

We started our study with the mechanical analogy approach which includes mainly the spring and the elastic methods. In both approaches, the stiffness of the springs or elastic material is varied locally based on mesh cell size or proximity to the moving body. The choice of the parameters of these methods is left to the user and is usually based on ad hoc rezoning about specific quality objectives. These methods cannot handle all types of meshes and can have problems especially when high aspect ratio meshes are used. They may also require many sub-iterations to complete the full deformation of a mechanical time step which makes them time-consuming. These schemes are also typically difficult to implement.

We presented in the following paragraph the adaptive MMPDE-based methods that can also be divided into two groups: the velocity-based and the location-based methods. One main difference between the two methods is that with the velocity-based methods the adaptive mesh is the result of time integration of the mesh velocity fields. Thus, the mesh history for previous time steps will influence the mesh behavior at the current instant and the mesh can become highly skew. For the location-based methods , the adaptive mesh at a given time is determined by the monitor function at that time.

We also presented an "original" method called the LMPDE method. This method is quite interesting but, as mentioned before, it is still not clear how it may be extented to the use of unstructured meshes.

An optimization method was also explained. The main idea is to generate at each time step a mesh as close as possible to the Lagrangian mesh but with higher quality. This method is flexible, generates high quality meshes and is also able to handle all types of meshes. One main drawback is that it is time-consuming.

Finally, we presented two interpolation methods that treat the problem of mesh deformation as a problem of interpolating the deformation between the moving geometry and the boundaries of the computational domain. The RBF method seems to be an interesting method in terms of robustness. The problem is that it is a quite costly method in terms of CPU time especially, when compared to the IDW method. This latter seems to be a very efficient method. Actually, it is an explicit mesh deformation method that does not require solving a system of equations. This explicit formulation reduces the computational cost and makes the parallelization easier. This method is also robust especially when coupled to some mesh smoothing methods.

As a perspective, since the IDW seems to be an efficient method, it will be implemented and tested and probably coupled to some smoothing or local remeshing operations. In the case of using some local remeshing operations, the use of a conservative interpolation step is also required [Phd Thesis Chahrazade BAHBAH].

References

[1] T. J. Baker. Adaptive modification for time evolving meshes. *Journal of Materials Science*, 38(20):4175–4182, 2003.

[2] Jeremiah U Brackbill and Jeff S Saltzman. Adaptive zoning for singular problems in two dimensions. *Journal of Computational Physics*, 46(3):342 368, 1982.

[3] Weiming Cao, Weizhang Huang, and Robert D Russell. A study of monitor functions for two-dimensional adaptive mesh generation. *SIAM Journal on Scientific Computing*, 20(6):1978–1994, 1999.

[4] Weiming Cao, Weizhang Huang, and Robert D. Russell. A moving mesh method based on the geometric conservation law. *SIAM Journal on Scientific Computing*, 24(1):118–142, 2002.

[5] Neil N Carlson and Keith Miller. Design and application of a gradient-weighted moving finite element code i: in one dimension. *SIAM Journal on Scientific Computing*, 19(3):728–765, 1998.

[6] Neil N Carlson and Keith Miller. Design and application of a gradient-weighted moving finite element code ii: in two dimensions. *SIAM Journal on Scientific Computing*, 19(3):766–798, 1998.

[7] Gaëtan Compere, Jean-François Remacle, Johan Jansson, and Johan Hoffman. A mesh adaptation framework for dealing with large deforming meshes. *International journal for numerical methods in engineering*, 82(7):843–867, 2010.

[8] Gaetan Compère, Jean-François Remacle, Johan Jansson, and Johan Hoffman. A mesh adaptation framework for dealing with large deforming meshes. *International Journal for Numerical Methods in Engineering*, 82(7):843–867, 2010.

[9] C. Dapogny, C. Dobrzynski, and P. Frey. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *Journal of Computational Physics*, 262:358 – 378, 2014.

[10] Alain Dervieux, Youssef Mesri, Frédéric Alauzet, Adrien Loseille, Laurent Hascoët, and Bruno Koobus. Continuous mesh adaptation models for cfd. *CFD Journal*, 12(3), 2008.

[11] Jean Donea, Antonio Huerta, J.-Ph. Ponthot, and A. Rodríguez-Ferran. *Arbitrary Lagrangian Eulerian Methods*. John Wiley and Sons, Ltd, 2004.

[12] John K Dukowicz and John W Kodis. Accurate conservative remapping (rezoning) for arbitrary lagrangian-eulerian computations. *SIAM Journal on Scientific and Statistical Computing*, 8(3):305 321, 1987.

[13] Arkady S Dvinsky. Adaptive grid generation from harmonic maps on riemannian manifolds. *Journal of Computational Physics*, 95(2):450–476, 1991.

[14] Nail A. Gumerov and Ramani Duraiswami. Fast radial basis function interpolation via preconditioned krylov iteration. *SIAM J. Sci. Comput.*, 29(5):1876–1899, September 2007.

[15] E. Hachem, T. Kloczko, H. Digonnet, and T. Coupez. Stabilized finite element solution to handle complex heat and fluid flows in industrial furnaces using the immersed volume method. *International Journal for Numerical Methods in Fluids*, 68(1):99–121, 2012.

[16] Elie Hachem, S Feghali, Thierry Coupez, and Ramon Codina. A three-field stabilized finite element method for fluid-structure interaction: elastic solid and rigid body limit. *International Journal for Numerical Methods in Engineering*, 104(7):566–584, 2015.

[17] Elie Hachem, Ghina Jannoun, Jérémy Veysset, and Thierry Coupez. On the stabilized finite element method for steady convection-dominated problems with anisotropic mesh adaptation. *Applied Mathematics and Computation*, 232:581–594, 2014.

[18] Elie Hachem, Ghina Jannoun, Jeremy Veysset, Marc Henri, Romuald Pierrot, Isabelle Poitrault, Elisabeth Massoni, and Thierry Coupez. Modeling of heat transfer and turbulent flows inside industrial furnaces. *Simulation Modelling Practice and Theory*, 30:35–53, 2013.

[19] Weizhang Huang. Variational mesh adaptation: isotropy and equidistribution. *Journal of Computational Physics*, 174(2):903–924, 2001.

[20] Weizhang Huang. Mathematical principles of anisotropic mesh adaptation. *Commun. Comput. Phys*, 1(2):276–310, 2006.

[21] Weizhang Huang and Robert D Russell. A high dimensional moving mesh strategy. *Applied Numerical Mathematics*, 26(1):63–76, 1998.

[22] Ghina Jannoun, Elie Hachem, Jerôme Veysset, and Thierry Coupez. Anisotropic meshing with time-stepping control for unsteady convection-dominated problems. *Applied Mathematical Modelling*, 39(7):1899–1916, 2015.

[23] Mehdi Khalloufi, Youssef Mesri, Rudy Valette, Elisabeth Massoni, and Elie Hachem. High fidelity anisotropic adaptive variational multiscale method for multiphase flows with surface tension. *Computer Methods in Applied Mechanics and Engineering*, 307:44–67, 2016.

[24] E Lefrançois. A simple mesh deformation technique for fluid-structure interaction based on a submesh approach. *International Journal for Numerical Methods in Engineering*, 75(9):1085–1101, 2008.

[25] Guojun Liao, Feng Liu, C Gary, Danping Peng, and Stanley Osher. Level-set-based deformation methods for adaptive grids. *Journal of Computational Physics*, 159(1):103–122, 2000.

[26] Y Mesri, H Digonnet, and T Coupez. Hierarchical adaptive multi-mesh partitioning algorithm on heterogeneous systems. In *Parallel Computational Fluid Dynamics 2008*, pages 299–306. Springer, 2010.

[27] Youssef Mesri. *Gestion et contrôle des maillages non structurés anisotropes: applications en aérodynamique*. PhD thesis, École doctorale Sciences fondamentales et appliquées (Nice), 2007.

[28] Youssef Mesri, Hervé Guillard, and Thierry Coupez. Automatic coarsening of three dimensional anisotropic unstructured meshes for multigrid applications. *Applied Mathematics and Computation*, 218(21):10500–10519, 2012.

[29] Youssef Mesri, Mehdi Khalloufi, and Elie Hachem. On optimal simplicial 3d meshes for minimizing the hessian-based errors. *Applied Numerical Mathematics*, 109:235–249, 2016.

[30] Keith Miller. Moving finite elements. II. *SIAM Journal on Numerical Analysis*, 18(6):1033–1057, 1981.

[31] Keith Miller and Robert N Miller. Moving finite elements. I. *SIAM Journal on Numerical Analysis*, 18(6):1019–1032, 1981.

[32] Benjamin Ong, Robert Russell, and Steven Ruuth. An hr moving mesh method for one-dimensional time-dependent pdes. In *Proceedings of the 21st International Meshing Roundtable*, pages 39–54. Springer, 2013.

[33] T.C.S. Rendall and C.B. Allen. Reduced surface point selection options for efficient mesh deformation using radial basis functions. *Journal of Computational Physics*, 229(8):2810 – 2820, 2010.

[34] Quentin Schmid, E Marchal, Jean-Michel Franchet, Julien Schwartz, Youssef Mesri, and Elie Hachem. Unified formulation for modeling heat and fluid flow in complex real industrial equipment. *Computers & Fluids*, 134:146–156, 2016.

[35] K Stein, T Tezduyar, and R Benney. Mesh moving techniques for fluid-structure interactions with large displacements. *Journal of Applied Mechanics*, 70(1):58–63, 2003.

[36] Zhijun Tan. Adaptive moving mesh methods for two-dimensional resistive magneto-hydrodynamic pde models. *Computers & fluids*, 36(4):758–771, 2007.

[37] **A. de Boer and M.S. van der Schoot and H. Bijl**. **Mesh deformation based on radial basis function interpolation**. *Computers and Structures*, 85(11):784 795, 2007. Fourth {MIT} Conference on Computational Fluid and Solid Mechanics.

[38] **Alan M Winslow**. **Numerical solution of the quasilinear poisson equation in a nonuniform triangle mesh**. *Journal of Computational Physics*, 1(2):149 – 172, 1966.

[39] **Alauzet, F. and Olivier, G. A New Changing-Topology ALE Scheme for Moving Mesh Unsteady Simulations**. *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 68(1):99–121, 2011.

[40] **Alauzet, Frédéric**. **A changing-topology moving mesh technique for large displacements**. *Engineering with Computers*, 30(2):175–200, 2014.

[41] **Batina, John T. Unsteady Euler airfoil solutions using unstructured dynamic meshes**. *AIAA Journal*, 28(8):1381–1388, August 1990.

[42] **Brackbill, JU**. **An adaptive grid with directional control**. *Journal of Computational Physics*, 108(1):38–50, 1993.

[43] **C. Farhat and C. Degand and B. Koobus and M. Lesoinne**. **Torsional springs for two-dimensional dynamic unstructured fluid meshes**. *Computer Methods in Applied Mechanics and Engineering*, 163(1):231 – 245, 1998.

[44] **Cao, Weiming and Huang, Weizhang and Russell, Robert D**. **Approaches for generating moving adaptive meshes: location versus velocity**. *Applied Numerical Mathematics*, 47(2):121–138, 2003.

[45] **Edward Luke and Eric Collins and Eric Blades**. **A fast mesh deformation method using explicit interpolation**. *Journal of Computational Physics*, 231(2):586 – 601, 2012.

[46] **Huang, Weizhang and Russell, Robert D**. **Moving mesh strategy based on a gradient flow equation for two-dimensional problems**. *SIAM Journal on Scientific Computing*, 20(3):998–1015, 1998.

[47] **Jonathan Landry and Azzeddine Soulaimani and Edward Luke and Amine Ben Haj Ali**. **Robust moving mesh algorithms for hybrid stretched meshes: Application to moving boundaries problems**. *Journal of Computational Physics*, 326:691 721, 2016.

[48] **Li, Ruo and Tang, Tao and Zhang, Pingwen**. **Moving mesh methods in multiple dimensions based on harmonic maps**. *Journal of Computational Physics*, 170(2):562–588, 2001.

[49] **Ngo, Cuong and Huang, Weizhang**. **A Study on Moving Mesh Finite Element Solution of the Porous Medium Equation**. *arXiv preprint arXiv:1605.03570*, 2016.

[50] **Patrick Knupp and Len G. Margolin and Mikhail Shashkov**. **Reference Jacobian Optimization-Based Rezone Strategies for Arbitrary Lagrangian Eulerian Methods**. *Journal of Computational Physics*, 176(1):93 – 128, 2002.

[51] **T.C.S. Rendall and C.B. Allen**. **Efficient mesh motion using radial basis functions with data reduction algorithms**. *Journal of Computational Physics*, 228(17):6231 – 6249, 2009.

[52] T.E. Tezduyar, S. Mittal, S.E. Ray, and R. Shih. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering*, 95(2):221 – 242, 1992.

[53] PD Thomas and CK Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA journal*, 17(10):1030–1037, 1979.

[54] John G Trulio and Kenneth R Trigger. Numerical solution of the one-dimensional hydrodynamic equations in an arbitrary time-dependent coordinate system. *University of California Lawrence Radiation Laboratory Report UCLR-6522*, 1961.

[55] Alan M Winslow. Adaptive-mesh zoning by the equipotential method. Technical report, Lawrence Livermore National Lab., CA (USA), 1981.