Proceedings of the 7th World Congress on Momentum, Heat and Mass Transfer (MHMT'22)
~~Lisbon, Portugal~~ Virtual Conference – April 07 – 09, 2022
Paper No. ENFHT 212
DOI: 10.11159/enfht22.212

Moving Grid Generation: An Unstructured FEM for Simulating Moving Body

Saeed Rafiei[1], Ebrahim Khajehpour[2]
[1]Department of Mechanical Engineering, Shahid Chamran University of Ahvaz
Ahvaz, Iran
s.rafiee322@gmail.com
[2]Department of Mechanical Engineering, Islamic Azad University, Tehran South Branch
Tehran, Iran
st_e_khajehpour@azad.ac.ir

Abstract **-** The present work aims to introduce a method of unstructured moving grid generation with an emphasis on the capability of simulating moving body flow problems and being produced automatically at each time step according to the position of the body, which arises in many engineering fields involving multiphase flows, including sedimentation, improvement of combustion, or reducing the erosion by small droplets in large turbines. The proposed technique is based on the finite element method. The generated grid was applied in the case of moving a tetrahedron body in the computational domain. In this regard, the information of the grid was mapped from each time step to the next time step with the help of an algorithm of data interpolation. Additionally, to reduce the calculations, a two-block grid was generated in which total remeshing took place only in the areas close to the moving body, and in more remote areas, grid regeneration was performed only in the required areas.

Keywords**:** FSI, Moving body, Unstructured grid generation, Finite element, Data structure, Total remeshing

1. Introduction

Regarding computational moving grids, numerous and varied range of applications involving moving boundaries and interfaces can be considered, the most important of which can be categorized as follows: fluid-structure interactions, fluid-particle interactions, multiple phases, and free-surfaces. Early researches in this context were conducted on solving one-dimensional systems of equations on a moving grid [1]. Another approach for such one space dimension problems, is the method of lines (MOL) that packages based on this method are usually used in the automatic numerical integration of time-dependent problems in partial differential equations (PDEs) [2]. Later researches proved that adopting finite element methods not only provide acceptable results for application to dominated transport equations on one-dimensional examples, e.g., tracking transient shocks through the domain, but also it would work in higher dimensions [3]. Another prominent application of moving grids is related to the simulation of the motion of bodies in fluid that fixed grids cannot answer the real physics of such problems. And so, require the use of an appropriate moving computational grid [4]. A supplementary method, designed to handle certain classes of flow problems with moving boundaries and interfaces is a Shear-Slip Mesh Update along with a thin layer of deforming space-time elements and limited remeshing [5]. Considering the general case of moving or deforming surfaces, two types of grids are most commonly used: body-conforming and embedded. The main characteristic of body conforming grids is that in these grids, the external mesh faces match up with the surfaces of the domain; whereas in embedded grids, a large mesh, encompasses the surface, and elements close to the surfaces are treated in a special way [6]. Another practical method, for example, has been proposed by Liu et al. [7] that is using a Delaunay graph of the original mesh and the movement of the grid is conducted using this Delaunay graph. While there are varied approaches that use moving grids to handle moving bodies, we can also use fixed-grid techniques which are based on Cartesian coordinates [8]. There are also meshfree methods for simulating moving bodies, as an example we can refer to paper of Wang et al. [9]. Alauzet [10] introduced a new changing-topology moving mesh algorithm that based the mesh deformation algorithm on a linear elasticity system in order to reduce the number of required solutions. Overlapping grid methodology is another approach to handle moving grid problems that provides the capability of grid resolution using spectral interpolation [11]. A good research, in which some test cases of moving grids were presented, was

carried out by Krause and Kummer [12]. They proposed a high-order immersed boundary method (IBM), based on a Galerkin discretization that used a sharp interface technique to define bodies.

As it was said, the problem of moving bodies in fluids, as a part of moving grid problems, is the case in many engineering fields involving multiphase flows, including sedimentation, improvement of combustion, or reducing the erosion by small droplets in large turbines. To this end, in this article we focused on a method of unstructured moving grid generation that has the ability to simulate rotations of moving bodies, especially tetrahedron bodies in the flow field based on finite element method (FEM), which provides the capability of making elements dense in the unstructured grid. In addition, it can be easily adapted to moving bodies.

Accordingly, first, the method of generating a fixed computational grid will be presented and next, the chosen algorithm for moving the grid will be described.

2. Computational Moving Grid Generation

2.1. Computational Grid Generation

The unstructured grid generation starts using an initial grid. In this method, the computational domain is first divided into several primary regions. Then the nodes are placed in each region separately. Grid's data is stored in the data structure based on the nodes and grid elements distinctly. This speeds up the search for needed grid data and speeds up this process. Another feature of this method is that it is automatic; so that the data structure is built for grid elements and nodes simultaneously and separately.

The generated grid has quadrilateral elements. The principles used in the production of this grid are based on the ability to provide general movement of a body in the flow field. That means the goal is to design a grid that has the ability to be generated automatically in all modes of transitional or rotational motion of a body with a small and large range of motion. Figure 1 shows the computational grid generated in this way.

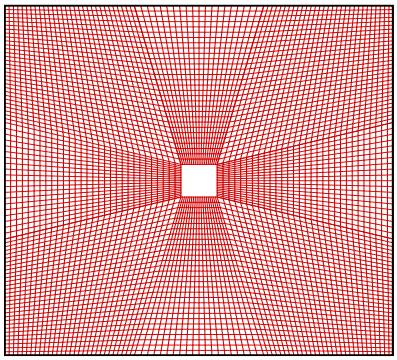


Fig.1: Computational grid around a body with 6000 quadrilateral elements

2.2. Data Structure

In order to have access to the grid information at any moment, this information is stored in the data structure. The data structure should be able to express the connection of grid elements correctly to have a high speed for searching grid information. The data structure used in this research is based on the elements of the grid. This data structure stores the data of the nodes that make up each element for all the elements in the grid.

In order to create a data structure, elements and nodes of the grid must first be numbered. This numbering is used to form the data structure. Data structure information is stored in matrices. In the next section, each of these matrices is examined separately.

2.3. Element Connectivity Matrix

Due to the selection of quadrilateral elements, in the element connectivity matrix, four columns are needed to store the number of nodes adjacent to each element. The order of information of each element must have a certain

arrangement and this order must be observed for all elements in the grid. It is necessary to apply a contract in how to number nodes adjacent to each element.

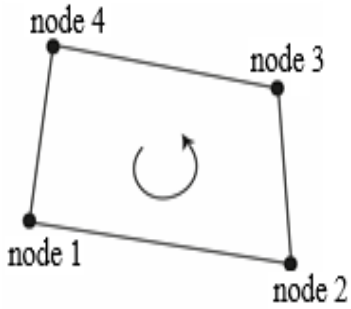


Fig. 2: Contract of the numbering of nodes associated with each element in the element connectivity matrix

According to the contract used in this study, the first node of each element is located in the lower left corner of the element. The number of each element is stored in the first column and the number of the first node in the second column of the connectivity matrix. Once the first node is specified, the other three nodes are stored in the next three columns of the connectivity matrix, respectively, with respect to the counterclockwise rotation in the element. This is done by respectively naming the nodes with node 1, node 2, node 3 and node 4 as an example in Figure 2. The resulted matrix is shown in Table 1 in the following.

2.4. Node Connectivity Matrix

In order to store node information, the numbers of all nodes along with their coordinates are stored in a matrix. In another matrix, the number of boundary nodes is stored along with their boundary conditions and initial values.

The contract used in this section is as follows: The first column is for storing the boundary node number; the four columns are for storing the horizontal and vertical velocity conditions, the temperature and pressure of each node, and four other columns are used to store the initial values of horizontal and vertical velocities, temperature and pressure of the nodes. Resulted matrices are presented as Table 2 in the following, respectively.

2.5. Clustering

Accuracy in calculating flow characteristics and variables such as velocity and pressure around a moving body requires a grid of finer elements around the body. Therefore, clustering is performed only in certain areas (i.e. around the moving body). The reason for this is to avoid high grid volume, especially in areas with low gradients of flow variables. This is done after generating the initial computational grid shown earlier.

2.6. Computational Moving Grid

Next, the computational domain is divided into several regions (blocks) and a separate grid is generated in each block. The generated computational moving grid is a two-block grid. In the first region, which includes areas close to the body, total remeshing method will be used. But in the second region, which covers more remote areas, regenerating the grid will be carried out only by local remeshing.

As the body moves in the flow field, at each time step a computational grid is generated in block 1 around the body. But the grid regeneration in block 2 will only be needed when the body is close enough to the boundaries of block 1 that the desired quality of the elements in this block is lost. If the body approaches the boundaries of block 1 more than the allowable value, this block will move horizontally or vertically. Therefore, the elements of block 2 are removed only in a row from one side of the separating boundary (between the two blocks) and added to the grid at the same time on the other side.

Another noteworthy point in the method used is that the number of elements in the whole computational domain will be constant over time and thus prevent the increase of calculations over time.

2.7. Communicating Updates

Due to the grid generation method used to ensure the movement of the body in the fluid, the information of the grid nodes must be calculated correctly at each time step. But the location of each node has changed at each time step compared to the previous time. Therefore, needed information must be interpolated between the old grid and the new grid in two consecutive time steps.

More precisely speaking, it is assumed that there is a new grid with nodes located at $x(t_{n+1})$, and also before that, $u(x(t_n), t_n)$ which are the values of the arbitrary variable of the flow in the nodes of the grid $x(t_n)$ at the previous time, i.e. t_n, have been calculated. Now, the goal is to interpolate the variable domain on the nodes of the new grid $x(t_{n+1})$ to obtain $u(x(t_{n+1}), t_{n+1})$. Here, t_n and t_{n+1} are two consecutive time steps.

After finding the element containing each node, using finite element approximations, the values of the required variables in each node of the new grid are calculated.

Information interpolation is possible by solving a set of nonlinear algebraic equations. This is shown in the following equations by showing the values of the new grid with the new subtitle and the values of the old grid with the old subtitle.

Using interpolator functions, the coordinate value of x, for example, at any point inside the element is as follows,

$$x(\zeta,\eta) = \frac{X_1}{4}(1-\zeta)(1-\eta) + X_2(1+\zeta)(1-\eta) + X_3(1+\zeta)(1+\eta) + X_4(1-\zeta)(1+\eta) \tag{1}$$

where, X is the horizontal coordinate of the four nodes of the element.

If (1) is written for both the X and Y coordinates of each node in the new grid, a non-linear equation system is obtained according to (2) where X_{new} and Y_{new} are known; and, ζ and η are the unknowns of the equation system.

$$\begin{aligned} X_{new}(\zeta,\eta) &= [C_1\zeta + C_2\eta + C_3\zeta\eta + C_4]_{old} \\ Y_{new}(\zeta,\eta) &= \left[\acute{C}_1\zeta + \acute{C}_2\eta + \acute{C}_3\zeta\eta + \acute{C}_4\right]_{old} \end{aligned} \tag{2}$$

The constants used in (2) are defined as follows.

$$\begin{aligned} C_1 &= \frac{1}{4}[-X_1 + X_2 + X_3 - X_4], \qquad C_2 = \frac{1}{4}[-X_1 - X_2 + X_3 + X_4] \\ C_3 &= \frac{1}{4}[X_1 - X_2 + X_3 - X_4], \qquad C_4 = \frac{1}{4}[X_1 + X_2 + X_3 + X_4] \end{aligned} \tag{3}$$

To solve this nonlinear system, Newton Method is used. For this purpose, the two functions $f(\zeta,\eta)$ and $g(\zeta,\eta)$ are considered as follows.

$$\begin{aligned} f(\zeta,\eta) &= [C_1\zeta + C_2\eta + C_3\zeta\eta + C_4]_{old} - X_{new} = 0 \\ g(\zeta,\eta) &= \left[\acute{C}_1\zeta + \acute{C}_2\eta + \acute{C}_3\zeta\eta + \acute{C}_4\right]_{old} - Y_{new} = 0 \end{aligned} \tag{4}$$

By writing these two functions as a Taylor series and intercepting it, the following equation system is finally obtained.

$$\begin{bmatrix} f_\zeta(\zeta_i,\eta_i) & f_\eta(\zeta_i,\eta_i) \\ g_\zeta(\zeta_i,\eta_i) & g_\eta(\zeta_i,\eta_i) \end{bmatrix} \begin{bmatrix} \Delta\zeta_i \\ \Delta\eta_i \end{bmatrix} = -\begin{bmatrix} f(\zeta_i,\eta_i) \\ g(\zeta_i,\eta_i) \end{bmatrix}, \qquad \begin{bmatrix} \zeta_{i+1} \\ \eta_{i+1} \end{bmatrix} = \begin{bmatrix} \zeta_i \\ \eta_i \end{bmatrix} + \begin{bmatrix} \Delta\zeta_i \\ \Delta\eta_i \end{bmatrix} \tag{5}$$

Equation system (5) is solved by the Gaussian elimination method by establishing conditions $-1 \leq \zeta \leq 1$ and $-1 \leq \eta \leq 1$ at the same time. By solving this equation system, the natural coordinates (ζ and η) of the node in the new grid relative to the element found in the old grid will be determined. The important point in the process is that first the element to be interpolated must be found correctly. Because only then can the values of natural coordinates be obtained obtained correctly and within the allowed range.

2.8. Computational Grid Convergence

To investigate the convergence of the computational grid, the simulation of the flow around the body in a case where the density of the body is 1.004 times the density of the fluid has been performed. The velocity of the body in the vertical direction was selected as the comparison criterion. The results of this comparison are shown in Figure 3. The Prandtl number in all results is 1, the Rayleigh number is 10,000 and the time step is 0.0005. According to the results of comparing the dimensions of the grid, a grid with 6272 elements has been used in the calculations.

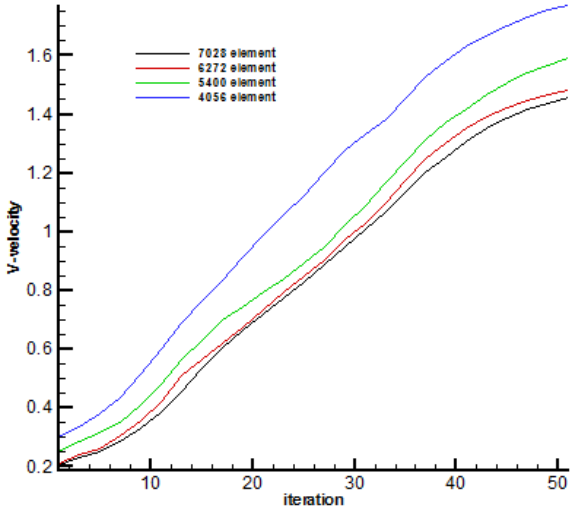


Fig. 3: Vertical velocity of the body in Rayleigh 10,000

3. Results

As mentioned in the previous section, in order to store grid information in the data structure, it was necessary to produce separate connectivity matrices and store the information separately in them. It was also said that a contract has to be applied in how to number nodes adjacent to each element.

Part of the connectivity matrix of the elements related to the generated grid is shown in Table 1.

Table 1: Element connectivity matrix

Element number	Node 1	Node 2	Node 3	Node 4
1	1	2	83	82
2	2	3	84	83
3	3	4	85	84
4	4	5	86	85

In the case of node connectivity matrix, as it was said, related information was stored in two separate matrices. The first stores node numbers and their coordinates. The second matrix, as said, was related to the boundary nodes, which is shown in Table 2 as part of this matrix.

In Table 2, the number 0 indicates Dirichlet conditions and the number 1 indicates Neumann conditions on the boundary.

Table 2: Boundary conditions and initial values

Node number	Boundary conditions				Initial values			
	Horizontal velocity	Vertical velocity	Pressure	Temperature	Horizontal velocity	Vertical velocity	Pressure	Temperature
10	1	1	0	1	0.0	0.0	0.0	0.5
11	1	1	0	1	0.0	0.0	0.0	0.5
12	1	1	0	1	0.0	0.0	0.0	0.5

As mentioned in the previous section, in order to avoid high grid volume in areas with low gradients of flow variables and also accuracy of flow characteristics and variables around the moving body, after generating the initial computational grid (shown in Figure 1), clustering was performed. The resulting grid is shown in Figure 4.

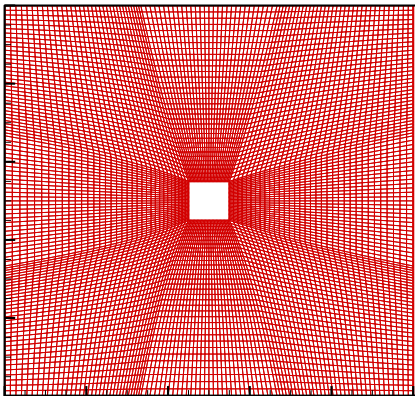


Fig. 4: Computational grid after clustering around the body

In the next step, we divided the computational domain into two blocks. As it was described, in the first region, total remeshing method was used, and the second region, regenerating the grid was carried out only by local remeshing. Figure 5 shows an overview of the two-block grid.

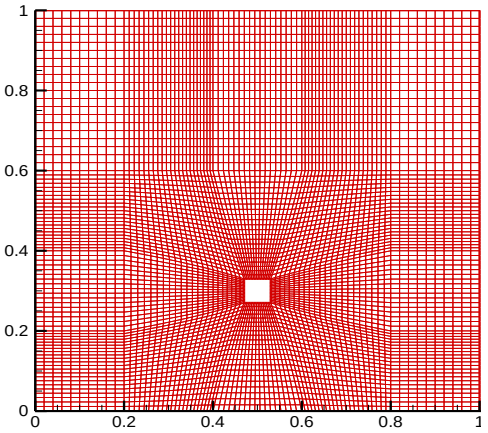


Fig. 5: Overview of two-block computational grid

It can also be observed the two blocks of the computational grid distinguished by a dashed line in Figure 6.

Finally, the motion of the body was performed in the computational grid, the result of which can be seen in the figures for different positions.

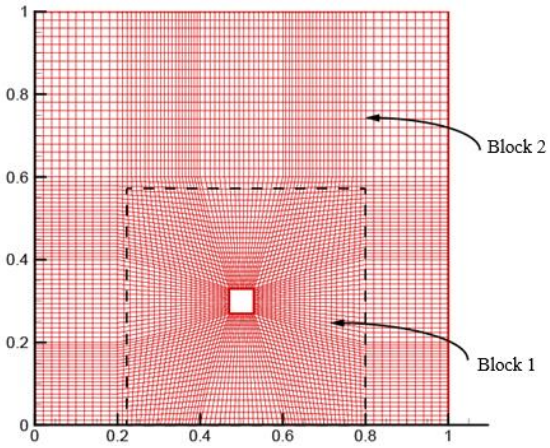
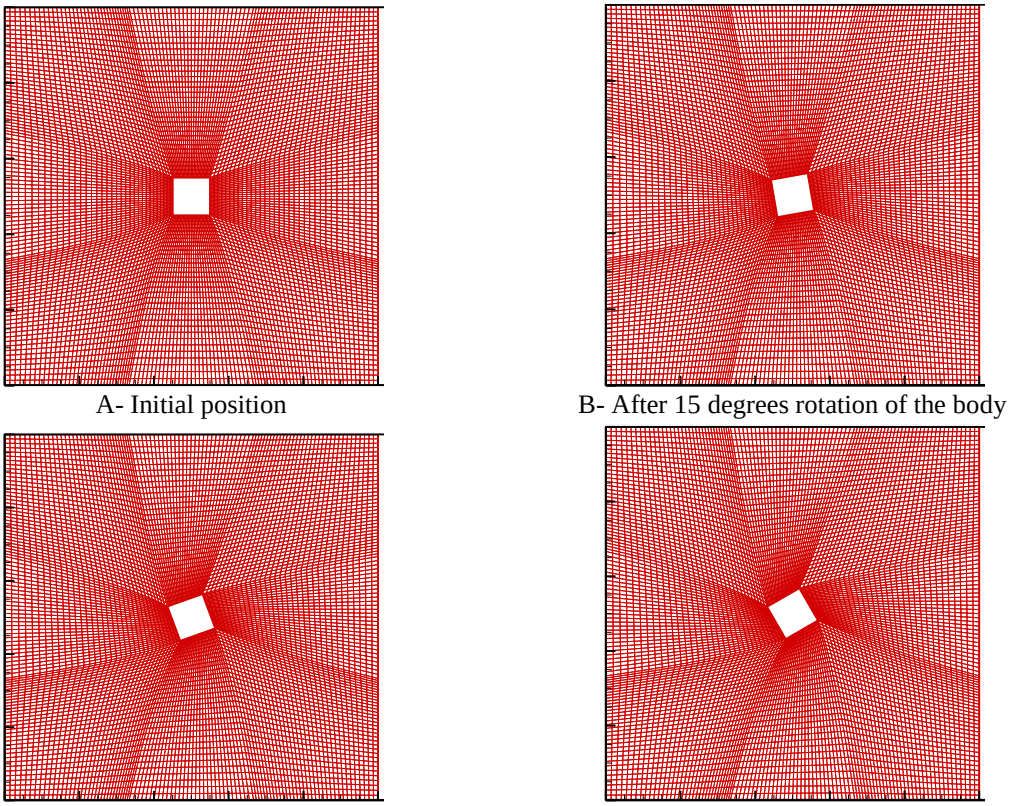


Fig. 6: The two blocks of the computational grid

In Figure 7, the generated computational grid is displayed in four tetrahedron body orientation positions. As it can be observed, the proposed technique could be able to simulate rotation of the body. In each of the shown positions, the grid is generated automatically according to the position of the body.



A- Initial position

B- After 15 degrees rotation of the body

C- After 20 degrees rotation of the body

D- After 30 degrees rotation of the body

Fig. 7: Computational grid around the body in four different positions

4. Conclusion

As seen, in problems with a moving boundary or with moving bodies in a fluid, especially in cases where a body moves under the influence of an unsteady flow and its path is not prescribed, a fixed computational grid does not have the ability to simulate the motion of the body in a flow field. Because in such cases, the location of the body changes at any time and accordingly, the grid around the body will also need to change.

There are many methods for moving grids. Each of these methods is produced according to the type of movement of domain boundaries. Of course, knowing how to solve the flow will also have a great impact on choosing the method of moving the grid or modeling it. As mentioned, the grid and method required for low-amplitude movements may be different from the grid and the method required for high-amplitude movements. In choosing the method for moving the grid, various factors are influential, including the type of boundary movement, the geometry of the problem and the type of grid. In this study:

1. The method selected in the unstructured computational grid generation was such that it was able to cover moving bodies in the flow field and be produced automatically at each time step according to the position of the body.
2. Grid information was stored at each step in the data structure designed for this purpose.
3. It was seen that with the movement of the body and as a result, the movement of the grid, the location of the nodes and hence the storage location of the flow variables changed. Therefore, in order to map the information from each time step to the next time step, an algorithm for data interpolation was designed.
4. To reduce the calculations, a two-block grid was generated. In this grid, only in the areas close to the moving body, total remeshing took place during the movement. In more remote areas, grid regeneration is done only in the required areas.
5. In order to increase the accuracy of calculations of flow variables in areas close to the body, clustering was performed around the body.

References

[1] J. G. Verwer, J. G. Blom, J. M. Sanz-Serna, An Adaptive Moving Grid Method for One-Dimensional Systems of Partial Differential Equations, Journal of Computational Physics 82, 454-486 (1989).

[2] R. M. Furzeland, J. G. Verwer, P. A. Zegeling, A Numerical Study of Three Moving-Grid Methods for One-Dimensional Partial Differential Equations Which Are Based on the Method of Lines, Journal of Computatioal Physics 89, 349-388 (1990).

[3] Bill Semper and Guojun Liao, A Moving Grid Finite-Element Method Using Grid Deformation, Numerical Methods for Partial Differential Equations, 11, 603-615 (1995).

[4] A. A. Johnson, T. E. Tezduyar, Simulation of multiple spheres falling in a liquid-filled tube, Computer Methods in Applied Mechanics and Engineering 134 (1996) 351-373.

[5] M. Behr, T. Tezduyar, The Shear-Slip Mesh Update Method, Computer Methods in Applied Mechanics and Engineering 174 (1999) 261-274.

[6] Rainald Löhner, Juan Cebral, Marcelo Castro, Joseph D. Baum, Hong Luo, Eric Mestreau, and Orlando Soto, ADAPTIVE EMBEDDED UNSTRUCTURED GRID METHODS, Mećanica Computacional Vol. XXIII, pp. 29-42 (2004).

[7] Xueqiang Liu, Ning Qin, Hao Xia, Fast dynamic grid deformation based on Delaunay graph mapping, Journal of Computational Physics 211 (2006) 405–423.

[8] Pengzhi Lin, A fixed-grid model for simulation of a moving body in free surface flows, Computers & Fluids 36 (2007) 549–561.

[9] X.Y. Wang, P. Yu, K.S. Yeo, B.C. Khoo, SVD–GFD scheme to simulate complex moving body problems in 3D space, Journal of Computational Physics 229 (2010) 2314–2338.

[10] Frédéric Alauzet, A changing-topology moving mesh technique for large displacements, Engineering with Computers volume 30, pages175–200(2014).

[11] Brandon Merrill, Yulia Peet, High-Order Moving Overlapping Grid Methodology for Aerospace Applications, 53rd AIAA Aerospace Sciences Meeting, January 5 - 9 2015, Kissimmee, FL

[12] Dennis Krause, Florian Kummer, An Incompressible Immersed Boundary Solver for Moving Body Flows using a Cut Cell Discontinuous Galerkin Method, Computers and Fluids, Volume 153, 10 August 2017, Pages 118-129.