The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

STABLE, EFFICIENT, AND HIGHER-ORDER MESH MOVEMENT IN LARGE-DISPLACEMENT FLUID-STRUCTURE INTERACTION

TERESA SCHWENTNER[1], DOMAGOJ BOŠNJAK[2] AND THOMAS-PETER FRIES[3]

Institute of Structural Analysis
Graz University of Technology
Lessingsstr. 25, 8010 Graz, Austria
www.ifb.tugraz.at
e-mail: [1]schwentner@tugraz.at, [2]bosnjak@tugraz.at, [3]fries@tugraz.at

Key words: Fluid-structure interaction, mesh generation, mesh update, moving domain

Summary. Mesh movement is a crucial aspect of simulations where the domain is time-dependent, e.g., in fluid-structure interactions (FSI). The fluid domain changes over time, and hence the mesh moves. The accuracy of the simulation highly depends on the quality of the mesh. Various different methods for mesh updates have been proposed. However, these methods are often not robust enough in case of higher-order meshes. Consequently, instead of *updating* the mesh, in this work a new mesh is *generated* in every time step, based on a block-structure consisting of valid coarse quads. Transfinite maps are used to generate sub-meshes within the block-structure, allowing any order and number of elements. The geometry information is assigned to the block edges by user-input, whereas the geometry on the FSI-interface is a result of the simulation itself. The remaining nodes of the block-structure are moved to generate valid meshes throughout the simulation. For moderate movements, these manipulations of the block-structure may be implemented based on case-dependent, empirical rules. However, larger displacements and more blocks in the block-structure require stable and efficient update schemes (instead of the resulting mesh in the simulation). Hence, different methods for this movement are compared. First, an optimization scheme is applied to the block-structure, optimizing the angles within the blocks without radically changing the position. This procedure is computationally efficient, as only the block nodes are optimized, not the entire mesh. As an alternative, a pseudo-solid approach is applied to the block-structure, where the displacement on the FSI-interface is prescribed as a boundary condition. Herein, the quality and applicability of these methods are compared.

1 INTRODUCTION

Fluid-structure interaction (FSI) problems are gaining more and more attention, for example due to the applications in biomechanics, e.g., cardiovascular systems [20, 19] or aerodynamics [2]. The simulation of such coupled problems, herein by using the finite element method (FEM), is notoriously difficult. One of the crucial aspects to consider is the time-dependent, moving mesh for the fluid domain. An invalid mesh would lead to an abortion of the whole simulation and poor mesh quality leads to poor results. Different methods have been proposed over the years. They

can be roughly categorized as physical analogy, interpolation or hybrid methods [17]. A physical analogy method that is commonly used is the pseudo-solid approach [25, 24]. The mesh is treated as a solid reacting to a displacement of the FSI-interface. This means a full FEM simulation is done to move the nodes of the simulation mesh. Others use the Laplace operator [16] or the bi-harmonic operator [11]. Interpolation methods are based on, e.g., the Delaunay interpolation [15] or radial basis functions [5]. Hybrid methods combine different approaches. The finite macro-element method [14] for structured meshes uses a subset of the nodes of the computational mesh. They are moved according to a pseudo-solid approach and transfinite interpolation is used to then move the remaining nodes of the computational mesh. Another similar approach [23] uses a combination of transfinite, inverse distance weighting or radial basis function interpolation for moving first the nodes of the block-structure and then the remaining nodes. However, none of the mentioned approaches cover higher-order elements. Updating a mesh that consists of higher-order elements is more difficult than updating a linear mesh. The robustness is reduced and often test case-dependent parameter adaptation is required. Furthermore, *updated* (rather than newly *generated*) higher-order meshes may often not be able to achieve results with optimal accuracy due to smoothness issues near re-entrant corners, see [22].

Consequently, we present a mesh regeneration approach based on a higher-order block-structured meshing scheme, where two different algorithms are compared to move the block-structure. The block-structure consists of valid but coarse quads, representing the topology. For the initial mesh, the geometry is assigned to the block edges, and based on this information, the nodes inside the blocks are generated using transfinite maps. In the following time steps, the geometry information for the FSI-interface is a direct result of the FSI-simulation. Since the FSI-interface has now moved, the block-structure has to move accordingly. For moderate movements empirical rules can be used, but for larger movement this may not lead to satisfactory results. To this end, two different approaches to move the block nodes are compared herein. As a first approach, an optimization scheme is used, which considers the angles between the block edges and the block edge length. The second scheme to move the block nodes uses a pseudo-solid approach based on the linear elastic material model.

The paper is organized as follows. In Section 2, the governing equations for the FSI-simulation are given. The higher-order block-structured mesh generation technique is presented in Section 3. Section 4 covers the two different approaches for the movement of the block vertices. Numerical results are given in Section 5 and conclusions are drawn in Section 6.

2 FLUID-STRUCTURE INTERACTION

For the coupled problem in the FSI-simulation, the computational domain is subdivided into a solid domain Ω_{S} and a fluid domain Ω_{F}. The interface of these two domains is denoted as Γ_{FSI}. A strongly coupled, partitioned scheme is used to solve the problem, only shortly described here for brevity.

For the fluid problem, the governing equations are given by the instationary, incompressible Navier–Stokes equations [12] in an Arbitrary Lagrangian–Eulerian (ALE) framework. The momentum and the continuity equations in the pressure-velocity formulation read

$$\varrho_{\mathrm{F}} \left(\dot{\boldsymbol{u}} + \overline{\boldsymbol{u}} \cdot \nabla_{\boldsymbol{x}} \boldsymbol{u} \right) - \nabla_{\boldsymbol{x}} \cdot \boldsymbol{\sigma} - \boldsymbol{f} = \boldsymbol{0} \quad \text{in} \quad \Omega_{\mathrm{F}}(t), \tag{1}$$

$$\nabla_{\boldsymbol{x}} \cdot \boldsymbol{u} = 0 \quad \text{in} \quad \Omega_{\mathrm{F}}(t). \tag{2}$$

ϱ_{F} is the fluid's density, $\boldsymbol{u}$ denotes the fluid velocity, where $\overline{\boldsymbol{u}}$ is the advective velocity calculated

by $\overline{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}_{\mathrm{M}}$, with the mesh velocity $\boldsymbol{u}_{\mathrm{M}}$. For a Newtonian fluid, the stress tensor $\boldsymbol{\sigma}$ is given by

$$\boldsymbol{\sigma}(\boldsymbol{u}, p) = -p\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}), \tag{3}$$

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla_{\boldsymbol{x}}\boldsymbol{u} + (\nabla_{\boldsymbol{x}}\boldsymbol{u})^{\mathrm{T}}\right), \tag{4}$$

with the fluid's pressure p and the dynamic viscosity μ.

The solid problem is formulated in a Lagrangian framework, always referring to the initial configuration. The governing equations for the St.Venant solid [1] are formulated as

$$\varrho_{\mathrm{S}}\ddot{\boldsymbol{d}} - \nabla_{\boldsymbol{X}} \cdot (\mathbf{F}\mathbf{S}) - \boldsymbol{f} = \mathbf{0} \qquad \text{in} \quad \Omega_{\mathrm{S}}^{0}, \tag{5}$$

with the solid's density ϱ_{S}, the displacements $\boldsymbol{d}$ and the volume forces $\boldsymbol{f}$. The deformation gradient $\mathbf{F}$ and the second Piola–Kirchhoff stress tensor $\mathbf{S}$ are defined as

$$\mathbf{F} = \mathbf{I} + \nabla_{\boldsymbol{X}}\boldsymbol{d}, \tag{6}$$

$$\mathbf{S} = \lambda\,(\mathrm{tr}\mathbf{E})\,\mathbf{I} + 2\eta\mathbf{E}, \tag{7}$$

respectively, where λ and η are the Lamé parameters, and the Green–Lagrange strain tensor is defined as

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^{\mathrm{T}}\mathbf{F} - \mathbf{I}\right). \tag{8}$$

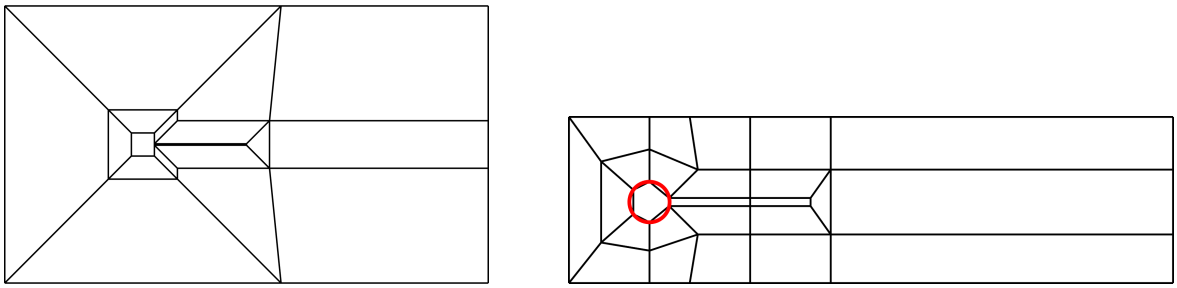
Geometrical consistency and mechanical equilibrium are necessary on the time-dependent FSI-interface. Further boundary conditions are omitted for brevity. The discretization in space is done by a finite element framework, using Taylor–Hood elements for the Navier-Stokes equations 1 and 2 to avoid any stabilization issues. For the temporal discretization, we use finite differences.

3 HIGHER-ORDER BLOCK-STRUCTURED MESH GENERATION

For the mesh generation approach applied herein, the usage of a block-structure is crucial. A constant number of elements and nodes throughout the whole simulation leads to simple tracking of the movement of each node. Then, the mesh velocity, required for the advective velocity in Eq. 1, can be calculated easily. Therefore, *no projection* between the meshes is required, as the ALE formulation allows the consideration of moving nodes naturally.

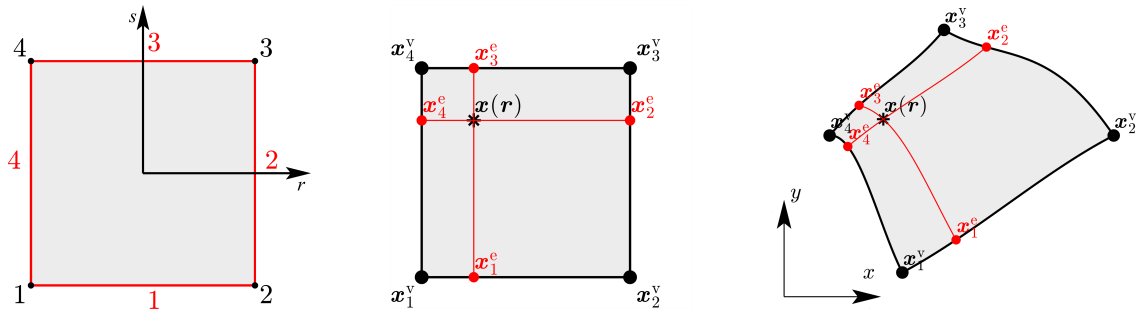
As a first step, the block-structure is defined, representing the topology. Herein, conforming quads are used for the blocks. Consequently, the generated block-structure is a linear and valid mesh, albeit coarse, see Fig. 1 for examples. As user-input, the coordinates of the block nodes and the connectivity matrix of the block-structure are required. Next, the initial geometry may be assigned to the block edges. This can be done by implicit and explicit definition, e.g., higher-order elements, NURBS, function evaluations, or isolines based on the level-set method [7, 8, 18]. In the following time steps, the geometry description on the FSI-interface is a direct result of the solid deformation. All remaining geometry descriptions from the initial mesh are again applied to the block-structure.

On each block edge, the number of elements n_i and order p has to be defined. The same order of the elements is required for the whole mesh and the same number of elements is required on



(a) (b)

Figure 1: Definition of some block-structures: (a) for the test case presented in Section 5 and (b) for some similar test case including a curved boundary.



(a) (b) (c)

Figure 2: (a) The reference block with numbered edges and vertices. (b) The points required to find the position of $\boldsymbol{x}(\boldsymbol{r})$. (c) The block and the position $\boldsymbol{x}(\boldsymbol{r})$ in the physical space.

opposite block edges. This leads to $n_{\mathrm{edge},i} = n_i \cdot p + 1$ nodes on the respective block edge and thus $n_{\mathrm{block}} = n_{\mathrm{edge},i} \cdot n_{\mathrm{edge},j}$. Next, transfinite maps are used to determine the position of the nodes inside the blocks, based on the position of the edge nodes. The general concept is based on the *blending-function method* [9, 10, 4] and was already outlined in [21] for two-dimensional cases and in [3] for three-dimensional cases. A reference block in the (r, s)-coordinate system, with numbered vertices and edges, can be seen in Fig. 2a. Then, corresponding bi-linear shape functions are defined as

$$N_1(\boldsymbol{r}) = \frac{1}{4}(1-r)\cdot(1-s), \quad N_2(\boldsymbol{r}) = \frac{1}{4}(1+r)\cdot(1-s),$$

$$N_3(\boldsymbol{r}) = \frac{1}{4}(1+r)\cdot(1+s), \quad N_4(\boldsymbol{r}) = \frac{1}{4}(1-r)\cdot(1+s).$$

Note, that these shape functions are not related to the resulting shape functions for the (higher-order) elements of the computational mesh. Additionally, a ramp function is defined for each edge

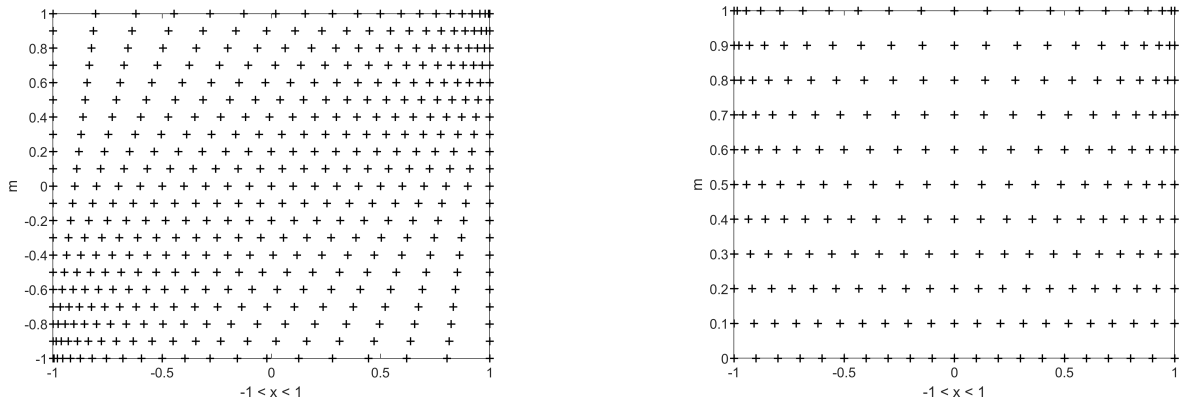
$$R_1 = N_1 + N_2, \quad R_2 = N_2 + N_3, \quad R_3 = N_3 + N_4, \quad R_4 = N_4 + N_1,$$

noting that the function value is 1 along the corresponding edge. For a point $\boldsymbol{r} = (r, s)^T$, the

position $\boldsymbol{x}(\boldsymbol{r})$ is then defined with the transfinite map

$$\boldsymbol{x}\left(\boldsymbol{r}\right) = \sum_{i=1}^{4} R_i\left(\boldsymbol{r}\right) \cdot \boldsymbol{x}_i^{\mathrm{e}} - \sum_{i=1}^{4} N_i\left(\boldsymbol{r}\right) \cdot \boldsymbol{x}_i^{\mathrm{v}},$$

where $\boldsymbol{x}_i^{\mathrm{e}}$ are the related coordinates of the edge points, and $\boldsymbol{x}_i^{\mathrm{v}}$ are the related coordinates of the vertices, as can also be seen in Fig. 2b. An interpretation in the physical coordinate system is also possible, see Fig. 2c. The nodes in the block interiors are then generated based on the nodes on the edges of the blocks, via transfinite interpolation. Next, a connectivity matrix associates the nodes to the desired (higher-order) elements. Combining the resulting sub-meshes leads to the computational mesh. This approach allows structured generation of any number of elements and any order of elements in the computational domain.



(a) (b)

Figure 3: The two different examples of the node distribution in the domain $[-1, 1]$; (a) shows the quadratic function with the scaling parameter $m \in \langle -1, 1\rangle$ and (b) shows the cubic function with the scaling parameter $m \in [0, 1\rangle$. For $m = 0$, no grading is applied.

For non-smooth solution properties, such as the case of boundary layers in fluid mechanics, or re-entrant corners in solid mechanics, one needs to consider the local mesh properties. This is most often resolved by adding completely new thin layers of elements to the mesh, thereby additionally refining the mesh in the area of interest with the goal of improving performance. Naturally, this procedure implies a higher element count, to accommodate for the aforementioned issues. On the other hand, one might consider *anisotropic mesh refinement*, where a non-uniform spatial resolution is induced from the start, producing smaller or thinner elements where it is necessary without increasing the amount of elements. Herein, we opt for the second approach, achieved by *mesh grading* on the block-structure level.

Specifically, instead of placing a uniform number of points on each edge of the block-structure, we consider a different node distribution. Fig. 3 illustrates reference examples of distributing nodes in the interval $[-1, 1]$, and shows two different, parametrized options of a non-uniform node distribution function. The quadratic function favors the left or the right side of the interval, whereas the cubic function produces a symmetric non-uniformity. Such an effect can be analogously achieved on any edge of the block-structure, which is then carried over to the final

node positioning by the transfinite maps. Naturally, if the given edge of the block-structure is assigned a geometry description, the resulting non-uniform nodes have to be properly mapped to match it.

This specific technique of creating the mesh enables the movement of the block-structure and tracking of the position of each node individually. No projection between the meshes is required, and the mesh velocity can be calculated easily. The movement of the block-structure is another crucial aspect of the overall approach, which is covered in the following section.

4 MOVEMENT OF THE BLOCK-STRUCTURE

For the regeneration of the computational mesh in every time step, the geometry description on the FSI-interface, and hence the positions of the corresponding nodes, is given by the simulation results. Based on that new position of the FSI-interface, the remaining nodes of the block-structure have to be adjusted. For moderate displacements this can be done by case-dependent, empirical rules, but for more complicated movements or to avoid user-input, we rather opt for schemes that move the nodes fully automatically. Two different methods are compared herein. The first method uses an optimization scheme, and the second one is based on the classic and well-known pseudo-solid approach [25, 24]. A crucial aspect of both options is the fact that they are used on the block-structure only, which is clearly more efficient compared to using these methods on the whole computational mesh.

4.1 Movement of the block-structure with an optimization scheme

One of the main issues of many popular mesh movement methods is their inability to control the quality, and ultimately the validity of the resulting mesh. Thus, we consider an optimization-based approach, with the goal of preserving the mesh quality, thereby enabling simulations to run through without significant hyperparameter tuning. This approach should not radically modify the block-structure to be uniform or "ideal", but rather make minor changes in each time step, with a similar order of magnitude as the solid displacement. For computational efficiency reasons, the optimization targets the block-structure, since it is significantly coarser than the simulation mesh.

The primary objective of the optimization scheme is the preservation of adequate angles within the mesh. To this end, we define a local error estimation for a given point $\boldsymbol{x}$ of the block-structure. We consider the immediate neighbours $\boldsymbol{a}_i$ and the remaining element neighbours $\boldsymbol{b}_i$, as shown in Fig. 4.

To locally optimize the position of the point $\boldsymbol{x}$, we consider the error function

$$e_D(\alpha) = |\alpha - D|^n, \quad n \in \mathbb{N} \text{ even},$$

where D denotes the ideal angle at the associated point, e.g., 90° for quadrilateral elements. The measured angles include all but one angle in each element to which $\boldsymbol{x}$ belongs. Following Fig. 4, the angles are given by

$$\alpha_i = \alpha_i(\boldsymbol{x}) = \angle(\boldsymbol{a}_{i+1} - \boldsymbol{x},\, \boldsymbol{a}_i - \boldsymbol{x}), \quad i = 1, \dots, m_1,$$

$$\alpha_i = \alpha_i(\boldsymbol{x}) = \angle(\boldsymbol{b}_i - \boldsymbol{a}_i,\, \boldsymbol{x} - \boldsymbol{a}_i), \quad i = m_1 + 1, \dots, m.$$

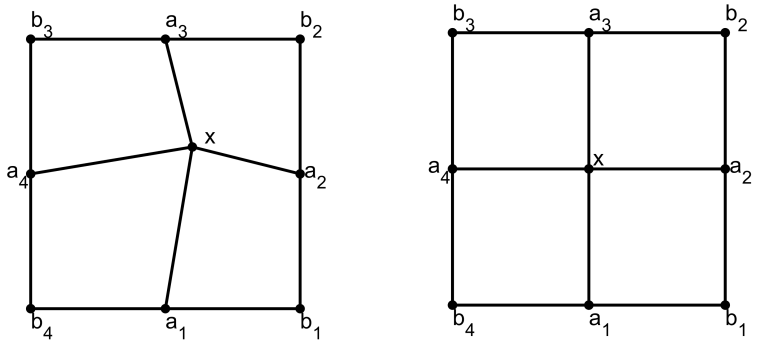


Figure 4: The local optimization scheme used to update the position of the point $\boldsymbol{x}$ based on the nodes of all elements containing $\boldsymbol{x}$: (a) the unoptimized position, and (b) the optimized position.

This yields m angle error functions

$$f_i(\boldsymbol{x}) = e_D(\alpha_i(\boldsymbol{x})), \quad i = 1, \dots, m.$$

The number of neighbours m varies depending on how many elements share the point $\boldsymbol{x}$. For the computation of the angle, we make use of the *atan*2 function, as the computation of the angle via the *acos* function may lead to possible numerical issues when performing gradient-based optimization. Overall, each function f_i has well-defined and continuous derivatives.

Optimizing using only this error function may lead to edge lengths tending to 0, since the function exclusively considers angles. Thus, an edge length penalization is necessary. Each edge is assigned an upper limit U_j and a lower limit L_j, either uniformly or locally, for example based on the initial block-structure. The upper limit is optional, whereas the lower limit is required. Following the same notation, the penalization of the edge length is given as

$$g_j(\boldsymbol{x};\, \boldsymbol{a}_j) = g_j(\boldsymbol{x}) = \begin{cases} C\big(\|\boldsymbol{x} - \boldsymbol{a}_j\|^2 - L_j^2\big)^2, & L_j > \|\boldsymbol{x} - \boldsymbol{a}_j\| \\ 0, & L_j \leq \|\boldsymbol{x} - \boldsymbol{a}_j\| \leq U_j \\ C\big(U_j^2 - \|\boldsymbol{x} - \boldsymbol{a}_j\|^2\big)^2, & \|\boldsymbol{x} - \boldsymbol{a}_j\| > U_j \end{cases}, \quad j = 1, \dots, k$$

Again, it is quite clear that each function g_j has well-defined and easy-to-compute derivatives. The constant C serves as a balancing factor to bring both errors to the same order of magnitude. Thereafter, the functions are combined together to form the final function which is to be optimized

$$\boldsymbol{h}(\boldsymbol{x}) = \big[f_1\, f_2\, \dots\, f_m\, g_1\, g_2\, \dots\, g_k\big]\,.$$

In order to balance the local angle errors, thereby ensuring they do not grow too large and cause blocks to become invalid, the optimization problem is set up as a (nonlinear) least squares problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{h}(\boldsymbol{x})\|_2^2.$$

Here, we apply a typical Gauss-Newton approach. Denoting the differential $\mathbf{J} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}$, the iterative scheme in the step $n+1$ is given by

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}\,\boldsymbol{h}(\boldsymbol{x}^n).$$

The last consideration is a way to ensure that the optimization is reasonably sized compared to FSI-caused mesh movements. Namely, if the optimization severely affected the nodal positions, errors would arise due to the mesh velocity. Thus, we limit the norm of each optimization step by simply projecting the optimized node position to the circle around the original node, whose radius is given by the dynamic movement limit. Given a circle projection function $\mathbf{P}$, one may project in every step:

$$\boldsymbol{x}^{n+1} = \mathbf{P}\big(\boldsymbol{x}^n - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}\,\boldsymbol{h}(\boldsymbol{x}^n)\big),$$

or only project after the final optimization step:

$$\boldsymbol{x}^{\text{end}} = \mathbf{P}(\boldsymbol{x}^{\text{end}}),$$

where we mostly rely on the first approach, which is a direct Gauss–Newton equivalent of the projected gradient descent method.

4.2 Movement of the block-structure with a pseudo-solid approach

This method treats the block-structure as a solid, reacting to a certain displacement, causing the deformation of the FSI-interface. A linear elastic material model is used, with the standard equilibrium equation, the stress tensor $\boldsymbol{\sigma}$, and the strain tensor $\boldsymbol{\epsilon}$, defined as

$$\nabla \cdot \boldsymbol{\sigma} = \mathbf{0},$$

$$\boldsymbol{\sigma} = \lambda_{\text{P}} \cdot \text{tr}\,(\boldsymbol{\epsilon})\,\mathbf{I} + 2\mu_{\text{P}}\boldsymbol{\epsilon},$$

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right),$$

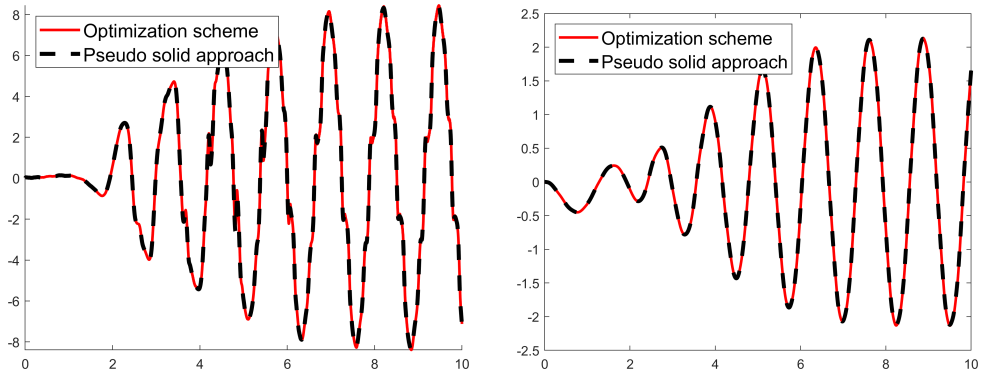
where λ_{P} and μ_{P} are the pseudo Lamé parameters. For the details regarding the weak formulation, we refer the reader to [24]. The elements (in this case the blocks) are often assigned different material properties, and thus different stiffnesses to avoid distorted elements especially in the boundary layers. For the pseudo Young's modulus of the block n we use herein $E_{\text{P}}^n = \left(\frac{A_n}{A_{\min}}\right)^m$, where A_n represents the area of the corresponding block and $A_{\min}$ the area of the smallest block. The choice of the exponent m is arbitrary, but usually $m \in [1, 3]$. It is noted that the fine-tuning of these pseudo-material parameters is sometimes cumbersome when applied to updating a full simulation mesh. Here, however, where the approach is applied to the coarse block-structure, the situation is much less sensitive. The boundary conditions on the FSI-interface are prescribed as the associated displacements, and the conditions on the rest of the boundary are zero displacements.

Usually, this approach is applied to the full mesh and a full FEM simulation is done for the movement of the individual nodes. In the present approach, the pseudo-solid is only applied to the block-structure and thus the system of equations that has to be solved is small in comparison, and the computation is fast. For higher-order elements, the pseudo-solid approach applied to the full mesh leads to invalid elements more frequently. When it is applied only to the blocks, which

are in fact very coarse and linear elements, the validity is preserved even for larger movements. The next step is then to apply the mesh generation on the updated block-structure, taking the geometry description from the initial mesh description, as well as the displacement of the FSI-interface into account.

5 NUMERICAL RESULTS

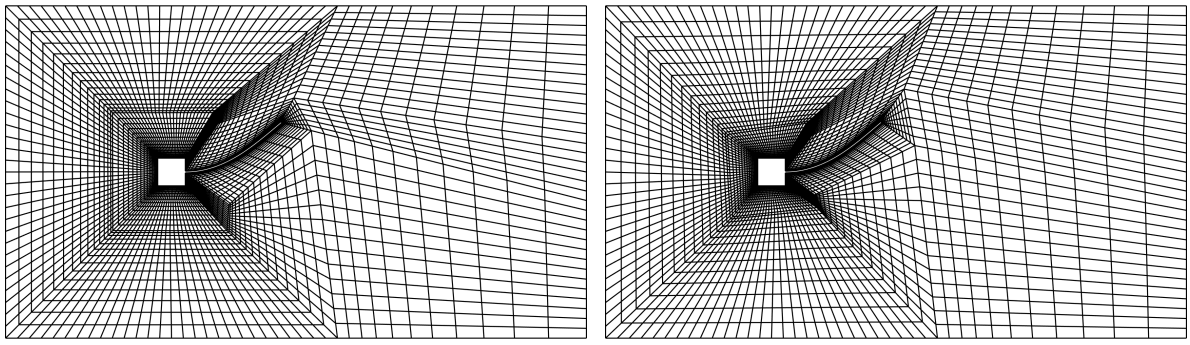
For comparison of the two approaches presented herein, we show a numerical test case. A full FSI-simulation is conducted, for the benchmark test case from Wall and Ramm [26, 13]. A flexible flap attached to a rigid square cylinder starts oscillating as it interacts with the fluid flow. For further information on the test case settings, we refer to [26, 13].



(a) (b)

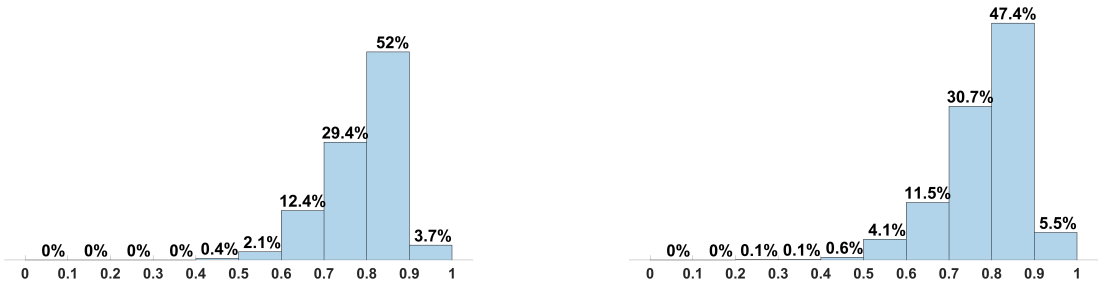
Figure 5: Comparison of the (a) vertical fluid force, and the (b) vertical displacement of the tip of the flap.

The vertical fluid force and displacement for both approaches are shown in Fig. 5, which are in good agreement with the literature. Nevertheless, for the optimization scheme some oscillations appear around some peaks of the fluid forces, which are in the range of 1 percent. The general behaviour and values are still met and no oscillations appear in the displacements. To compare the mesh quality, the fluid mesh after 1500 time steps (7.5 seconds) is considered, where the vertical displacement of the tip of the flap is close to its peak. The mesh for the optimization scheme is shown in Fig. 6a, and the mesh for the pseudo-solid approach is shown in Fig. 6b. The quality metrics used for the comparison are the element-wise *scaled Jacobian* and the element-wise *equiangular skewness* [6]. For the scaled Jacobian, the values may be in the range of $\langle -\infty, 1]$, where a negative value indicates an invalid element, and a value of 1 indicates an ideal element. The histograms of the scaled Jacobian for the aforementioned meshes are shown in Fig. 7. The equiangular skewness quality is defined as $S_\mathrm{e} = \max\{\frac{\theta_\mathrm{max}-\theta_\mathrm{ideal}}{180°-\theta_\mathrm{ideal}}, \frac{\theta_\mathrm{ideal}-\theta_\mathrm{min}}{\theta_\mathrm{ideal}}\}$, where θ_ideal is 90° for an ideal quad element and θ_max and θ_min denote the maximal and minimal angle within an element, respectively. The equiangular skewness is optimal for a value of 0. For the equiangular skewness, the histograms for both meshes are shown in Fig. 8. Both metrics suggest better mesh quality for the optimization approach. Nevertheless, the difference in the simulation results is marginal, as can be seen in Fig. 5, showing the good behaviour of both approaches.



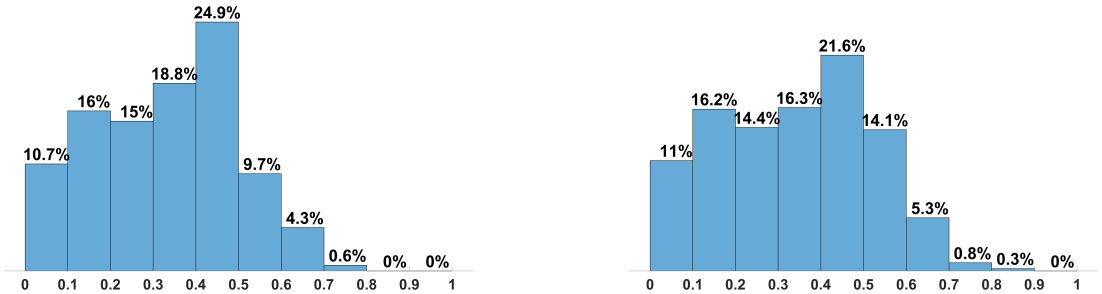
(a) (b)

Figure 6: The computational meshes after 1500 time steps (7.5 seconds) at the peak of the vertical displacement of the tip of the flap for (a) the optimization scheme and the (b) pseudo-solid approach.



(a) (b)

Figure 7: The scaled Jacobian for the meshes shown in (a) Fig. 6a and (b) Fig. 6b.



(a) (b)

Figure 8: The equiangular skewness for the meshes shown in (a) Fig. 6a and (b) Fig. 6b.

6 CONCLUSIONS

Within this work, we have presented a higher-order block-structured mesh regeneration approach based on an update of the block-structure. No projection between the meshes at different time-steps is required, as the structured approach allows the tracking of the node displacement and velocity required for the ALE formulation. Two different methods for the movement of the block-structure are compared. Both approaches lead to valid elements throughout the simulation. The first method based on an optimization scheme leads to higher mesh quality, whereas the second method uses the pseudo-solid approach, which is a well-established method in the context of mesh updates of full simulation meshes. A benefit of both approaches is computational efficiency, as they are only applied to the block-structure rather than the entire simulation mesh. They enable the generation of higher-order elements without the loss of element validity, and the structured approach avoids any projection requirements.

REFERENCES

[1] K.J. Bathe. *Finite Element Procedures*. Prentice-Hall, Englewood Cliffs, NJ, 1996.

[2] Y. Bazilevs, K. Takizawa, T.E. Tezduyar, M.C. Hsu, N. Kostov, and S. McIntyre. Aerodynamic and FSI analysis of wind turbines with the ALE-VMS and ST-VMS methods. *Archive Comp. Mech. Engrg.*, 21:359–398, 2014.

[3] D. Bošnjak, A. Pepe, R. Schussnig, D. Schmalstieg, and T.P. Fries. Higher-order block-structured hex meshing of tubular structures. *Eng Comput .*, pages 1–21, 2023.

[4] S.A. Coons. Surfaces for computer-aided design of space forms. MAC-TR-41, MIT, Cambridge, USA, 1967.

[5] A. de Boer, M.S. van der Schoot, and H. Bijl. Mesh deformation based on radial basis function interpolation. *Computers & Structures*, 85:784–795, 2007.

[6] G. De Santis, M. De Beule, K. Van Canneyt, P. Segers, P. Verdonck, and B. Verhegghe. Full-hexahedral structured meshing for image-based computational vascular modeling. *Med. Eng. Phys.*, 33:1318–1325, 2011.

[7] T.P. Fries and S. Omerović. Higher-order accurate integration of implicit geometries. *Internat. J. Numer. Methods Engrg.*, 106:323 – 371, 2016.

[8] T.P. Fries, S. Omerović, D. Schöllhammer, and J. Steidl. Higher-order meshing of implicit geometries–part I: Integration and interpolation in cut elements. *Comp. Methods Appl. Mech. Engrg.*, 313:759–784, 2017.

[9] W.J. Gordon and C.A. Hall. Construction of curvi-linear co-ordinate systems and applications to mesh generation. *Internat. J. Numer. Methods Engrg.*, 7:461 – 477, 1973.

[10] W.J. Gordon and C.A. Hall. Transfinite element methods: blending function interpolation over arbitrary curved element domains. *Numer. Math.*, 21:109 – 129, 1973.

[11] B.T. Helenbrook. Mesh deformation using the biharmonic operator. *Internat. J. Numer. Methods Engrg.*, 56:1007–1021, 2003.

[12] T.J.R. Hughes, W.K. Liu, and T.K. Zimmermann. Lagrangian-Eulerian finite element formulation for incompressible viscous flows. *Comp. Methods Appl. Mech. Engrg.*, 29:329 – 349, 1981.

[13] B. Hübner, E. Walhorn, and D. Dinkler. A monolithic approach to fluid-structure interaction using space-time finite elements. *Comp. Methods Appl. Mech. Engrg.*, 193:2087–2104, 2004.

[14] J.H. Ko, J.W. Kim, S.H. Park, and D. Byun. Aerodynamic analysis of flapping foils using volume grid deformation code. *J. Mech. Sci. Technol.*, 23:1727–1735, 2009.

[15] X. Liu, N. Qin, and H. Xia. Fast dynamic grid deformation based on Delaunay graph mapping. *J. Comput. Phys.*, 211:405–423, 2006.

[16] R. Löhner and C. Yang. Improved ALE mesh velocities for moving bodies. *Commun. Numer. Meth. Engrg.*, 12:599–608, 1996.

[17] E. Luke, E. Collins, and E. Blades. A fast mesh deformation method using explicit interpolation. *J. Comput. Phys.*, 231:586–601, 2012.

[18] S. Omerović and T.P. Fries. Conformal higher-order remeshing schemes for implicitly defined interface problems. *Internat. J. Numer. Methods Engrg.*, 109:761–912, 2017.

[19] R. Schussnig, D.R.Q. Pacheco, and T.P. Fries. Robust stabilised finite element solvers for generalised newtonian fluid flows. *J. Comput. Phys.*, 442:110436, 2021.

[20] R. Schussnig, D.R.Q. Pacheco, and T.P. Fries. Efficient split-step schemes for fluid-structure interaction involving incompressible generalised Newtonian flows. *Computers & Structures*, 260:106718, 2022.

[21] T. Schwentner and T.P. Fries. Fluid-structure interaction with fully coupled mesh generation. In *Proceedings in Apllied Mathematics and Mechanics*, volume 23, Chichester, 2023. John Wiley & Sons.

[22] T. Schwentner and T.P. Fries. Fully coupled, higher-order, block-structured mesh generation in fluid-structure interaction. submitted.

[23] S. Sen, G. De Nayer, and M. Breuer. A fast and robust hybrid method for block-structured mesh deformation with emphasis on FSI-LES applications. *Internat. J. Numer. Methods Engrg.*, 111:273–300, 2017.

[24] K. Stein, T. Tezduyar, and R. Benney. Mesh moving techniques for fluid-structure interactions with large displacements. *J. Appl. Mech., ASME*, 70:58–63, 2003.

[25] K. Stein, T.E. Tezduyar, and R. Benney. Automatic mesh update with the solid-extension mesh moving technique. *Comp. Methods Appl. Mech. Engrg.*, 193:2019–2032, 2004.

[26] W.A. Wall and E. Ramm. Fluid-structure interaction based upon a stabilized (ale) finite element method. In *4th World Congress on Computational Mechanics – New Trends and Applications*, CIMNE, Barcelona, Spain, 1998.