Seventh International Conference on Computational Fluid Dynamics (ICCFD7), Big Island, Hawaii, July 9-13, 2012

ICCFD7-1301

The Nitsche Method of the Navier-Stokes Equations for Immersed and Moving Boundaries

J. Benk*, M. Ulbrich** and M. Mehl***
Corresponding author: benk@in.tum.de

* Department of Computer Science, TU München, Germany.
** Department of Mathematics, TU München, Germany.
*** Institute for Advanced Study, TU München, Germany.

Abstract: The usual way for a flow simulation with complex boundaries is to generate an unstructured mesh that represents the boundary accurately. In this classical approach, the generation and handling of the mesh is a complex task and the finite element method (FEM) on unstructured meshes can generate significant computational overhead, especially in terms of memory requirements and access, an increasingly crucial aspect on massively parallel computing architectures. An alternative approach is the Immersed Boundary (IB) method. In this paper, we investigate the Nitsche Method for the Navier-Stokes equations (NSE) with immersed boundaries. This method avoids the usage of computationally costly unstructured meshes by using an adaptive Cartesian mesh instead. In contrast to unstructured meshes, Cartesian meshes can be partitioned in a load-balanced way without a central storage of the whole mesh information and are highly storage efficient even in a sequential context. For an accurate simulation of our scenarios, no-slip boundary conditions need to be imposed on complex boundary, that is not represented by the mesh facets. For this purpose, we employ the Nitsche Method to impose these conditions in a weak form. We extend the approach for moving boundaries, which enables us to compute fluid-structure interaction (FSI) scenarios. The results of various FSI benchmark scenarios, presented at the end of this paper, verify our approach.

Keywords: Nitsche Method, Navier-Stokes Equations, Immersed Boundary, Moving Boundary, Fluid Structure Interaction.

1 Introduction

The representation of complex boundaries and their boundary conditions (BC) poses several challenges in various applications. The application we consider here is a finite element (FEM) based 2D and 3D flow computation, where the fluid is modeled by the incompressible Navier-Stokes equations. In this case, the classical way to represent complex boundaries is the employment of unstructured meshes, where the boundary is represented by the mesh's facets. Generating an unstructured mesh induces a major computational and in particular storage overhead. Especially in parallel computations, the meshing and the partitioning of the computational domain into load-balanced domains on distributed memory systems is a potential bottleneck [1]. Obviously, the handling and the storage of such an unstructured mesh, capable to represent complex boundaries, is more costly than a structured one. Therefore, several methods have been developed for efficient an accurate representation of complex boundaries on structured meshes. One intuitive way is to use the facet cells of the structured mesh (e.g [2]). This leads to an $O(h)$ order boundary representation, where h denotes the mesh width. Especially in 3D, this first order representation proves to be inefficient, even if combined with mesh adaptivity, where the required mesh resolution for the boundary leads to too high cell numbers.

This property of structured meshes leads to methods that represent the geometry explicitly by a separate

entity. These methods are called Immersed Boundary (IB) Methods and use an explicit and independent representation of the boundary. With this approach, the question arises how to impose the BC on a geometry that is non conforming with the boundary. In these cases, the boundary geometry intersects the boundary cells in an arbitrary way. Thus, to impose a given BC on such intersected cells turns out to be challenging. [3] gives an overview of various IB methods in fluid dynamics and categorizes the methods in **discrete** and **continuous** approaches. A review of various IB methods is also included in [4].

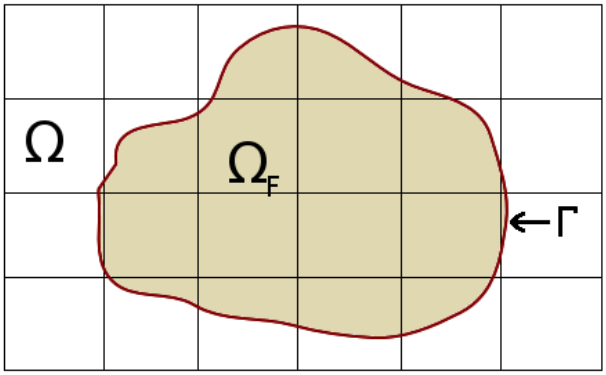

Figure 1: Illustration of an Immersed Boundary. The boundary is illustrated by the curve intersecting the cells of a regular Cartesian mesh. On such a boundary, we want to impose given Dirichlet BC. Ω represents the computational domain, whereas Ω_F denotes the fictitious domain.

In this paper, we apply the Nitsche Method to impose a Dirichlet BC for the NS equations on IBs. This method is formulated in a weak form as presented in Sect. 2. Hence, this IB method falls into the category of continuous IB approaches. This formulation of the Nitsche Method for the NS equations has already been used in [5, 6], but in both cases, the method was employed on the mesh's boundary facets, where a BC could be imposed in the classical way as well. We, however, use this formulation in an IB setting (see Fig. 1), where special cut-cell and boundary integrals are required for the cells that are intersected by the boundary geometry. Section 2 describes the Nitsche Method for fixed and immersed boundaries, the developed geometry representations, and the required cut-cell and boundary integrals in 2D and 3D. In Sect. 3, we extend our approach for moving boundaries within a transient flow field. Scenarios with severe geometry changes are challenging in particular for unstructured meshes, where costly remeshing of the domain might be required. For our approach, however, the mesh stays the same and fixed for the whole simulation, which is, thus, a crucial advantage. One of the important applications of moving boundaries in a flow field are simulations of fluid-structure interactions (FSI). In this case, the consistent imposition of BCs and boundary force computations play a crucial role. We introduce in Sect. 4 our approach to couple the fluid and the structure, where the Nitsche Method is used on the fluid side. In order to demonstrate the consistent imposition of Dirichlet BC on moving and immersed boundaries, we compute in Sect. 5 several FSI benchmark scenarios. The paper closes with a short summary and outlook in Sect. 6.

2 The Nitsche Method of the Navier-Stokes Equation on Immersed Boundaries

We start with the statement of the incompressible Navier-Stokes equations (NSE). The computational domain is denoted by $\Omega \subset \mathbb{R}^d$ with $d = 2, 3$, where the momentum equation and the continuity equation hold:

$$\rho^f \frac{\partial \mathbf{v}}{\partial t} - \nu^f \Delta \mathbf{v} + \rho^f (\mathbf{v} \cdot \nabla) \mathbf{v} + \nabla p = \mathbf{f} \ in\ \Omega, \tag{1}$$

$$\nabla \cdot \mathbf{v} = 0 \ in\ \Omega. \tag{2}$$

$$\mathbf{v} = \mathbf{g} \ on\ \Gamma. \tag{3}$$

ρ^f denotes the density of the fluid, whereas ν^f represents the kinematic viscosity. $\mathbf{f}(t) \in L^2(\Omega)^d$ represents external volume forces acting on the fluid. The flow field is determined by the velocity vector field $\mathbf{v}$ and the scalar pressure field p.

The problem, given by Eq. (1) and (2), is closed by Dirichlet boundary conditions (BC) for the velocity on $\Gamma := \partial\Omega$ or a combination of Dirichlet and Neumann-type BCs. We denote the Dirichlet data by $\mathbf{g}(t) \in H^{1/2}(\Gamma)^d$ Dirichlet BCs are in our focus on immersed boundaries such as the one illustrated in Fig. 1.

In the following, our goal is to present the Nitsche Method to impose Dirichlet BCs for the Stokes and Navier-Stokes equations. The first step towards the Nitsche formulation is to derive the weak-form of Eq. (1) and (2). We choose a suited function space for the velocity vector and for the pressure. For the components of $\mathbf{v} = (v_1, \ldots, v_d)$, we use the same discrete space. Since integration by parts is required during the derivation of the weak form, we assume that $\mathbf{v}(t) \in H^1(\Omega)^d$. The Sobolev space $H^1(\Omega)$ ensures the existence of weak derivatives. The pressure field can be an element of $L^2_0(\Omega)$ in space. Analogously to the unknown space, the test space for the velocity is chosen as $\psi \in H^1(\Omega)^d$ and for the pressure $\xi \in L^2_0(\Omega)$. We denote the $(d+1)$-dimensional test function of both equations by $\phi = (\psi, \xi)$. Equation (1) is tested with the velocity's test function ψ, whereas ξ tests Eq. (2). After these multiplications of the momentum equation Eq. (1) with ψ and the continuity equation Eq. (2) with ξ, the terms $\int_\Omega -\nabla p \cdot \psi \, dx$ and $\int_\Omega \nu^f \Delta \mathbf{v} \cdot \psi \, dx$ are integrated by parts. Considering a boundary conforming mesh, the resulting boundary integral usually vanishes, since the test functions have compact support on Ω and $\psi|_\Gamma = 0$. In the case of immersed boundaries, the degrees of freedoms on the Dirichlet boundary cells are considered as unknowns. Hence, the boundary integrals do not vanish:

$$\begin{aligned}
(-\nu_f \Delta \mathbf{v}, \psi)_\Omega &= \nu_f \int_\Omega \nabla \mathbf{v} : \nabla \psi \, dx - \nu_f \int_\Gamma \partial_n \mathbf{v} \cdot \psi \, dS(x) \\
&= \nu_f \left(\nabla \mathbf{v}, \nabla \psi\right)_\Omega - \nu_f \langle \partial_n \mathbf{v}, \psi \rangle_\Gamma, \\
(\nabla p, \psi)_\Omega &= -\int_\Omega p \left(\nabla \cdot \psi\right) dx + \int_\Gamma p \, n \cdot \psi \, dS(x) \\
&= -\left(p, \nabla \cdot \psi\right)_\Omega + \langle p \, n, \psi \rangle_\Gamma.
\end{aligned}$$

In the resulting equation, the weak form of the transient Navier-Stokes equations including the boundary integrals is written in the compact notation:

$$\begin{aligned}
\rho^f \left(\frac{\partial \mathbf{v}}{\partial t}, \psi\right)_\Omega + \nu^f \left(\nabla \mathbf{v}, \nabla \psi\right)_\Omega - \nu_f \langle \partial_n \mathbf{v}, \psi \rangle_\Gamma + \rho^f \left(\left(\mathbf{v} \cdot \nabla\right) \mathbf{v}, \psi\right)_\Omega \\
- \left(p, \nabla \cdot \psi\right)_\Omega + \langle p \, n, \psi \rangle_\Gamma + \left(\nabla \cdot \mathbf{v}, \xi\right)_\Omega - \left(\mathbf{f}, \psi\right)_\Omega &= 0. \qquad (4)
\end{aligned}$$

The formulation of Eq. (4) does not include the Dirichlet BCs, but those will be included by the Nitsche method.

The idea of the Nitsche method was introduced for the Poisson equation in [7]. It consist of defining an energy functional that measures the deviation of a discrete solution from the PDE solution in Ω and from the given Dirichlet BC on Γ. Analytically minimizing this functional results in the Nitsche Formulation of the given problem that also includes the Dirichlet BCs. For further details on the derivation of the Nitsche Method for the Poisson equation, we refer to the original paper [7], whereas for the Stokes and Navier-Stokes equation we refer to [4, 5, 6].

In the following, we simply restate the Nitsche Formulation of the Stokes and Navier-Stokes equations, without showing the derivation of the formula. For the sake of simplicity, we consider $\rho^f = 1$ and the stationary[1] case $\mathbf{v}_t = 0$. As a first step, we regroup the terms in Eq. (4). The volume integrals are denoted by

$$a(u, \phi) := \nu_f \left(\nabla \mathbf{v}, \nabla \psi\right)_\Omega + \left(\left(\mathbf{v} \cdot \nabla\right)\mathbf{v}, \psi\right)_\Omega - \left(p, \nabla \cdot \psi\right)_\Omega + \left(\nabla \cdot \mathbf{v}, \xi\right)_\Omega,$$

with $u(\mathbf{v}, p)$, and the boundary integrals are denoted by c

$$c(u, \psi) := -\nu \langle \partial_n \mathbf{v}, \psi \rangle_\Gamma + \langle pn, \psi \rangle_\Gamma.$$

[1] However, this approach also covers the transient case.

With this notation, the stationary[2] the NSE, where the Dirichlet BC has not been imposed yet, reads as

$$a(u,\phi) + c(u,\psi) = (f,\psi)_\Omega \;\; \forall\, \phi. \tag{5}$$

Using the Nitsche Method implies adding penalty terms and terms that maintain the skew-symmetry of the Stokes operator [5, 6] in Eq. (5). The skew symmetric counter term $\hat{c}$ of c has the following form:

$$\hat{c}(\mathbf{v},\phi) := -\nu_f \langle \partial_n \psi, \mathbf{v}\rangle_\Gamma - \langle \xi n, \mathbf{v}\rangle_\Gamma.$$

Further, the penalty terms $\nu_f \frac{\gamma_1}{h}\langle \mathbf{v},\psi\rangle_\Gamma + \frac{\gamma_2}{h}\langle \mathbf{v}\cdot n, \psi\cdot n\rangle_\Gamma$ are also added to Eq. (5).

Collecting all the listed terms results in the Nitsche Method of the stationary[3] NSE (5). We denote the discrete velocity space by V_h and the pressure space by Z_h. If the spaces V_h and Z_h are chosen such that the resulting discretization is not LBB stable [8], further stabilization terms are added to Eq. (5), as it is the case in [5, 4]:

$$\begin{aligned} &a(u_h,\phi_h) + c(u_h,\psi_h) + \hat{c}(\mathbf{v}_h,\phi_h) + \nu_f \frac{\gamma_1}{h}\langle \mathbf{v}_h,\psi_h\rangle_\Gamma + \frac{\gamma_2}{h}\langle \mathbf{v}_h\cdot n, \psi_h\cdot n\rangle_\Gamma \\ &= (\mathbf{f},\psi_h)_\Omega + \hat{c}(\mathbf{g},\phi_h) + \nu_f \frac{\gamma_1}{h}\langle \mathbf{g},\psi_h\rangle_\Gamma + \frac{\gamma_2}{h}\langle \mathbf{g}\cdot n, \psi_h\cdot n\rangle_\Gamma \;\; \forall\, \phi_h \in V_h \times Z_h. \end{aligned} \tag{6}$$

In Eq. (6), h denotes the local mesh width on the boundary Γ, where the Dirichlet BC $\mathbf{g}$ is imposed on. γ_1 and γ_2 are the penalty coefficients, which for our applications we set as $\gamma_1 = \gamma_2 = 10^2 - 10^3$. The formulation of Eq. (6) is consistent in the sense that the solution satisfies the variational problem. Furthermore, convergence for $h \longrightarrow 0$ is also assured.

Equation (6) shows the generality of Nitsche's approach, since it does not restrict the chosen discrete spaces V_h and Z_h. Hence, a Nitsche formulation can be employed in a mesh-free context as well [9]. In this paper, we focus on the mesh-based implementation of Eq. (6). We mentioned previously that Eq. (6) was already employed by [5] and [6] in similar forms. In both works, the Nitsche Method was employed on a boundary conforming mesh, where a Dirichlet BC could be imposed in a classical way. The novel approach of our work [10, 4] is that we employ this method in an immersed boundary context illustrated in Fig. 1 and for moving boundaries (Sect. 3).

Since the boundary is represented by a separate entity, the domains Ω and Ω_F can be represented by a memory and computationally efficient mesh. For this reason, we chose adaptive **Cartesian** meshes in 2D and 3D. This type of mesh has also good parallel capabilities, which was proven in [1], but will remain unexploited in this paper.

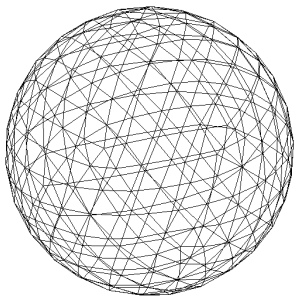

Figure 2: Illustration of a triangulated surface representing a sphere in 3D. (picture from [4])

In the following, we present step by step our approach and methodology to implement the Nitsche Method in 2D and 3D. The first element is the boundary geometry representation, for which we use polygons in 2D and triangulated surfaces in 3D. A triangulated surface representing a spherical surface is illustrated o in Fig. 2. Such simplicial boundary objects are relatively easy to handle, especially for the computations of

[2] For the Nitsche Method the stationary and transient problems lead to the same terms.
[3] and also transient

intersection points, which play an important role in volume and boundary integrals for the Nitsche Method in Eq. (6).

Such, the domains Ω and Ω_F are defined by a polygon in 2D and triangulated surfaces in 3D. According to Eq. (6), the only features required for the Nitsche Method are volume and boundary integrals. Considering the first example in Fig. 1, all non-intersected cells can be treated in a classical way, since they are completely in- or outside of the domain Ω. Therefore, only the intersected cells need to be treated in a special way (called cut-cell method). One example of a polygon intersecting a cell is illustrated in Fig. 3 in 2D, where a function f needs to be integrated on the Ω part of the cell or on the line segments of the polygon representing $\Gamma = \partial\Omega$.

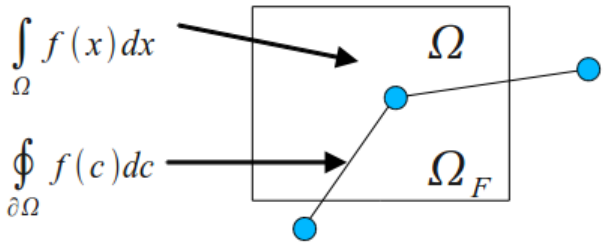

Figure 3: One possible scenario of a polygon and cell intersection. It illustrates that a given function f needs to be integrated on $\partial\Omega$ or in Ω. (picture from [4])

In the following, we consider the intersection of a cell E and the domain Ω similar to the illustration on Fig. 3. Given a function f, the task is to compute $\int_{E\cap\Omega} f\,dx$ and $\oint_{E\cap\Gamma} f\,dc$. Knowing that the underlying geometry representation is a polygon in 2D, the domain $E \cap \Omega$ can be decomposed into elementary objects, that can be integrate with up to machine precision. Fig. 4 shows three different scenarios in 2D of cell-

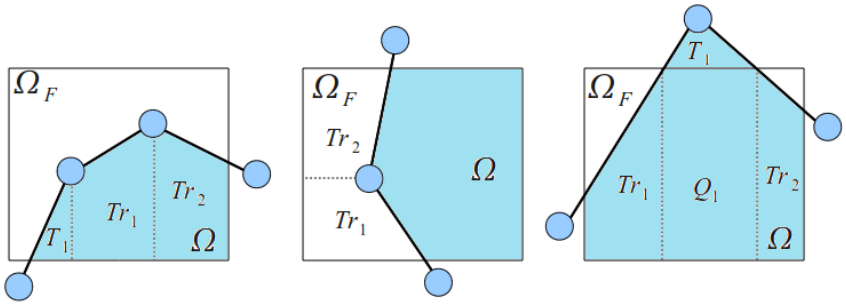

Figure 4: Illustration of the decomposition of a cell in elementary geometrical objects. This decomposition is given by the line segments of the polygon. In all three cases, the integrals in $E \cap \Omega$ or in $E \cap \Omega_F$ can be computed up to machine precision. (picture from [4])

geometry intersection, where in each case $E \cap \Omega$ or $E \cap \Omega_F$ can be be decomposed in a set of triangles (T_i), quads (Q_i), and trapezoids (Tr_i). The same rule also applies to the boundary integral on $E \cap \Gamma$, where a given function for the cell E needs to be integrated on the line segments contained in the cell E.

The same decomposition of the boundary and the domain inside a cell E can also be applied in 3D. In this case, the geometry is a triangulated surface that can intersect a brick cell in an arbitrary way. In 3D, to compute the integrals $\int_{E\cap\Omega} f\,dx$ and $\oint_{E\cap\Gamma} f\,dc$ up to machine precision would imply much more complicated algorithms than in 2D with polygons. Therefore, we only approximate the intersection surface by using the intersection points on the edges of the brick. This approximation of the intersection of a brick cell and a triangulated surface is shown in Fig. 5 for four possible scenarios. Fig. 5 shows only the intersection points[4], which are marked with blue circles. These intersection points give rise to the approximated intersection surface which is also formed by triangles. The surface integral of $\oint_{E\cap\Gamma} f\,dc$ is computed on this approximated

[4]Intersection between the edge of the brick cell and the triangulated surface

triangulated surface $(T_1, \ldots, T_n)$, $n = 3 : 6$. Cases, where an edge is intersected more than once or where the approximated intersection surface can not be built, can be eliminated by additional local mesh refinement. Therefore, in 3D, we do not consider these cases.

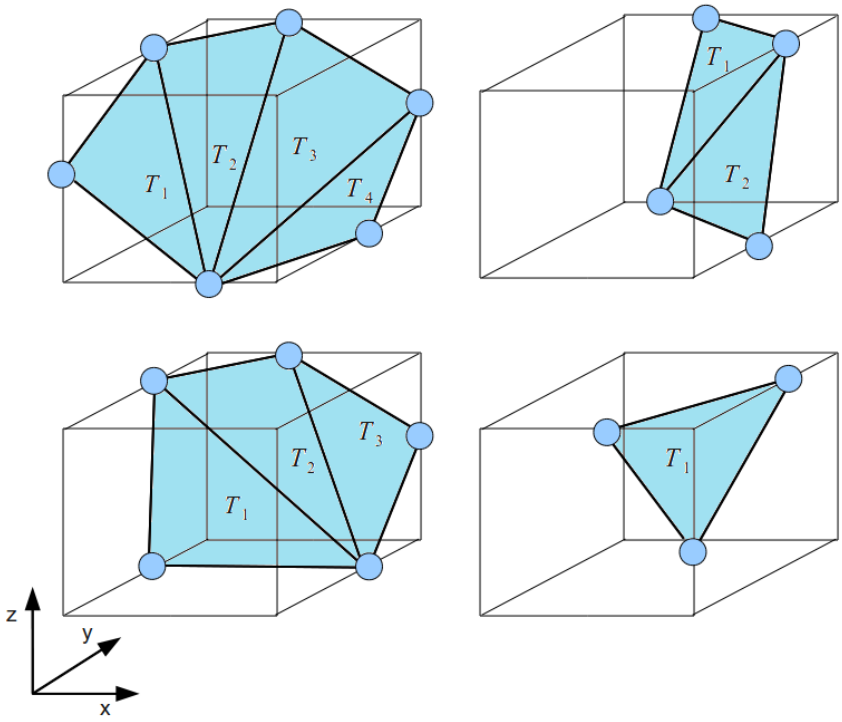

Figure 5: The intersection points on the edges form a convex, approximated, and triangulated surface within a brick cell E. (picture from [4])

For the volume integral of $\int_{E\cap\Omega} f(x)\,dx$, we employ the same idea of elementary decomposition. We use the presented approximated surface to build the decomposition of one brick cell into elementary objects. By projecting this surface onto one of the faces of the brick cell, we obtain the aimed volume decomposition. Fig. 6 illustrates the volume decomposition of the top left intersection case in Fig. 5 into elementary 3D cells (two prism and one tetrahedron cells).

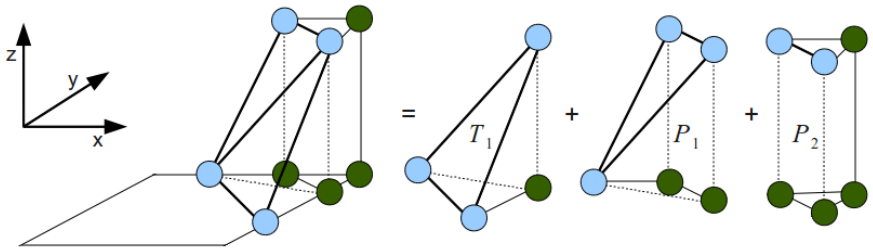

Figure 6: The illustration shows the decomposition of the top left case in Fig. 5 into elementary 3D cells. The $E \cap \Omega$ part of the brick is decomposed into one tetrahedron and two prism cells. (picture from [4])

In [10] and [4], we described the mathematical formulation of the concrete boundary and volume integrals, which were implemented within the Sundance PDE toolbox [11]. For cases, where $\int_{E\cap\Omega} 1\,dx \ll \int_{E\cap\Omega_F} 1\,dx$, a potential singularity would be introduced by the volume integration, since the element matrix of such cells would have nearly zero entries. In order to avoid such singularities, we also weight the Ω_F domain in the

cut-cell integration. In this way, the integral on cell E has the following form:

$$\alpha_1 \int_{E\cap\Omega} f\,dx + \alpha_2 \int_{E\cap\Omega_F} f\,dx, \tag{7}$$

where $\alpha_1 = 1.0$ but $\alpha_2 \in [10^{-8}; 10^{-5}]$. Using a non-zero α_2 parameter assures that the entries of the element matrix have at least a minimal size. Hence, the presented volume integrals on cells should not represent a significant source of singularity. For further details on the presented volume and boundary integrals, we refer to [10, 4].

2.1 Numerical Benchmark Results

In our previous work [10], we computed a stationary 2D benchmark scenario, where we were able to match the benchmark values specified in [12]. To extend the presented approach for transient simulations is rather simple. One only needs to add the time derivative (v_f, ψ) to Eq. (6). The results of a transient 2D benchmark scenario are presented in [4].

Here, we focus on a 3D stationary benchmark scenario (3D-1Z in [12]). In 3D, the generation of a boundary conforming mesh is more costly than in 2D. Furthermore, the parallel partitioning of an unstructured mesh becomes a significant computational overhead. In contrast, our approach does not pose these problems[5] even in 3D computations. In our approach, most of the computational overhead is restricted to the treatment of intersected cells, e.g. the computation of the volume and boundary integrals on these cells.

The 3D-1Z benchmark scenario is a 3D channel flow. In the middle upwind part of the channel, a cylinder obstacle is placed as illustrated in Fig. 7. In the upwind boundary of the channel, a velocity Dirichlet BC drives the flow in the channel. For further details on this benchmark scenario, we refer to [12].

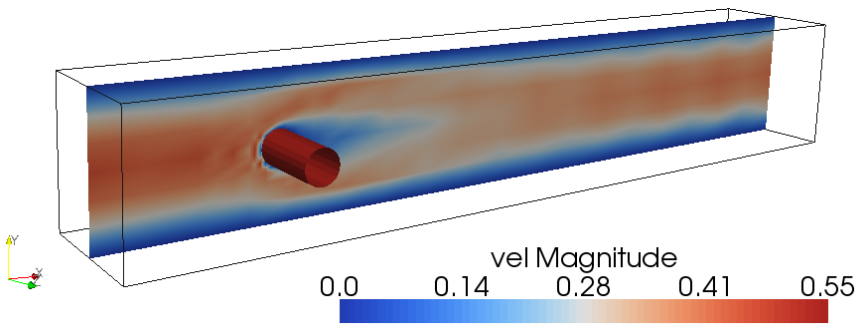

Figure 7: The 3D-1Z [12] benchmark scenario.

The benchmark values in this case are the benchmark lift and drag coefficients measured on the surface of the cylinder (see Fig. 7). These values can be computed by a surface integral over the boundary Γ [12],

$$F_\phi = \oint_\Gamma \phi \cdot \sigma_f(u) \cdot \mathbf{n}\, dS(x), \tag{8}$$

where $\sigma_f(u) = 2\nu\varepsilon_f(\mathbf{v}) - pI$ is the stress tensor, $\varepsilon_f(\mathbf{v}) = \frac{1}{2}(\nabla\mathbf{v} + \nabla\mathbf{v}^T)$ is the strain tensor, and ϕ is a unit vector pointing in force direction. We set $\phi = (1,0,0)^T$ for the total drag force computation, whereas $\phi = (0,1,0)^T$ results in the lift force integration. For more details on the computation of the benchmark values, we refer to [12].

In order to fulfill the LBB condition [8], we choose the Q_2 basis for the velocities and Q_1 for the pressure. The resulting Q_2Q_1 element in 3D requires 89 degrees of freedom for a brick element. Since this element type also results in a larger bandwidth of the system matrix, we also tried pressure stabilized Petrov-Galerkin (PSPG) Q_1Q_1 elements [13, 14]. Such a Q_1Q_1 PGPS element needs only 32 degrees of freedom for a brick

[5] due to the adaptive Cartesian mesh

element. Thus, computations can be done on meshes with higher resolution compared to higher order elements.

In the following, we show the results of the 3D-Z1 benchmark scenario in Tab. 1 with Q_2Q_1 and Q_1Q_1 elements. With the PSPG Q_1Q_1 elements, we were able to achieve considerably higher spatial resolution, than with the LBB-stable Q_2Q_1 elements. The limiting factor for the Q_2Q_1 is not just the total number of unknowns in the system, but also the resulting system matrix that becomes more dense and requires more storage.

Q_1Q_1, $\gamma_1 = \gamma_2 = 10^2$	Drag	Lift	#Cells
$40 \times 13 \times 13$, $l = 1$	6.100	0.021	18916
$20 \times 7 \times 7$, $l = 2$	5.810	0.077	25764
$22 \times 8 \times 8$, $l = 2$	6.053	0.071	32783
Benchmark intervals [12]	6.05 - 6.25	0.008-0.01	
Q_2Q_1, $\gamma_1 = \gamma_2 = 5 \cdot 10^2$	Drag	Lift	#Cells
$25 \times 10 \times 10$, $l = 1$	5.539	-0.107	2502
$27 \times 11 \times 11$, $l = 1$	6.000	0.078	7624
$26 \times 12 \times 12$, $l = 1$	6.021	0.432	9113
Benchmark intervals [12]	6.05 - 6.25	0.008-0.01	

Table 1: Results of the 3D-1Z benchmark computations. The columns *Drag* and *Lift* represent the two coefficients computed by the surface integrals. The first table contains the PSPG Q_1Q_1 elements' results, whereas the lower table shows the results of the Q_2Q_1 element. Besides the initial spatial resolution in the first column, l represents the number of additional refinement iterations around the obstacle. (results from [4])

The measured coefficients in Tab. 1 with increasing spacial resolution show that the drag coefficients with the PSPG Q_1Q_1 elements match the benchmark interval. The lift coefficients with the same element type miss the reference interval only slightly.

With the Q_2Q_1 elements, we were able to compute only 9113 elements. But even though, the measured drag coefficient differs from the lower bound of the reference interval only at the third digit. On the other hand, the lift coefficient has the same sign as the benchmark value, but the error is larger in comparison to the previous results. The poor result in the lift values might be due to the low mesh resolution at the obstacle. The results in Tab. 1 demonstrate that the developed volume and boundary integrals on intersected cells in 3D applied to the Nitsche Method perform well and were capable to compute benchmark values for the Navier-Stokes equations.

The implementation of this computation was done in the Sundance PDE toolbox [11]. Part of the code is presented in [4].

3 Moving Boundaries with the Nitsche Method

In this section, we extend the presented Nitsche Method also for moving boundaries in the fluid. Since we are not employing a space-time discretization, we focus on the obstacle move between two consecutive snapshots at t_1 and t_2. The movement of the boundary between these two snapshots is discrete, and from the perspective of time step t_2 it is seen as a jump. Therefore, the imposition of the Dirichlet velocity BCs[6] and the treatment of the boundary jump require a special approach. Furthermore, we are using a fixed mesh approach, where the mesh stays fixed for the whole simulation.

To illustrate this issue, we consider the illustration in Fig. 9, where we present the position of a circular obstacle in the flow at two discrete times, t_1 and t_2. The problem lies in the fact that at time t_2 the domain Ω and Ω_F has changed, and for the time discretization needed $\mathbf{v}_{t_1}$ might not be defined on the whole of Ω_{t_2}. In Fig. 7, we consider the domain $\Omega_{F1} \setminus \Omega_{F2}$ at time step t_2. This domain was previously (at time t_1) part of the fictitious domain, and, accordingly, the velocities $\mathbf{v}_{t_1}$ might be set in an unphysical way. Furthermore,

[6] These velocities correspond to the velocity of the boundary.

the domain $\Omega_{F2} \setminus \Omega_{F1}$ disappears from the computational domain at time t_1 and might cause the violation of the continuity equation (Eq. (2)) in the newly intersected cells.

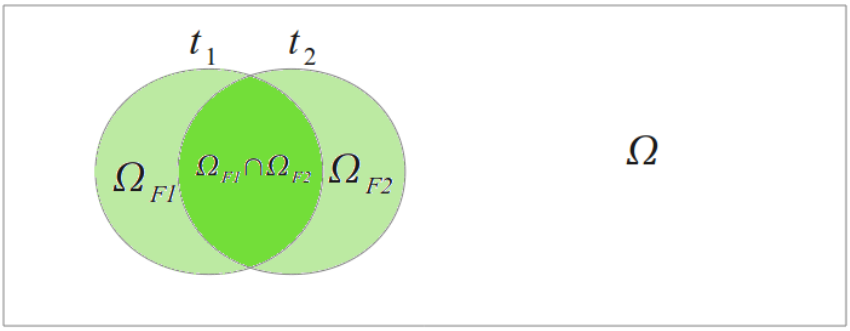

Figure 8: Illustration of the moving boundary issue at two different snapshots t_1 and t_2. Ω_{F1} and Ω_{F2} represent the fictitious domain at the two different time stamps. The computational domain Ω is also changing during this time step (picture from [4]).

Our solution to these problems is presented next. This approach is based on a fixed mesh that minimizes these effects impacting the solution of the transient NSE, as we impose the Dirichlet BC consistently with the Nitsche Method of Eq. (6). The first component of our approach is to use a fully implicit time discretization. Since the velocities $\mathbf{v}_{t_1}$ on $\Omega_{F1} \setminus \Omega_{F2}$ are not defined, we want to limit the impact of these values. Using a fully implicit time discretization implies that $\mathbf{v}_{t_1}$ will only appear in the discretization of the $\frac{d\mathbf{v}}{dt}$ term[7] in the momentum equation (Eq. (1)).

The second component of our approach is to limit the values of $\mathbf{v}_{t_1}$ on $\Omega_{F1} \setminus \Omega_{F2}$ to avoid large extreme values on the fictitious domain. In order to achieve this, we solve a weakly weighted Poisson equation on Ω_F at each time step ($\beta \Delta \mathbf{v} = 0$ on Ω_F). The weight β is usually set to 10^{-6}, similar to the value of α_2 in Eq. (7). First, this equation assures that the system matrix is solvable, since on each compute cell there must be an equation defined. Second, it assures that the values near the boundary will not take extreme values compared to the Dirichlet values.

Additionally, we also assume that the discrete move of the boundary is relatively small compared to the mesh width. This further assures that the velocity values at the new position of the boundary will have values close to the Dirichlet value, and that the 'undefined' velocities $\mathbf{v}_{t_1}$ will have a limited effect in the momentum equation.

In the following, we condense all the mentioned ideas into one formula that we use for the moving boundaries with the Nitsche Method. Based on the location of the point $x \in \mathbf{R}^d$ in the mesh, we have the following two equations:

$$
\begin{aligned}
&\left(\frac{\mathbf{v}_{h,t_2} - \mathbf{v}_{h,t_1}}{\Delta t}, \psi_h\right) + a(u_{h,t_2}, \phi_h) + c(u_{h,t_2}, \psi_h) + \hat{c}(\mathbf{v}_{h,t_2}, \phi_h) + \nu_f \frac{\gamma_1}{h} \langle \mathbf{v}_{h,t_2}, \psi_h \rangle_\Gamma + \frac{\gamma_2}{h} \langle \mathbf{v}_{h,t_2} \cdot n, \psi_h \cdot n \rangle_\Gamma \\
&= (\mathbf{f}, \psi_h)_\Omega + \hat{c}(\mathbf{g}_{t_2}, \phi_h) + \nu_f \frac{\gamma_1}{h} \langle \mathbf{g}_{t_2}, \psi_h \rangle_\Gamma + \frac{\gamma_2}{h} \langle \mathbf{g}_{t_2} \cdot n, \psi_h \cdot n \rangle_\Gamma \quad \forall\, \phi_h \in V_h \times Z_h \,,\ if\ x\ \in \Omega \\
&\beta\,(\nabla u_{h,t_2}, \nabla \phi_h) = 0 \quad \phi_h \in V_h \times Z_h \,,\ if\ x \in \Omega_F.
\end{aligned} \tag{9}
$$

In Eq. (9), we denote the unknowns at time t_2 by u_{h,t_2}. Since the Dirichlet BCs $\mathbf{g}$ might be also time variant, we denote the actual BC by $\mathbf{g}_{t_2}$.

In the following sections, we use the presented approach for the moving boundaries in fluid structure interaction (FSI) simulations. In FSI computations, it is crucial that the Dirichlet boundary is imposed consistently such that the resulting forces are also consistent and produce the correct structural deformations.

[7] in the weak form $\left(\frac{\mathbf{v}_{t_2} - \mathbf{v}_{t_1}}{\Delta t}\right) \psi$

4 Fluid Structure Interaction with the Nitsche Method

In this section, we briefly describe our FSI approach, which, according to our knowledge is a novel approach. We use the Nitsche Method on the fluid side as presented in Sect. 3, to impose the Dirichlet BC on the moving boundaries, whereas on the structure side we impose the Neumann BC in a classical way. It is crucial that on the fluid side the BCs on the moving boundaries are imposed in a consistent way, such that the correct coupling stress[8] vectors are computed. The numerical results in Section 5 verify our approach.

The structure equation is naturally set-up in the Lagrangian framework, where the boundary of the structure relative to the mesh stays fixed, and the displacement of the material point is defined by the displacement on the same point. In this work, we focus only on the fluid flow, modeled by the incompressible NS equations and set-up in the Eulerian framework. Hence, we do not present here the equations of the elastic structure. For a detailed description of the elastic body model and its solver we refer to [15, 4]. We consider the structure equation as a black box solver, that requires the stress vector t_s on the boundary as input. The resulting quantities after a given time step Δt are the displacement $\mathbf{u}_s$ and the velocity $\mathbf{v}_s$ vectors (see Fig. 9). $\mathbf{u}_s$ defines the new position of the boundary Γ_E. Γ_E denotes the boundary in the Eulerian framework that corresponds to the fluid's boundary. Γ_L denotes the structure's boundary in the Lagrangian framework that stays fixed by definition. Using this notation, the new position of the boundary is given by $\Gamma_E = \{x + \mathbf{u}_s \mid x \in \Gamma_L\}$.

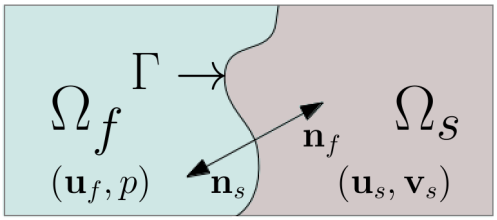

Figure 9: Coupling of the interface quantities (displacement, velocity, and stress) between the structure domain Ω_s and the fluid domain Ω_f. The illustration is made in the Eulerian framework (fluid side). $(\mathbf{u}_f, p)$ denotes the state of the fluid, whereas $(\mathbf{u}_s, \mathbf{v}_s)$ denotes the transient state of the structure (picture from [4]).

The structure's velocity $\mathbf{v}_s$ is the Dirichlet BCs for the fluid. The stress vectors on the fluid side are computed by $\sigma_f \cdot \mathbf{n}_f = t_s$, where σ_f was already used and defined in Eq. (8). By Summarizing all these aspects of the coupling, we can derive the following equations on the coupling interface Γ_E:

$$\begin{aligned} &\mathbf{v}_f = \mathbf{v}_s, \\ &\sigma_f \cdot \mathbf{n}_f = t_s, \\ &\Gamma_E := \{x + \mathbf{u}_s \mid x \in \Gamma_L\}. \end{aligned} \tag{10}$$

We already mentioned that the structure and fluid problems are set-up in two different frameworks and are also solved by separate solvers.[9] This approach of solving an FSI problem is called **partitioned** solving. In contrast to the **monolithic** solving, the partitioned approach solves the two problems completely separately and then couples the interface quantities. Oroginally, the interface quantities where coupled only once at the discrete time steps that is called as **explicit coupling**. However, this type of partitioned coupling proves to be unstable for scenarios with given properties [16, 17], such as the density ratio of the fluid and structure and the stiffness of the structure. In these category of scenarios fall the transient benchmark scenarios in 2D that we compute here. Therefore, in each time step an inner iteration loop is required, that ensures the convergence of the coupled quantities at each time step. This approach is called **implicit** partitioned coupling. For this inner iteration we employ the Aitken method that was first used for FSI in [18].

In the stationary case, the partitioned coupling approach is significantly simplified. Then, by default, $\mathbf{v}_f = \mathbf{v}_s = 0$ holds on the boundary, and therefore only the last two equations from Eq. (10) hold. For the

[8] or force

[9] for a given time step Δt

partitioned coupling, only one iteration loop needs to be done, until the remaining two coupling quantities converge. For further details on FSI coupling approaches, we refer to [16, 19, 2].

5 FSI Results

In this section, we compute various FSI scenarios in order to verify our Nitsche approach of imposing the Dirichlet BCs on moving and fixed boundaries in the fluid. The 2D FSI benchmark scenarios from this section are described in [15]. Unfortunately, there are no similar validated benchmark scenarios in 3D. Therefore, we set up one stationary 3D FSI scenario, which is described in more detail in [4]. For more details on the scenario setups and on these computations, we refer to [4, 15].

5.1 Stationary FSI

We start with the 2D stationary benchmark that is called FSI1 [15]. The elastic structure is placed into the fluid channel, where a parabolic inflow BC drives the flow. The obstacle is formed by a rigid cylinder and by an elastic bar that is attached to the downwind part of the cylinder as shown in Fig. 10. Since this obstacle is not placed symmetrically into this channel, a small positive lift coefficient is measured. The structure and the fluid equations are solved separately, and as we described in Section 2, we impose the zero velocity BC with the Nitsche Method. With a constant underrelaxation factor (see [4]), we iterate until the displacement update of the boundary is below a given value $\varepsilon = 10^{-7}$. The benchmark values in all cases are the displacements at the middle point of the elastic bar's right tip. Beside the displacements, the benchmark drag and lift forces are also measured.

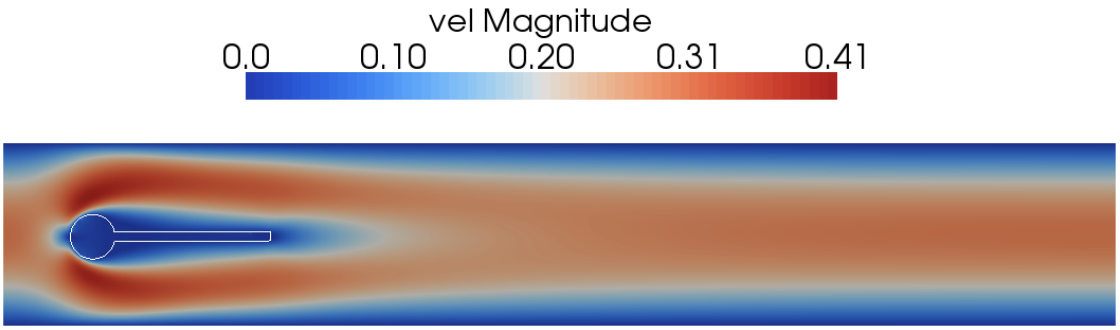

Figure 10: Illustration of the FSI1 and FSI3 [15] scenario configurations (picture from [4]).

We compute the FSI1 scenario with increasing mesh resolution. The results are shown in Tab. 2. The columns of Ax and Ay show the X and the Y displacements at the measuring point. Furthermore, Fl is the number of Q_2Q_1 fluid elements, whereas Sr is the number of elements in the structure's mesh. Beside these two quantities, the number of polygon points also plays a role in the spatial resolution of the problem that is denoted by $Poly$. One can observe that with increasing spatial resolution the measured displacement

#Fl × #Sr × #Poly	Ax	Ay	Drag	Lift
2601 × 3380 × 136	1.86e-5	0.00123	14.0779	0.820194
36666 × 13370 × 1093	2.21e-5	0.000703	14.1982	0.810081
54104 × 13370 × 1093	2.18e-5	0.000846	14.2236	0.793047
Benchmark values [15]	2.27e-5	0.000821	14.295	0.7638

Table 2: Stationary FSI1 results. The spatial resolution of the fluid, structure, and polygon (#Fl × #Sr × #Poly) is specified by the number of elements. Ax and Ay represent the displacements at the reference point, whereas Drag and Lift represent the drag and lift forces computed on the boundary of the structure.

and forces are converging to the benchmark values in [15]. It is important to note that even with relatively

low resolution (first row of Tab. 2) the measured displacements and forces are quantitatively accurate and the relative error for all four is less that 50%. This also shows the true potential of the Nitsche Method for immersed boundaries.

In the next step, we consider a stationary 3D FSI scenario that is not a benchmarks scenario and is presented in Fig. 11. The flow channel in Fig. 11 has a size of $2.5 \times 0.41 \times 0.41$. The fluid has a density of $10^3 \frac{kg}{m^3}$, similar to the previous FSI1 2D scenario and its parabolic inflow velocity[10] is $0.45 \frac{m}{s}$. All other parameters from this scenario correspond to the previous FSI1 scenario [15]. In this channel, the elastic obstacle is a vertical bar, characterized by the Young modulus of $E = 0.4 \cdot 10^6 \frac{kg}{ms^2}$ and the Poisson ratio of $\nu_s = 0.4$. This structure object has a size of $0.05 \times 0.25 \times 0.1$ and is placed at the position of $(0.45, 0.0, 0.125)$ into the flow channel. As illustrated in Fig. 11, we use an initial 3D mesh resolution for the fluid and further refine the cells around the obstacle. For spatial discretization, we use the PSPG Q_1Q_1 elements, which we already employed successfully for the 3D-1Z scenario [12]. This scenario is also described in more details in [4].

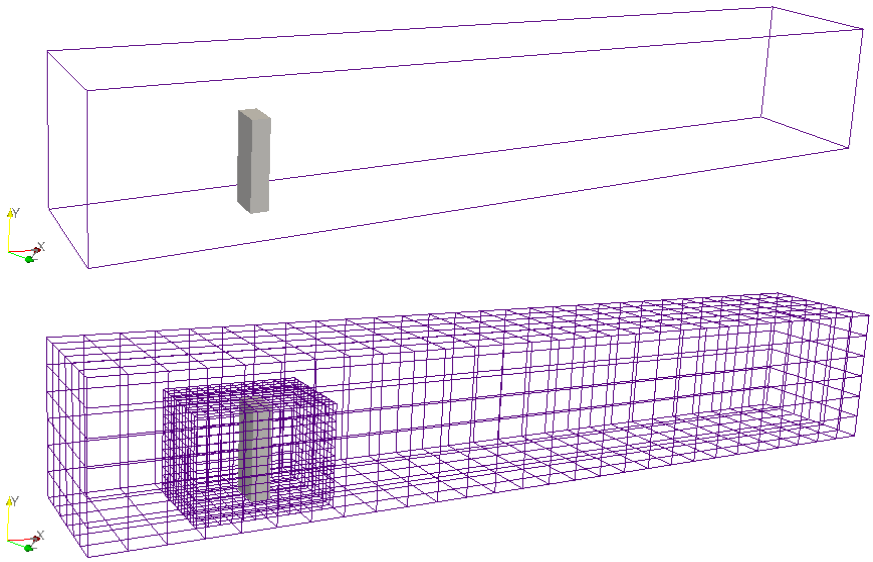

Figure 11: Illustration of the stationary 3D FSI scenario setup (picture from [4]).

We use the same stationary coupling algorithm. The coupling surface has 620 triangles. On the elastic bar, at the Lagrangian position of $(0.475, 0.25, 0.175)$ that represents the midpoint of the top face of the bar, we measure the displacement vector of $(1.1667e-2, -4.2293e-4, 6.619e-3)$. As we expected, the most dominant displacement is in the x-direction. However, due to the non-central position of the elastic bar, there is also a significant displacement in z-direction. This scenario further demonstrates the usability of our approach in 3D.

5.2 Transient FSI

As a final example, we consider the transient 2D FSI3 benchmark scenario from [15]. This is also the only 2D transient benchmark scenario that is considered in the review paper of [17]. The scenario setup is the same as for FSI1, except that the inflow velocity is increased, such that an oscillation of the elastic bar is produced. For a detailed description of the scenario and the model parameters, we refer to [15]. In this scenario, we employ the approach for the moving boundaries presented in Section 3 and prove its usability for FSI simulations.

This scenario with partitioned coupling requires implicit coupling, which makes the simulation of this scenario particularlyy expensive.[11] Therefore, we compute this scenario with lower resolution than the

[10] in average

[11] The simulation time is about 6 seconds, whereas the discrete time step was 0.001 second. Additionally, in average 13

stationary FSI1 scenario. The resulting time-dependent displacements and forces are shown in Tab. 3. Since the position of the reference point (Ax, Ay) is time-dependent, we measure the average value and the amplitude of the oscillation. The time-dependent behavior of the system is represented by the frequency of the oscillation. The computed displacements especially in the y-direction match the benchmark values from [15].

#Fl×#Sr×#Poly	Ax	Ay	Drag	Lift
2507 × 864 × 136	-0.02866 ±0.02857[11.1]	-0.00111 ±0.03301[5.5]	524.5 ±23.5[11.1]	56.50 ±214.50[5.5]
5133 × 864 × 136	-0.00288 ±0.00282[11.4]	0.00166 ±0.03452[5.7]	532 ±25[11.4]	-1.5 ±229.50[5.7]
Benchmark [15]	-0.00269 ±0.00253[10.9]	0.00148 ±0.03438[5.3]	457.3 ±22.66[10.9]	2.2 ±149.78[5.3]

Table 3: Results of the transient FSI3 scenario. The offset, the amplitude, and the frequency are measured for each of the four benchmark quantities.

This amplitude in y-direction is the main characteristic measure of this benchmark. The displacements in x-direction are also near the benchmark value. The lift and drag forces show a larger deviation compared to the values in [15], but even in these cases the relative error is between $10-50\%$. We show the time dependent drag and lift forces on Fig. 12. Especially in the drag force plot, we observe a high frequency noise, that is due to our fixed mesh approach. However, these distortions are not significant in their amplitude and duration and they do not influence the results in a significant way. In the frequency domain, the relative error of the measured frequency is less than10%. By summarizing all these results, we can conclude that using the presented approach for moving boundaries we were able to reproduce the benchmark values of a transient FSI scenario. Therefore, we verified our approach for FSI applications. We illustrate again the moving boundaries on fixed meshes on Fig. 13, where we show two different snapshot of the fixed mesh and with the structure mapped into the fluid domain. The arrows on the boundary represent the stress vectors acting on the surface of the structure. For further details on the computed results, we refer to [4].

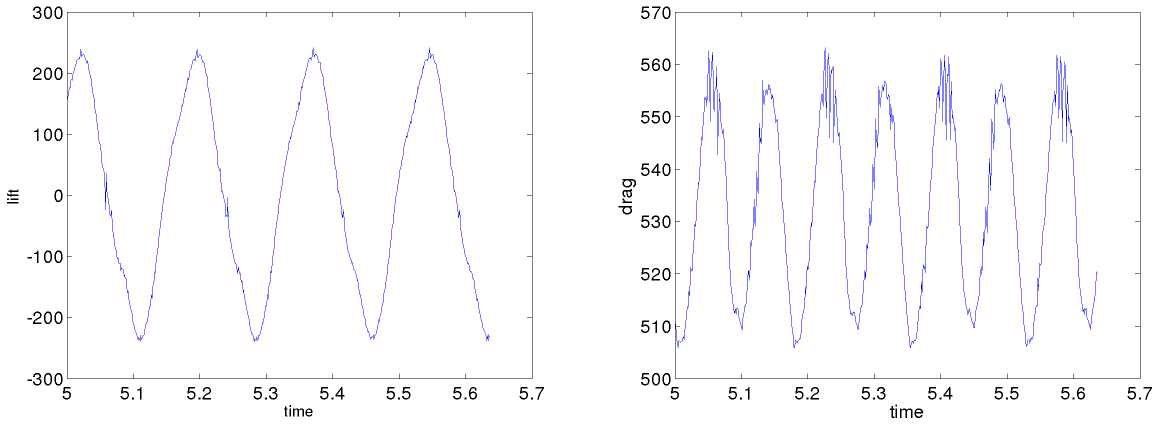

Figure 12: The time-dependent lift and drag forces of the FSI3 simulation (picture from [4]).

6 Summary and Future Work

In this paper, we presented the Nitsche Method for the Navier Stokes equations to impose Dirichlet BCs on immersed boundaries. In such a context, the boundary is represented explicitly by an separate entity.

implicit iterations were required for one implicit time step.

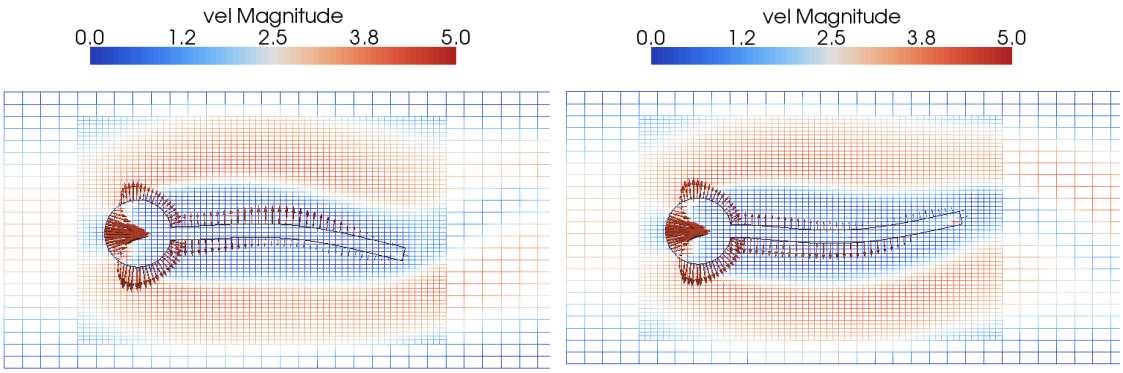

Figure 13: Two different snapshots of the FSI3 scenario. The structure is mapped into the fluid mesh. The arrows on the structure's boundary show the acting stress vectors.

We used polygons in 2D and triangular surfaces in 3D. Besides the Nitsche formulation of the equations, we implemented special cut-cell and boundary integrals for the intersected cells. Furthermore, we extended our approach for moving boundary scenarios, which allow us to compute transient FSI scenarios. This approach has major advantages in comparison to the unstructured mesh approach. First, we can use a storage-efficient and easy to decompose Cartesian meshes. Second, the mesh stays fixed during the simulation and no mesh operations are required. We computed various scenarios in 2D and 3D that verify our approach for fixed and moving boundaries. For more details on our approach and on the computed scenarios, we refer to [4].

On the other hand, we did not exploited some of the advantages of our approach such as the parallel capabilities of the adaptive Cartesian mesh. All the computations were done within the Sundance PDE toolbox [11], that uses external linear algebra solvers. As a future work, it also remains to develope or find a suitable preconditioner and iterative solver, which solve the resulting system matrix[12] with a high parallel efficiency. Such parallel solvers with Cartesian meshes could reduce the parallel computing time especially in 3D.

References

[1] C. Burstedde, L.C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[2] H.-J. Bungartz, J. Benk, B. Gatzhammer, M. Mehl, and T. Neckel. *Fluid-Structure Interaction – Modelling, Simulation, Optimisation, Part II*, volume 73 of *LNCSE*, chapter Partitioned Simulation of Fluid-Structure Interaction on Cartesian Grids, pages 255–284. Springer, Berlin, Heidelberg, October 2010.

[3] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37(1):239–261, 2005.

[4] J. Benk. *Immersed Boundary Methods within a PDE Toolbox on Distributed Memory Systems.* Dissertation (submitted), Technische Universität München, 2012.

[5] R. Becker. Mesh adaptation for Dirichlet flow control via Nitsche's method. *Communications in Numerical Methods in Engineering*, 18(9):669–680, 2002.

[6] P. Hansbo and M. Juntunen. Weakly imposed dirichlet boundary conditions for the brinkman model of porous media flow. *Applied Numerical Mathematics*, 59(9):1274–1289, 2009.

[7] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36:9–15, 1971.

[8] P. M. Gresho, R. L. Sani, and M. S. Engelman. *Incompressible Flow and the Finite Element Method.* John Wiley & Sons, 1998.

[12] matrix that results from the Nitsche formulation

[9] A. Huerta, T. Belytschko, T. Fernandez-Mendez, and T. Rabczuk. *Meshfree Methods.* vol. 1 of Encyclopedia of Computational Mechanics ch. 10, pp. 279-309. Wiley, 2004.

[10] J. Benk, M. Mehl, and M. Ulbrich. Sundance pde solvers on cartesian fixed grids in complex and variable geometries. In *Proceedinggs of the ECCOMAS Thematic Conference CFD & Optimization, Antlya, Turkey, May 23-25, 2011*, 2011.

[11] K. Long. Sundance Website. [http://www.math.ttu.edu/ klong/Sundance/html/index.html], 2007.

[12] M. Schäfer and S. Turek. Benchmark computations of laminar flow around a cylinder. In *Flow simulation with high-performance computers. Bd. 2.*, volume 52 of *Notes on numerical fluid mechanics*, pages 547–566. Vieweg, Braunschweig, January 1996.

[13] T. E. Tezduyar and Y. Osawa. Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering*, 190(31):411–430, 2000.

[14] T.J.R. Hughes, G. Scovazzi, and L.P. Franca. *Multiscale and Stabilized Methods.* in Encyclopedia of Computational Mechanics , eds. E. Stein, R. De Borst, T. J. R. Hughes. Wiley, 2004.

[15] J. Hron and S. Turek. Proposal for numerical benchmarking of fluid-structure interaction between elastic object and laminar incompressible flow. In H.-J. Bungartz and M. Schäfer, editors, *Fluid-Structure Interaction*, number 53 in Lecture Notes in Computational Science and Engineering, pages 371–385. Springer-Verlag, 2006.

[16] J. Degroote. *Development of algorithms for the partitioned simulation of strongly coupled fluid-structure interaction problems.* Dissertation, Ghent University. Faculty of Engineering, 2010.

[17] S. Turek, J. Hron, M. Razzaq, H. Wobker, and M. Schäfer. *Numerical Benchmarking of Fluid-Structure Interaction: A Comparison of Different Discretization and Solution Approaches*, volume 73 of *Lecture Notes in Computational Science and Engineering*, chapter 15, pages 413–424. Springer Berlin Heidelberg, 2010.

[18] U. Kuettler and W.A. Wall. Fixed-point fluid-structure interaction solvers with dynamic relaxation. In *Computational Mechanics.* Springer, 2008.

[19] A. Gerstenberger. *An XFEM based fixed-grid approach to fluid-structure interaction.* Dissertation, Technische Universität München, 2010.