# Higher order space–time finite element methods on evolving and fixed domains

**– applied to the Navier–Stokes equations and wave problems –**

Von der Fakultät für Maschinenbau und Bauingenieurwesen
der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg
zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

DISSERTATION

vorgelegt von

Mathias Anselmann

aus Neustadt an der Weinstraße

Hamburg 2022

**Gutachter:**

**Prof. Dr. Markus Bause**
Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg, Deutschland

**Prof. Dr. Thomas Richter**
Otto-von-Guericke-Universität Magdeburg, Deutschland

Tag der mündlichen Prüfung: 11. Mai 2022

# Higher order space–time finite element methods on evolving and fixed domains

**– applied to the Navier–Stokes equations and wave problems –**

Doctoral Thesis

approved by the Department of Mechanical Engineering and Civil Engineering
of the Helmut-Schmidt-University /
University of the German Federal Armed Forces Hamburg
for obtaining the academic degree of Doktor-Ingenieur (Dr.-Ing.)

presented by

Mathias Anselmann

from Neustadt an der Weinstraße

Hamburg 2022

**Referees:**

**Prof. Dr. Markus Bause**
Helmut-Schmidt-University / University of German Federal Armed Forces Hamburg, Germany

**Prof. Dr. Thomas Richter**
Otto von Guericke University Magdeburg, Germany

Day of the completion of the oral examination: May 11, 2022

# Zusammenfassung

In dieser Arbeit werden verschiedene Aspekte von Raum–Zeit Finite Elemente Diskretisierungen für die Navier–Stokes Gleichungen und Wellenprobleme untersucht. Zunächst wird in Chapter 1 ein Überblick über den aktuellen Stand der Wissenschaft gegeben und in Chapter 2 die Notation vorgestellt.

In Chapter 3 werden Strömungsprobleme auf zeitabhängigen Gebieten untersucht. Diese spielen in der Ingenieurpraxis eine große Rolle, beispielsweise bei der Simulation von Fluid-Struktur-Interaktionsproblemen, wie sie etwa bei der Interaktion eines Fluids mit einem Rotor vorkommt. Dazu werden in dieser Arbeit Starrkörper betrachtet, die sich in einem viskosen Fluid bewegen, welches durch die zeitabhängigen Navier–Stokes Gleichungen beschrieben wird. Das bedeutet wiederum, dass sich das Strömungsgebiet, in Abhängigkeit von der Position des Starrkörpers, verändert. Ein klassischer Weg zur Simulation solcher Probleme ist die Arbitrary-Lagrangian-Eulerian-Methode (ALE-Methode), die auf der Transformation des Strömungsgebiets auf ein Referenzgebiet basiert. Diese Transformation hat allerdings den Nachteil, dass sie rechenintensiv ist, Nichtlinearitäten hervorrufen kann und zudem, bei größeren Verformungen des Strömungsgebietes, eine Aktualisierung des zugrunde liegenden Rechengitters unausweichlich ist. Solch ein Remeshing ist ebenfalls sehr rechenaufwändig und bedeutet in der Regel, dass alle, der Simulation zugrunde liegenden Matrizen und Vektoren, neu aufgebaut werden müssen. Um dies zu vermeiden, wird in der vorliegenden Arbeit ein Verfahren vorgestellt, bei dem alle Berechnungen auf einem festen Hintergrundgitter durchgeführt werden, das willkürlich von dem Starrkörper geschnitten werden kann. Diese Methode bezeichnet man als Cut Finite Elmenente Methode (CutFEM). Voraussetzung für eine Implementierung ist es, Funktionen über willkürlich geschnittene Zellen zu integrieren, um die zugrunde liegenden Systemmatrizen und Vektoren assemblieren zu können. Dazu wurde ein flexibles Verfahren entwickelt, was auf der iterierten Anwendung von 1d Integralen beruht.
Im Falle von Zellen, die durch das Schneiden mit dem Starrkörper sehr klein werden, entstehen unphysikalische, steile Gradienten in der diskreten Lösung auf diesen Zellen. Diese resultieren in einer schlechten Konditionierung der Systemmatrix und nu-

merischen Instabilitäten. Um das Problem zu umgehen, wurde eine sogenannte „ghost penalty“ Stabilisierung implementiert, welche die Gradienten beschränkt und das Verfahren robust macht. Darüber hinaus setzt der Stabilisierungsoperator die Strömungslösung implizit in das gesamte Starrkörpergebiet fort. Dies ermöglicht es, alle Rechnungen auf einem festen, zeitunabhängigen Hintergrundgitter durchzuführen. Alle bisher genannten Aspekte wurden in numerischen Konvergenztests und Beispielrechnungen überprüft.

In Chapter 4 wird dann zunächst für zeitunabhängige Gebiete ein geometrisches Mehrgitterverfahren für Finite Elemente Raum–Zeit Diskretisierungen höherer Ordnung der Navier–Stokes Gleichungen implementiert. Das Verfahren basiert auf einem Vanka Glätter, der in jedem Zeitschritt hochfrequente Fehleranteile über alle Ort–Zeit Freiheitsgrade dämpft. Alle wesentlichen Programmteile sind mittels des Message Passing Interface (MPI) parallelisiert. Das Mehrgitterverfahren wird in der Arbeit genutzt, um den 3d DFG Benchmark „Fluss um einen Zylinder“, mit mehr als 96 Millionen Orts–Zeit Freiheitsgraden pro Zeitschritt, zu simulieren. Das Adaptieren des Verfahrens auf Probleme mit zeitabhängigen Gebieten, in Verbindung mit CutFEM, resultiert, abhängig von der Größe des Starrkörpers, in erhöhten Iterationszahlen. Die Ursachen dafür werden in diesem Kapitel der Arbeit ebenfalls analysiert und numerisch untersucht.

In Chapter 5 werden stetige Finite Elemente Methoden für die Zeitdiskretisierung mit klassischen Kollokationsmethoden kombiniert, um effiziente Verfahren höherer Ordnung in der Zeit zu erhalten, die zudem in einer diskreten Lösung höherer Regularität resultieren. Solche Galerkin–Collocation Methoden werden in dieser Arbeit zuerst für die Wellengleichung und anschließend für die Navier–Stokes Gleichungen entwickelt und numerisch analysiert. Im Vergleich zu traditionellen, kontinuierlichen Finite Elemente Zeitdiskretisierungen konnte mit diesen Methoden die Zeitschrittweite, bei gleicher Qualität der Lösung, deutlich erhöht werden und sorgte für ein Einsparen von Rechenleistung. Insbesondere bei gekoppelten Problemen könnten Verfahren dieser Art vorteilhaft sein, wenn Koeffizienten der Ableitung des einen Problems in das andere Problem eingehen.

# Abstract

In this work, various aspects of space–time finite element discretizations for the Navier–Stokes equations and wave problems are investigated. First, an overview of the current state of the science is given in Chapter 1 and the notation is introduced in Chapter 2.

Then, in Chapter 3, flow problems on time-dependent domains are studied. These play a major role in engineering practice, for example in the simulation of fluid-structure interaction problems, such as the interaction of a fluid with a rotor. In this work, rigid bodies moving in a viscous fluid described by the time-dependent Navier–Stokes equations are considered as an example problem for this purpose. This means that the flow area changes depending on the position of the rigid body. A classical way to simulate such problems is the Arbitrary-Lagrangian-Eulerian method (ALE method), which is based on the transformation of the flow domain to a reference mesh. This transformation has the disadvantage that it is computationally intensive, can induce nonlinearities, and also, for larger deformations of the flow domain, an update of the underlying computational grid is inevitable. Further, such a remeshing is again computationally expensive and usually means that all matrices and vectors of the underlying simulation have to be rebuilt. In this work, a method is presented in which all computations are performed on a fixed background grid that can be arbitrarily cut by the rigid body. This method is called the Cut Finite Elmenent Method (CutFEM). A prerequisite for an implementation is to be able to integrate functions over arbitrarily cut cells in order to be able to assemble the underlying system matrices and vectors. To accomplish this, a flexible method was developed, based on the iterated application of 1d integrals.
In the case of cells that become very small, if they are cut by the rigid body, unphysical steep gradients of the solution arise in these cells. These result in poor conditioning of the system matrix and numerical instabilities. To circumvent the problem, a so-called "ghost penalty" stabilization was implemented, which constrains the gradients and makes the method robust. Furthermore, the stabilization operator implicitly propagates the flow solution into the entire rigid-body domain. This allows all calculations

to be performed on a fixed, time-independent background grid. All aspects mentioned so far have been verified in numerical convergence tests and simulations.

Then, in Chapter 4, a geometric multigrid method for higher order finite element space–time discretizations of the Navier–Stokes equations is then implemented for time-independent domains. The method is based on a Vanka smoother, which dampens high-frequency error components over all space-time degrees of freedom in each time step. All major parts of our code are implemented in parallel using the Message Passing Interface (MPI). The multigrid method, used in this work, is verified by simulating the 3d DFG benchmark "flow around a cylinder", with more than 96 million space–time degrees of freedom per time step. Adapting the method to problems with time-dependent domains in conjunction with CutFEM results in increased iteration counts, depending on the size of the rigid body. The causes of this are analyzed and investigated numerically in this work.

In Chapter 5 classical continuous finite element methods for time discretization are combined with collocation methods to obtain efficient higher order methods in time, which also result in a discrete solution of higher regularity. Such Galerkin–Collocation methods are developed and applied first for the wave equation and then for the Navier–Stokes equations. Compared to traditional continuous finite element time discretizations, we could significantly increase the time step size with these methods, while maintaining the same quality of the solution. Especially in coupled problems, methods of this type could be advantageous if coefficients of the derivative of one problem are included in the problem description of the other problem.

# 1 Introduction and outline

In this work higher order space-time discretizations for the Navier–Stokes equations and wave problems are presented and computationally studied. These equations have attracted researcher's interest for many decades now. The Navier–Stokes equations describe the motion of viscous fluids. Despite the intense research and their importance in science and engineering, there are still many open questions regarding these coupled set of equations.

In contrast to fluids, the wave equation describes the expansion of waves in a domain. In many scenarios these two equations are coupled. The aim of such a multi-physics setting is to simulate the interaction of fluids with, for example, elastic structures. Such settings have many applications in engineering, for example when it comes to the simulation and optimization of wings in turbines or planes. In the following sections we will give a brief introduction to the current state of higher order space-time Finite Element Methods (FEM), that we use later in this work to solve flow and wave problems. We will further give an overview of the Cut Finite Element Method (CutFEM), that is used in this work to solve flow problems on grids, which are not fitted to the underlying physical domains. Finally we give an outline of the thesis.

## 1.1 Higher order space-time FEM

Many scenarios of technical relevance aim to simulate multi physics scenarios, like the movement of a rigid or elastic body in a fluid. Then one wants to obtain certain data, like forces, that act on the structure, as exactly as possible. One way to get efficiently solutions of high quality and high precision is to use higher order space-time methods. Such high-order numerical approximations of partial differential equations have been strongly investigated in the last decades. They are known to be efficient if they approximate functions with large elements of high polynomial degree in regions of high regularity. Prominent examples are *hp*- and spectral element methods in application

areas such as computational fluid dynamics or computational mechanics. Their theoretical convergence analysis and adaptive *hp*- and spectral element methods are still subject of current research.

Higher order methods offer the potential to achieve accurate results on computationally feasible grids with a minimum of numerical costs on economical meshes. Driven by the tremendous increase in computing power of modern high performance computing systems and recent progress in the technology of algebraic solvers, including efficient techniques of preconditioning, space-time finite element approaches have recently attracted high attention and have been brought to application maturity; cf., e.g., [1]–[4]. Space-time finite element methods offer appreciable advantages over discretizations of mixed type based on finite difference techniques for the discretization of the time variable (e.g., by Runge-Kutta methods) and, for instance, finite element methods for the discretization of the space variables. In particular, advantages are the natural embedding of higher order members in the various families of schemes, the applicability of functional analysis techniques in their analyses due to the uniform space-time framework and the applicability of well-known adaptive mesh refinement techniques, including goal-oriented error control [5].

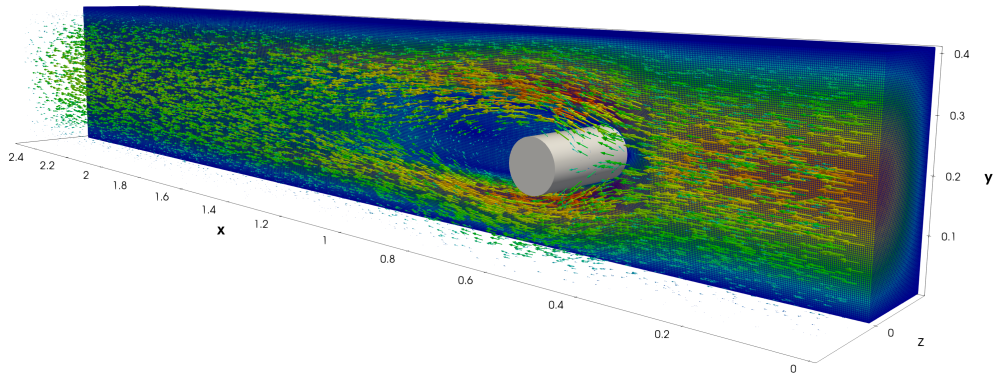



Figure 1.1: Flow around a cylinder, simulated by higher order space-time FEM.

In the meantime a broad variety of implementations of space-time finite element methods do exist. The families of schemes differ by the choices of the trial and test spaces. This leads to continuous or discontinuous approximations of the time variable (cf. e.g., [6], [7]). Whereas high-order approaches have been considered for the approximation of the spatial variables, first- or second-order implicit schemes are often still used for the discretization of the time variable. One reason for this is the increased complexity

of the underlying linear systems. Recently the traditional continuous in time FEM schemes were combined with classical collocation schemes [8], [9]. By combining these two methods one gets higher order solutions with less complex linear systems than pure FEM in time schemes and the additional benefit, that the discrete solution is of higher regularity.

## 1.2 FEM on moving and evolving domains

When it comes to the simulation of coupled problems, like fluid-structure interaction (FSI), one has to consider multiple domains, e.g. fluid and rigid ones. In dynamic scenarios, these domains can also be evolving or moving. A classical and well-known way of coping with such a situation is by using the Arbitrary–Lagrangian–Eulerian (ALE) method (cf. [10]–[14]). In this method the domains are mapped to an arbitrary chosen reference domain, where the computations are performed. This has the advantage, that the reference mesh doesn't have to be rearranged, when the physical domains move. In classical ALE approaches one drawback is, that large motions of the physical domains lead to a poor quality of the physical mesh (cf., e.g. [15]–[17]), since it has to track and resolve moving boundaries or interfaces. This mesh deformation impacts the transformation of the model equations to the reference domain and, thereby, the stability of the overall ALE approach. This makes at a certain point remeshing inevitable, which is computationally expensive. Further the solutions have to be projected from one mesh to another, which can cause an accumulation of projection errors (cf. [18]).

In the last years, methods that perform the computations on a fixed background grid, where non fitted subdomains are embedded, established. One way of resolving a discontinuity (like a domain boundary or a crack) on a fixed grid is to enrich the finite element space on the specific cells, that have to resolve the discontinuity. This method is called *eXtended Finite Element Method* (XFEM). It was introduced by [19] and applied to fracture mechanical problems. Later it was also applied to problems of thophase flow [20]. In this work we will make use of a technique that is called *Cut Finite Element Method* (CutFEM), see [21]–[23]. Here, classical finite elements are used, but they are cut arbitrarily by discontinuities like physical domain interfaces. This can be seen as a XFEM method, where the finite element space is multiplied with a Heaviside enrichment function [24]. CutFEM has been analyzed and studied numerically for a wide range of different Finite Element discretizations, e.g. [25]–[27]. Even the simulation on complex domains [28]–[31] or even the simulation of coupled problems,

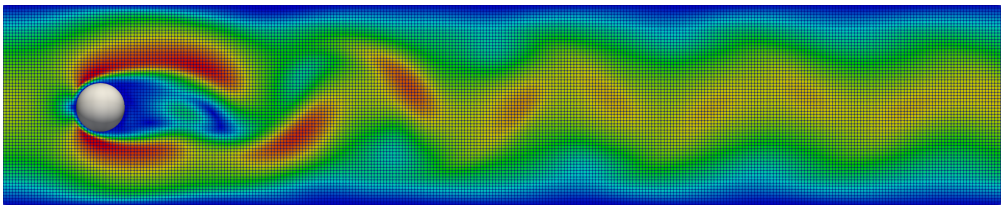


Figure 1.2: A flow profile simulated with CutFEM. The Cartesian computational background mesh is not fitted to the rigid ball's surface.

for instance of fluid–structure interaction, [32]–[34] was proven to work efficiently with CutFEM techniques.
One big advantage of this approach is, that no transformation of the physical domain to a fixed computational reference domain is induced, which reduces the computational costs in comparison to ALE schemes. Another advantage, when using CutFEM, is that no remeshing is necessary, since the background grid is fixed and independent of the underlying problem. One drawback of these CutFEM approaches is, that tiny cut cells can result in nonphysical over- and undershoots of the solution and in a bad conditioning of the underlying system matrix. This can be overcome by adding certain, so-called ghost penalty, stabilization techniques (e.g. [35], [36]). To apply a CutFEM approach one needs:

1. An explicit or implicit description of the underlying domains.
2. A method to integrate over these domains on the fixed reference grid. Basically this leads to the question, how one can perform the integration over just certain subcells, that are cut by the physical domains.
3. An efficient way to stabilize the solution, in order to reduce unphysical oscillations.
4. Treatment of a ghost domain or artificial cut cells in the linear system.

## 1.3 Efficient solver technology

The accurate numerical simulation of finite element discretizations remains to be a challenging task, especially if three space dimensions are considered, due to the sake of physical realism. The efficient solution of the arising algebraic systems of equations with a huge number of unknowns is usually a difficult task. Often, the linear solver represents the limiting factor for the level of mesh resolution and its number of degrees

of freedom. If higher order time discretizations are used, this even puts an additional facet of complexity on the structure of the resulting linear systems and their numerical solution.

When it comes to the discretization of flow problems with mixed format stable pairs of finite elements the situation even gets worse, since the linearization leads to linear systems of equations with saddle point structure. Solving such indefinite saddle point problems has been studied intensively in the literature, cf. [37], [38]. Using a direct solver is a suitable approach for problems of small dimensions. Due to the increase in computational costs and memory, two-dimensional problems that are of interest in practice or even three-dimensional simulations are not feasible for direct linear solvers, even if parallelism is used. In such cases, Krylov subspace methods [39] or multigrid schemes [40] are typically applied.

A classical choice for a Krylov method is the (flexible) generalized minimal residual (GMRES) method. One drawback of the GMRES solver is that an additional amount of memory is allocated in each iteration. As a remedy, a restart that typically leads to a lower rate of convergence can be used; cf. [41]. To improve the convergence of the GMRES method, a preconditioner is typically applied within the GMRES iterations. If the density and viscosity of the flow are constant, the "pressure convection-diffusion" (PCD) block preconditioner results in a mesh independent convergence behavior; cf. [38], [42], [43]). This holds at least for flow problems with low to medium Reynolds numbers; [44]. As an alternative, algebraic multigrid (AMG) methods can be used for preconditioning the GMRES method. For saddle point problems, the AMG preconditioner can not be applied in a natural way. In [45], its application becomes feasible by an appropriate transformation of the underlying saddle point problem.

In challenging benchmarks, e.g., for flow around a cylinder [46]), geometric multigrid (GMG) methods have proven to belong to the most efficient solvers, that are currently available, cf. [40], [47]. Numerical studies showed, that the performance, robustness and efficiency of the GMG methods can be improved further, if they are applied as preconditioners for Krylov iterations; cf. [48], [49]. This combination has shown to work robustly even in challenging three-dimensional simulations [50]. GMG methods have also been applied successfully in two space dimensions along with higher order space-time finite element discretizations of convection-diffusion equations [51], [52] or the Navier–Stokes equations [1], [53], [54]. Applying the GMG method involves several complexities. Firstly, one needs to store the problem structure on various mesh levels and transfer information from finer mesh levels to coarser mesh levels by prolongation operators and vice versa by restriction operators. Secondly, parallel assembly routines add a further layer of complexity to the previous ones. Finally, the key ingredient

of the GMG method, that essentially influences its performance, is the choice of the smoother, that damps high frequency errors on successively coarser mesh levels. The classical Gauss-Seidel smoother is not applicable to the Navier–Stokes equations, due to the saddle point structure of the discrete system. Two popular choices for this kind of problems are the Vanka type smoothers [55] and the Braess–Sarazin type smoothers [56]. In numerical studies, Vanka type smoothers have shown to outperform the Braess-Sarazin ones [47].

## 1.4 Outline

In Chapter 2 we briefly introduce the notation, that is used throughout this work.

Afterwards, in Chapter 3, we apply a higher order space-time finite element method to the Navier–Stokes equations. We study incompressible, viscous flow on an evolving (i.e. time dependent) domain. This is considered a prototype model towards the simulation of full FSI problems. In our setting, the movement of a rigid body is prescribed explicitly by a smooth function in the fluid domain. In our novel approach, we use a fixed background mesh for the whole simulation and perform all the computations there. This has the advantage that the underlying data structure (as the sparsity pattern or the distribution of the degrees of freedom) is fixed and doesn't change throughout the whole simulation. To achieve this, we apply a flexible integration scheme over the fluid domain and implicitly extend the solution to the whole background domain. All the aspects of the implementation are highly parallelized, using the Message Passing Interface (MPI). In various numerical simulations we verify all aspects of this approach and the implementation.

In Chapter 4 a geometric multigrid method, that is used as a preconditioner for generalized minimal residual (GMRES) iterations, is proposed and analyzed computationally for higher order space-time finite element discretizations of the Navier–Stokes equations in two and three space dimensions. The preconditioner is based on a local Vanka smoother, that acts, in contrast to classical finite difference schemes, in each time interval on all space-time degrees of freedom. It is based on a completely parallel implementation, utilizing the *deal.II* C++ library [57]. So the assembly routines, as well as the solver, are both parallelized using the Message Passing Interface (MPI).

In Chapter 5 we computationally analyze a new kind of time integration scheme, that combines continuous Galerkin-Petrov finite element methods with classical collocation

schemes. This combination leads to less complex linear systems and a superior ratio of convergence rate to the needed degrees of freedom. Further the fully discrete solution is of higher regularity in time, which can be advantageous in scenarios where one considers coupled problems and the coefficient of time derivatives of one problem acts as an input in the other problem. The first application of such schemes to the wave equation lead to very promising results, that outperformed classical time integration schemes. Therefore, we expanded the application also to the Navier–Stokes equations, where new problems arose, like the need for initial pressure values and derivatives.

# 2 Function spaces and notation

Here, we introduce the functions spaces and the notation, that are used in this work to present the space-time finite element approaches.

By $L^2(S)$ we denote the function space of square integrable functions on a domain $S$ while $H^1(S)$ is the usual Sobolev space of functions in $L^2(S)$ which have first order weak derivatives in $L^2(S)$. By $S$ we denote either a domain $\Omega$, which is time independent or time dependent. In the time dependent case, the position of the domain is evolving in time, so $\Omega = \Omega(t)$ and we use the abbreviation $\Omega(t) \coloneqq \Omega^t$, cf. Fig. 2.1.

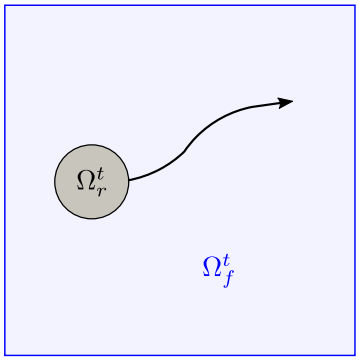



Figure 2.1: A rigid domain $\Omega_r^t$, that is moving inside an evolving fluid domain $\Omega_f^t$.

Further, $\langle \cdot, \cdot \rangle_S$ is the standard inner product of $L^2(S)$. We define the subspace of $L^2(S)$ with mean zero

$$L_0^2(S) \coloneqq \left\{ v \in L^2(S) \mid \int_S v \, \mathrm{d}x = 0 \right\}$$

and the subspace of $H^1(S)$ of functions with zero boundary values (in the sense of traces) on the Dirichlet portion $\Gamma_D \subset \partial S$ of the boundary $\partial S$ of $S$ as $H^1_{0,\Gamma_D}(S)$. Finally, by $H^{1/2}(\Gamma_D)$ we denote the space of all traces on $\Gamma_D \subset \partial S$ of functions in $H^1(S)$.

## Time independent domains

In the case of time independent domains, we define the space of vector fields in $\boldsymbol{L}^2(\Omega)$, where the divergence is also in $\boldsymbol{L}^2(\Omega)$:

$$\boldsymbol{L}^2_{\mathrm{div}}(\Omega) \coloneqq \left\{ \boldsymbol{v} \in \boldsymbol{L}^2(\Omega) \mid \nabla \cdot \boldsymbol{v} \in \boldsymbol{L}^2(\Omega) \right\}.$$

By $\boldsymbol{L}^2_{\mathrm{div},0,\Gamma_D}(\Omega)$, we denote the space of vector fields in $\boldsymbol{L}^2_{\mathrm{div}}(\Omega)$, where $\nabla \cdot \boldsymbol{v} = 0$ with zero boundary values (in the sense of traces) for the normal component of $\boldsymbol{v}$ on the Dirichlet portion $\Gamma_D \subset \partial\Omega$, so $\boldsymbol{v} \cdot \boldsymbol{n} = 0$ on $\Gamma_D$.
Next we let

$$\boldsymbol{V}_{\mathrm{div}} \coloneqq \left\{ \boldsymbol{v} \in \boldsymbol{H}^1(\Omega) \;\middle|\; \nabla \cdot \boldsymbol{v} = 0 \text{ in } \Omega \right\}$$

and define the following spaces:

$$\begin{aligned} V_I &\coloneqq \left\{ \boldsymbol{v} \in L^2\Big(I; \boldsymbol{H}^1(\Omega) \cap \boldsymbol{V}_{\mathrm{div}}\Big) \mid \partial_t \boldsymbol{v} \in L^2\Big(I; \boldsymbol{H}^{-1}(\Omega)\Big) \right\}, \\ V_{0,I} &\coloneqq \left\{ \boldsymbol{v} \in L^2\Big(I; \boldsymbol{H}^1_{0,\Gamma_D}(\Omega)\Big) \right\}, \end{aligned} \tag{2.1}$$

where $\boldsymbol{H}^{-1}(\Omega)$ is the dual space of $\boldsymbol{H}^1(\Omega)$. Next we define

$$L^2_{0,I} \coloneqq L^2\Big(I; L^2_0(\Omega)\Big), \qquad L^\infty_I \coloneqq L^\infty\Big(I; \boldsymbol{L}^2_{\mathrm{div}}(\Omega)\Big). \tag{2.2}$$

## Evolving domains

In the case of domains, which are evolving in time, we first let

$$\boldsymbol{V}^t_{\mathrm{div}} \coloneqq \left\{ \boldsymbol{v} \in \boldsymbol{H}^1\left(\Omega^t\right) \;\middle|\; \nabla \cdot \boldsymbol{v} = 0 \text{ in } \Omega^t \right\}.$$

Next we will make use of the following spaces:

$$\begin{aligned} V_I &\coloneqq \left\{ \boldsymbol{v} \in L^2\Big(I; \boldsymbol{H}^1\left(\Omega^t\right) \cap \boldsymbol{V}^t_{\mathrm{div}}\Big) \;\middle|\; \partial_t \boldsymbol{v} \in L^2\Big(I; \boldsymbol{L}^2\left(\Omega^t\right)\Big) \right\}, \\ V_{0,I} &\coloneqq \left\{ \boldsymbol{v} \in L^2\left(I; \boldsymbol{H}^1_{0,\Gamma^t_D}\left(\Omega^t\right)\right) \right\}. \end{aligned} \tag{2.3}$$

In the definition of $V^t_I$ in (2.3), the weak partial derivative with respect to the time variable $t$ is defined as an element of $\boldsymbol{L}^2(\Omega^t_I)$ for the space-time domain $\Omega^t_I \coloneqq \bigcup_{0 \le t \le T} \Omega^t \times \{t\}$.

We also define

$$L^2_{0,I} \coloneqq L^2\left(I; L^2_0\left(\Omega^t\right)\right) , \qquad L^\infty_I \coloneqq L^\infty\left(I; \boldsymbol{H}^1\left(\Omega^t\right)\right) . \tag{2.4}$$

## Space discretization

For the space discretization, let $\mathcal{T}_h = \{T\}$ be a family of shape-regular decompositions of the domain $\Omega$ into (open) quadrilaterals or hexahedrals $K$ with maximum cell size $h$. For $r \geq 0$, let $\hat{\mathbb{Q}}_r$ denote the space of polynomials of degree at most $r$ in each variable and $\hat{\mathbb{P}}_r$ the space of at most degree $r$. We put

$$\mathbb{Q}_r(T) \coloneqq \left\{ v_{h|T} = \hat{v}_h \circ F_T^{-1} : \hat{v}_h \in \hat{\mathbb{Q}}_r \right\} , \quad \mathbb{P}_r(T) \coloneqq \left\{ v_{h|T} = \hat{v}_h \circ F_T^{-1} : \hat{v}_h \in \hat{\mathbb{P}}_r \right\} ,$$

with the reference mapping $F_T$ from the reference cell $\widehat{T}$ to element $T$.

We define the finite element spaces

$$H^r_h \coloneqq \left\{ v_h \in C(\overline{\Omega}) \mid v_{h|T} \in \mathbb{Q}_r(T) \; \forall \; T \in \mathcal{T}_h \right\} , \tag{2.5}$$

$$H^r_{h,\mathrm{disc}} \coloneqq \left\{ v_h \in L^2(\Omega) \mid v_{h|T} \in \mathbb{P}_r(T) \; \forall \; T \in \mathcal{T}_h \right\} . \tag{2.6}$$

For the flow problems in this work we will make use of the Taylor–Hood family of finite element functions. For some natural number $r \geq 2$ we then put

$$V_h = H^{(r)}_h , \quad V^0_h = H^{(r)}_h \cap H^1_{0,\Gamma}(\Omega) , \quad Q_h = H^{(r-1)}_h \tag{2.7}$$

and

$$X_h \coloneqq V_h \times Q_h , \quad X^0_h \coloneqq V^0_h \times Q_h ,$$

as well as

$$\boldsymbol{V}_h = (V_h)^{dim} , \quad \boldsymbol{V}^0_h = (V^0_h)^{dim} , \quad \boldsymbol{X}_h = \boldsymbol{V}_h \times Q_h , \quad \boldsymbol{X}^0_h = \boldsymbol{V}^0_h \times Q_h ,$$

where $dim \in 2, 3$ stands for the spatial dimension of the problem. In the same way we also introduce the conforming, inf-sup stable element pairs with discontinuous pressure elements. With

$$Q_{h,\mathrm{disc}} = H^{(r-1)}_{h,\mathrm{disc}}$$

and

$$X_{h,\mathrm{disc}} := V_h \times Q_{h,\mathrm{disc}}\,, \quad X^0_{h,\mathrm{disc}} := V^0_h \times Q_{h,\mathrm{disc}}\,,$$

we define

$$\boldsymbol{X}_{h,\mathrm{disc}} = \boldsymbol{V}_h \times Q_{h,\mathrm{disc}}\,, \quad \boldsymbol{X}^0_{h,\mathrm{disc}} = \boldsymbol{V}^0_h \times Q_{h,\mathrm{disc}}\,. \tag{2.8}$$

The discrete space of weakly divergence free functions is denoted by

$$\boldsymbol{V}^{\mathrm{div}}_h = \left\{\boldsymbol{v}_h \in \boldsymbol{V}_h \mid \langle \nabla \cdot \boldsymbol{v}_h, q_h\rangle = 0 \text{ for all } q_h \in Q_{h(,\mathrm{disc})}\right\}\,.$$

Finally, we define the spaces

$$V_{I,h} \coloneqq \{\boldsymbol{v}_h \in L^2(I;\boldsymbol{V}_h) \mid \partial_t \boldsymbol{v} \in L^2(I;\boldsymbol{V}_h)\}\,,$$

$$L^2_{0,I,h} \coloneqq L^2(I;Q_{h(,\mathrm{disc})})\,.$$

## Time discretization

For the time discretization we decompose the time interval $I = (0,T]$ into $N$ subintervals $I_n = (t_{n-1}, t_n]$, where $n \in \{1,\dots,N\}$ and $0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T$ such that $I = \bigcup_{n=1}^N I_n$. We put $\tau = \max_{n=1,\dots N} \tau_n$ with $\tau_n = t_n - t_{n-1}$. Further, the set of time intervals $\mathcal{M}_\tau := \{I_1,\dots,I_n\}$ is called the time mesh. For a Banach space $B$, we let $L^2(0,T;B)$, $C([0,T];B)$, and $C^m([0,T];B)$, $m \in \mathbb{N}$, be the Bochner spaces of $B$-valued functions, equipped with their natural norms. For a subinterval $J \subseteq [0,T]$, we use the notations $L^2(J;B)$, $C^m(J;B)$, and $C^0(J;B) := C(J;B)$. Further, for a Banach space $B$ and any $k \in \mathbb{N}$, we let

$$\mathbb{P}_k(I_n;B) = \left\{ w_\tau : I_n \mapsto B \;\middle|\; w_\tau(t) = \sum_{j=0}^{k} W^j t^j\,,\ \forall t \in I_n\,,\ W^j \in B\ \forall j \right\}. \tag{2.9}$$

For an integer $k \in \mathbb{N}$ we introduce the space of globally continuous functions in time

$$X^k_\tau(B) := \left\{w_\tau \in C(\overline{I};B) \mid w_{\tau|I_n} \in \mathbb{P}_k(I_n;B)\ \forall I_n \in \mathcal{M}_\tau\right\}\,, \tag{2.10}$$

and for an integer $l \in \mathbb{N}_0$ the space of globally $L^2$-functions in time

$$Y^l_\tau(B) := \left\{w_\tau \in L^2(I;B) \mid w_{\tau|I_n} \in \mathbb{P}_l(I_n;B)\ \forall I_n \in \mathcal{M}_\tau\right\}\,.$$

We will also make use of the abbreviation for the discrete space-time function spaces

$$\boldsymbol{X}_{\tau,h}^{k} \coloneqq (X_{\tau}^{k}(V_h))^{dim} \times X_{\tau}^{k}(Q_h)\,, \quad \boldsymbol{Y}_{\tau,h}^{k} = (Y_{\tau}^{k}(V_h))^{dim} \times Y_{\tau}^{k}(Q_h) \tag{2.11}$$

and

$$\boldsymbol{X}_{\tau,h}^{k,0} \coloneqq (X_{\tau}^{k}(V_h^0))^{dim} \times X_{\tau}^{k}(Q_h)\,, \quad \boldsymbol{Y}_{\tau,h}^{k,0} = (Y_{\tau}^{k}(V_h^0))^{dim} \times Y_{\tau}^{k}(Q_h)\,, \tag{2.12}$$

with $dim \in 2, 3$.

# 3 Cut finite element methods for flows problems on evolving domains

Many flow problems of practical relevance, for example in engineering applications, are time-dependent problems of multi–physics. Typically, these problems consist of multiple spatial domains, which represent areas with different physical behavior, that can be described by partial differential equations. One prominent example is the interaction of a fluid with a solid structure, which is called fluid–structure interaction (FSI). When the fluid encounters a solid body, it causes stresses and strains in the solid body, whose size depends on the properties of the flow. If the deformation of the solid body is negligible, one can model the structure as a rigid body, which doesn't change its form. Otherwise the solid body is modeled as an elastic body with various material models. In many applications the domains are not fixed. Often a body, that is assumed to be rigid, moves in a fluid (for example when simulating wind turbines), so we have an evolving domain or in case of multiple such rigid bodies moving domains. The movement of the rigid body in contrast leads to a change of the fluid domain, so the fluid domain is evolving. If the solid body is elastic the structure can also be an evolving domain, change its shape and deform depending on the fluid flow properties. Solving such problems is still a very challenging task, especially, if the movement of the underlying domains is large.

Cut finite element methods (CutFEM) are suitable, when it comes to solving partial differential equations on such evolving domains with moving boundaries. They are subject to active current research and have been analyzed and studied numerically for parabolic problems, using low order finite elements and BDF time stepping schemes; cf. e.g. [25]. This analysis was recently extended to the time-dependent Stokes problem using, for the spatial discretization, lowest order Taylor-Hood elements [26] or equal-order elements along with stabilization [27], combined with BDF time-stepping schemes. For instance, see [26, Thm. 5.16] for a rigorous error estimate. Also, the simulation of incompressible flow on complex domains [28]–[31] or even the simulation of coupled problems of multi-physics, for instance of fluid–structure interaction, with moving interfaces [32]–[34] becomes feasible with CutFEM based techniques. The ap-

preciable advantage of the CutFEM approach is, that all the computations can be done on a time-independent, fixed background mesh. However, mesh cells then are intersected in an arbitrary manner by moving boundaries of the domain or inner interfaces, for instance, if fluid-structure interaction is studied. In contrast to the well-known Arbitrary–Lagrangian–Eulerian (ALE) method (cf. [10], [11], [14]), no transformation of the physical domain to a fixed computational reference domain is induced by CutFEM and the computational mesh is not necessarily fitted to the outer boundaries of the domain or, for instance, the boundaries of enclosed rigid domains if flow around obstacles is studied. In classical ALE approaches, larger motions and deformations of the domains lead to a poor quality of the non-transformed, physical mesh (cf., e.g. [15]–[17]), since the computational mesh has to track and resolve moving boundaries or interfaces. This mesh deformation impacts the transformation of the model equations to the reference domain and, thereby, the stability of the overall ALE approach. To preserve the mesh quality and stability of the transformation, remeshing becomes inevitable. (cf. [18]). Fixed-mesh ALE approaches (cf. [58]) try to overcome this issue by combining a fixed background mesh with the ALE concept. In each time step, the discrete functions are projected onto a fixed background mesh, so that additional degrees of freedom, used to preserve the mesh quality by introducing supplementary grid nodes, are naturally defined by the ALE approach.

The CutFEM approach overcomes the ALE transformation of the model equations to a reference domain. In CutFEM, the model is discretized on a fixed background mesh in its native formulation. The CutFEM methodology is based on three key ingredients: Firstly, boundary conditions of Dirichlet type are imposed in a weak variational form by using Nitsche's method, cf. [59]–[63]. Secondly, a flexible and efficient approach to integrate over all types of mesh cells resulting from intersections of computational grid cells by moving boundaries, referred to as cut cells, is needed. The efficient integration over such cut cells requires the computation of volume integrals over portions of the underlying finite element cells. This integration should not disturb the convergence order of the underlying space-time discretization scheme. One way to do this is to divide the cut cells into sub-elements and to apply a standard quadrature formula on the resulting triangulation; cf. [64], [65]. However, in this approach the mesh information has to be recomputed or rearranged for assembling the algebraic system, which counteracts the methodology and advantages of CutFEM techniques. There exist some remedies for this drawback, for instance, by using a refined sub-mesh for the integration over the cut cells, without adjusting the underlying mesh and degrees of freedom (cf. [66], [67]), which however is still a challenging task, in particular in the case of three

space dimensions. An alternative is given by using the divergence theorem and transforming the volume integrals to surface integrals (line integrals in the two-dimensional case). The latter ones can then be computed by a moment fitting method (cf. [68]) or suitable quadrature formulas (cf. [69]). Our approach proposes a different technique, that deviates from the previous ones. Iterated integrals of multivariate calculus are computed over all portions of cells intersected by the domain's boundaries by applying (iterated) one-dimensional quadrature formulas. This results in a flexible and efficient algorithm.

The third and last ingredient of CutFEM techniques for evolving domains is the extension of discrete functions to fictitious or ghost subdomains or the entire computational mesh, respectively, along with the ghost penalization of the discrete variational equations; cf. [25]–[27], [29], [70]. The stabilization of the variational problem aims at reducing unphysical and spurious oscillations that are due to unavoidable irregular (small) cuts of mesh cells by moving boundaries. A strong analogy to the discretization of convection-dominated problems (cf. [71]) can be observed. Further, the stabilization also aims at improving the conditioning of the resulting algebraic system; cf. [25], [35], [63]. Proposed stabilization techniques are based on suitable extensions of the discrete functions to some neighborhood of the flow domain; cf. [26], [72]. We refer to this extension of the flow domain as a fictitious or ghost subdomain, since the physical unknowns are not defined in the domain's extension by some mathematical problem. Usually, the prolongation of the discrete functions is restricted to some thin layer adjacent to the domain's boundary; cf. [25], [26], [73], [74]. However, in our approach we extend all quantities to the entire computational mesh. More precisely, for our prototype model problem of flow around a moving rigid obstacle in a pipe (cf. Fig. 3.1) this means that the velocity and pressure variable are extended to the entire rigid obstacle. The appreciable advantage of this augmented extension is that the underlying degrees of freedom and mesh information are fixed over the whole simulation time. This frees us from an expensive remeshing, a redistribution of degrees of freedom and a rebuilding or updating of the sparse system matrix data structure. For all time steps of the considered time interval, a non-condensed system matrix for all degrees of freedom of the entire computational grid, as the union of the flow domain and fictitious or ghost subdomains, is computed. No adaption of the matrix's data structure with respect to active degrees of freedom of the flow domain and non-active or fictitious degrees of freedom of the ghost subdomains, varying in time due to the evolving domain, is required. As concerns the stabilization itself, our approach differs from the popular derivative jump ghost penalization technique used in, e.g., [21], [29],

[32], [73], [75]. In this approach jumps in the (higher order) derivatives are penalized over facets which requires the computationally expensive evaluation of all higher order derivatives up to the maximum polynomial degree. We adopt a computationally cheaper and more direct concept of ghost penalization suggested in [25], [36] for linear convection-diffusion-reaction equations and generalized in [26] to the Stokes system. In this approach, no derivatives are required and the penalization is not depending on the polynomial degree of the spatial finite element spaces, just on some scaling factors in terms of negative powers of the spatial mesh size and the viscosity. This direct stabilization is carefully adapted to the mathematical setting of the Navier–Stokes system with two unknown variables and their non-equal order, inf-sup stable approximation in space. The stability of the proposed choice of the scaling factors in terms of the spatial mesh size is demonstrated.

Finally, we mention that an arbitrary order, discontinuous Galerkin method is applied for the discretization of the time variable. In [23], [75], a piecewise linear and discontinuous in time discretization, that is combined with CutFEM techniques, is developed. The integration over the cut cells follows similar ideas as used here, but differs by subdividing the flow region of the cut cells into triangles (in two space dimensions) and integrating over these triangles whereas iterated integration over the (partially) curved subcell is applied here. Our motivation for using a discontinuous approximation in time is to overcome some difficulty in the pressure approximation of equal-order and continuous in time discretizations of Navier–Stokes solutions. Precisely, the latter lacks from the availability of an (continuous and discrete) initial value for the pressure variable that is not provided explicitly by the Navier–Stokes system, but needed for the unique definition of the full pressure trajectory. We note that discrete pressure approximations of a continuous in time approximation, for instance in the Gauss quadrature nodes of a Gauss quadrature formula in time, become accessible without any initial pressure values, which however is not sufficient to compute the complete pressure trajectory. The discontinuous discretization in time can be applied in a natural way to time-dependent domains and combined with the extension of discrete functions to fictitious domains and the integration over cut cells. By using a discontinuous test basis, we reduce the space-time formulation to a time-marching scheme. Appreciable advantage of this approach is that algebraic systems of reasonable dimension are obtained. However, for higher order time discretizations the block structure of the resulting algebraic system is still complex and requires highly efficient algebraic solvers. For the simulations presented in Sec. 3.5, we use a parallel direct solver. This is done in order to separate the effects of the CutFEM techniques from the influence of the solver. In Chapter 4 we ad-

dress the implementation of a geometric multigrid preconditioner for an outer GMRES solver for such type of problems. Throughout the whole chapter, we restrict ourselves to the two dimensional case. Transferring the ideas to three spatial dimensions should be possible, but is more involved, especially due to the more complex cuts, that can arise there.

In our numerical studies in Sec. 3.5, we consider a sequence of six test problems of increasing complexity to analyze the stability and performance properties of the proposed approach. Firstly, its convergence properties are analyzed for a time-independent domain. Secondly the influence of the size of the ghost penalty zone is computationally analyzed in a static scenario. Thirdly, we investigate the convergence behavior in a time-dependent domain. Fourthly the DFG benchmark [46] of flow around a cylinder in two space dimensions is considered. Even though the domain is time-independent, it's worth to analyze the accuracy of the proposed approach for this well-understood setting and to compare it with discretizations based on body-fitted techniques. Thereby, our integration over cut cells, the extension of discrete functions to fictitious (ghost) subdomains of the computational mesh and the ghost penalization are evaluated. In the fifth and sixth experiment, time-independent domains are studied again. Flow around moving rigid bodies is investigated and illustrated for two different settings.

## 3.1 Problem description

We start by giving a brief derivation of the Navier–Stokes equations in Sec. 3.1.1. Next we sketch the mathematical problem in Sec. 3.1.2. Afterwards we make some assumptions on the CutFEM space discretization in Sec. 3.2.1.

### 3.1.1 Derivation of the Navier–Stokes equations

For completeness we briefly want to give a derivation for the Navier–Stokes equations. At first we need the conservation of mass, which is expressed through the mass continuity equation

$$\partial_t \rho + \nabla \cdot (\rho \boldsymbol{v}) = 0\,, \tag{3.1}$$

with the density $\rho$ and the fluid velocity $\boldsymbol{v}$. Since we assume that the flow is incompressible, $\rho$ is constant in time and space. Therefore, the first term in Eq. (3.1) vanishes and we get

$$\nabla \cdot \boldsymbol{v} = 0\,. \tag{3.2}$$

We further make use of the balance of momentum, which is described by the so called momentum equation

$$\rho \partial_t \boldsymbol{v} + \rho(\boldsymbol{v} \cdot \nabla)\boldsymbol{v} - \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{v}, p) = \rho \boldsymbol{f}\,. \tag{3.3}$$

Here $\boldsymbol{\sigma}(\boldsymbol{v}, p)$ denotes the stress tensor, $p$ the fluid pressure and $\boldsymbol{f}$ external forces. In this work we only consider Newtonian fluids, where the viscous stresses are linearly correlated to the strain. Therefore, we can simplify the stress tensor to

$$\boldsymbol{\sigma}(\boldsymbol{v}, p) = (-p + \lambda \nabla \cdot \boldsymbol{v})\mathbb{I} + \mu(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^\top)\,, \tag{3.4}$$

where $\lambda$ is the first and $\mu$ is the second Lamé parameter, that denote fluid parameters, which are assumed to be constant in this work. Since we assume incompressible flows we make use of Eq. (3.2) and the $\lambda \nabla \cdot \boldsymbol{v}$ term in Eq. (3.4) vanishes. Using this, Eq. (3.3) can then be expressed as

$$\rho \partial_t \boldsymbol{v} + \rho(\boldsymbol{v} \cdot \nabla)\boldsymbol{v} + \nabla p - \mu \nabla \cdot (\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^\top) = \rho \boldsymbol{f}\,. \tag{3.5}$$

Dividing this equation by $\rho$ leads to

$$\partial_t \boldsymbol{v} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} + \frac{1}{\rho}\nabla p - \nu \nabla \cdot (\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^\top) = \boldsymbol{f}\,. \tag{3.6}$$

Here $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity.

For a dimensionless formulation we set $\rho \equiv 1$ and combine Eq. (3.6) with Eq. (3.2), to get the Navier–Stokes equations

$$\begin{aligned} \partial_t \boldsymbol{v} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} - \nu \nabla \cdot (\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^\top) + \nabla p &= \boldsymbol{f}\,, \\ \nabla \cdot \boldsymbol{v} &= 0\,. \end{aligned} \tag{3.7}$$

Now we make use of the vector identity $\nabla \cdot (\nabla \boldsymbol{v})^\top = \nabla(\nabla \cdot \boldsymbol{v})$. With Eq. (3.2) this results in the fact, that the $\nabla \boldsymbol{v}^\top$ term in Eq. (3.7) vanishes. With $\nabla \cdot (\nabla \boldsymbol{v}) = \Delta \boldsymbol{v}$ we can rewrite the Navier–Stokes equations as

$$\begin{aligned} \partial_t \boldsymbol{v} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} - \nu \Delta \boldsymbol{v} + \nabla p &= \boldsymbol{f}\,, \\ \nabla \cdot \boldsymbol{v} &= 0\,. \end{aligned} \tag{3.8}$$

As it's pointed out in [76], both strong forms of the Navier–Stokes equations, Eq. (3.7) and Eq. (3.8), are equivalent. But this is only valid on the assumption, that the velocity solution is divergence free. Since this is not always true on a discrete level, the discrete solutions of Eq. (3.7) and Eq. (3.8) might differ (see [77]).

### 3.1.2 Prototype flow problem on evolving domains

Without loss of generality, in this work we study a prototype model problem that is sketched in Fig. 3.1. Restricting ourselves to considering incompressible viscous flow around a moving, indeformable, rigid body in a pipe is only done in order to simplify the notation, reduce technical ballast and focus on the essential features of the numerical techniques. The problem is of high interest in practice and can be regarded as a test problem for more sophisticated applications, for instance, as a building block for fluid-structure interaction. By $\Omega = (0, L) \times (0, W)$, with some $L > 0$ and $W > 0$, we denote the rectangular pipe of length $L$ and width $W$. Let $T > 0$ be the final time. About the rigid body $\Omega_r(t)$ we make the following assumption.

**Assumption 3.1.** *The smooth motion of the bounded rigid body $\Omega_r^t := \Omega_r(t)$, with positive measure $meas_2(\Omega_r^t) > 0$ and sufficiently smooth boundary $\Gamma_r^t := \partial\Omega_r(t)$, for $t \in [0, T]$, is supposed to be given. Contact between the pipe boundary $\Gamma_p := \Gamma_i \cup \Gamma_w \cup \Gamma_o$ and the rigid body is assumed not to occur such that $\overline{\Omega_r^t} \subset \Omega$ and, with some suitable constant $\delta > 0$,*

$$dist(\Gamma_r^t, \Gamma_p) \geq \delta > 0\,, \quad \text{for } t \in [0, T]\,.$$

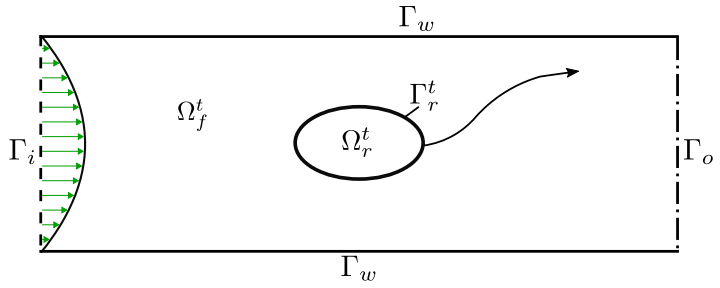



Figure 3.1: Problem setting and corresponding notation with pipe's boundary $\Gamma_p := \Gamma_i \cup \Gamma_w \cup \Gamma_o$.

By $\Omega_f^t := \Omega_f(t) = \Omega \backslash \overline{\Omega_r^t}$, with $\Omega_f^t \subset \Omega$, we denote the open domain filled with fluid. For $\Omega_f^t$, with $t \in [0, T]$, and $I := (0, T]$ we consider solving the incompressible Navier–Stokes system

$$\partial_t \boldsymbol{v} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} - \nu \Delta \boldsymbol{v} + \nabla p = \boldsymbol{f} \quad \text{in } \Omega_f^t \times I\,, \tag{3.9a}$$

$$\nabla \cdot \boldsymbol{v} = 0 \quad \text{in } \Omega_f^t \times I\,, \tag{3.9b}$$

$$\boldsymbol{v} = \boldsymbol{0} \quad \text{on } \Gamma_w \times I\,, \tag{3.9c}$$

$$\boldsymbol{v} = \boldsymbol{g}_i \quad \text{on } \Gamma_i \times I\,, \tag{3.9d}$$

$$\boldsymbol{v} = \boldsymbol{g}_r \quad \text{on } \Gamma_r^t \times I\,, \tag{3.9e}$$

$$\nu \nabla \boldsymbol{v} \cdot \boldsymbol{n} - \boldsymbol{n} p = \boldsymbol{0} \quad \text{on } \Gamma_o \times I\,, \tag{3.9f}$$

$$\boldsymbol{v}(0) = \boldsymbol{v}_0 \quad \text{in } \Omega_f^t\,. \tag{3.9g}$$

In Eq. (3.9), the velocity field $\boldsymbol{v}$ and the pressure $p$ are the unknown variables. In Eq. (3.9a), the parameter $\nu > 0$ denotes the fluid's viscosity and the right-hand side function $\boldsymbol{f}$ is a given external force. In Eq. (3.9f), the field $\boldsymbol{n}$ is the outer unit normal vector. In Eq. (3.9g), the function $\boldsymbol{v}_0$ denotes the prescribed initial velocity. On $\Gamma_i$, an inflow flow profile is prescribed by Eq. (3.9d). The boundary parts $\Gamma_w$ are fixed walls with no-slip boundary (3.9c), and $\Gamma_o$ represents an outflow boundary that is modeled by the do-nothing boundary condition (3.9f); cf. [78]. The boundary condition (3.9e) ensures that the fluid follows the prescribed rigid body's motion $\boldsymbol{g}_r$ on $\Gamma_r^t$. Here, $\boldsymbol{g}_r$ is the prescribed material velocity of the particles from $\Gamma_r^t$. To simplify the notation, we put

$$\boldsymbol{g} = \begin{cases} \boldsymbol{0} & \text{on } \Gamma_w \times I\,, \\ \boldsymbol{g}_i & \text{on } \Gamma_i \times I\,, \\ \boldsymbol{g}_r & \text{on } \Gamma_r^t \times I \end{cases}$$

and, with $\Gamma_D^t := \Gamma_i \cup \Gamma_w \cup \Gamma_r^t$, we rewrite the conditions (3.9c) to (3.9e) by

$$\boldsymbol{v} = \boldsymbol{g} \quad \text{on } \Gamma_D^t \times I\,.$$

We assume, that a unique (weak) solution of the system (3.9) exists. This implies that sufficient smoothness conditions about the evolving flow domain are satisfied. For the existence and uniqueness of solutions to the Stokes and Navier–Stokes system on time-dependent domains we refer to, e.g., [27], [79], [80]. In particular, we refer to [79, p. 156, Thm. 1, p. 164, Cor.] for the existence of local in time solutions to (3.9) or solutions for sufficiently small data. Even though solutions to (3.9) might lack (due to the polygonal pipe boundary) higher order regularity, that is needed to expect an optimal order convergence behavior of higher order discretization methods, members of such families are applied here. In the numerical approximation of partial differential equations it is widely accepted that higher order methods achieve accurate results on computationally feasible grids, even if the solution of the mathematical problem is not sufficiently smooth on the whole space-time domain. This also applies to the lack of regularity of Navier–Stokes solutions for $t \to 0$ under realistic assumptions about the data of the problem (cf. [81]–[83]) which is not considered here.

For the weak problem formulation we introduce the semi-linear form $A_S : (\boldsymbol{H}^1(S) \times L_0^2(S)) \times (\boldsymbol{H}_{0,\Gamma}^1(S) \times L_0^2(S)) \to \mathbb{R}$ by

$$A_S((\boldsymbol{v},p),(\boldsymbol{\psi},\xi)) := \langle(\boldsymbol{v}\cdot\boldsymbol{\nabla})\boldsymbol{v},\boldsymbol{\psi}\rangle_S + \nu\langle\nabla\boldsymbol{v},\nabla\boldsymbol{\psi}\rangle_S - \langle p,\nabla\cdot\boldsymbol{\psi}\rangle_S + \langle\boldsymbol{\nabla}\cdot\boldsymbol{v},\xi\rangle_S\,, \tag{3.10}$$

for $(\boldsymbol{v},p) \in \boldsymbol{H}^1(S) \times L_0^2(S)$ and $(\boldsymbol{\psi},\xi) \in \boldsymbol{H}_{0,\Gamma}^1(S) \times L_0^2(S)$. For given $\boldsymbol{f} \in \boldsymbol{L}^2(S)$ we introduce the linear form $L : \boldsymbol{H}^1(S) \to \mathbb{R}$ by

$$L_S(\boldsymbol{\psi};\boldsymbol{f}) := \langle\boldsymbol{f},\boldsymbol{\psi}\rangle_S\,, \tag{3.11}$$

for $\boldsymbol{\psi} \in \boldsymbol{H}_{0,\Gamma}^1(S)$.

For the analysis of solutions to (3.9) or the discrete scheme a more sophisticated framework is needed. The proof of well-posedness and error analyses are more involved than in the case of fixed domains. In particular, a diffeomorphism mapping the evolving domain to a reference domain and the transformation of the model equations to an equivalent system on the reference domain by the diffeomorphism are utilized strongly. For further details we refer to, e.g., [27] for the Stokes system and to [84] for an abstract framework for parabolic partial differential equations on evolving spaces.

## 3.2 Space–time finite element discretization with CutFEM

Here we introduce our approximation of the Navier–Stokes system (3.9) by combining CutFEM and variational time discretization techniques. Firstly, we introduce the variational formulation of the Navier–Stokes system (3.9). The next subsection is devoted to the discretization in space. Nitsche's method [61] is applied to incorporate Dirichlet boundary conditions in a weak form. Further, the ghost penalty stabilization along with the extension of the discrete in space functions to fictitious subdomains of fluid flow, i.e. to the domain of the moving rigid body here, are introduced. Together, the stabilization and the extension provide an implicit definition of the discrete functions on the background mesh for the time-independent pipe domain $\Omega$. The stabilization reduces spurious oscillations by irregular cuts of finite elements by the moving rigid body $\Omega_r^t$ and improves the conditioning of the resulting algebraic system; cf. [25]. Finally, the discretization in time by the discontinuous Galerkin method is done and the resulting fully discrete problem is presented.

### 3.2.1 Assumptions about the space discretization

Let $\mathcal{T}_h = \{K\}$ be a family of shape-regular, structured decompositions of the pipe $\Omega$ (cf. Fig. 3.1) into (open) quadrilaterals $K$ with maximum cell size $h$. Precisely, $\mathcal{T}_h$ is the computational background mesh. It is not fitted to the boundary of the moving rigid body $\Omega_r^t$; cf. Fig. 3.2. We make the following assumptions.

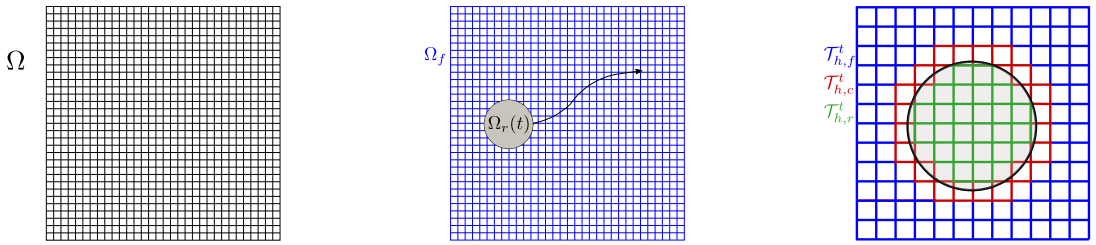
**Assumption 3.2.** *The family $\mathcal{T}_h = \{K\}$ satisfies:*

1. *The background mesh is independent of the time $t$.*
2. *The mesh is cartesian, such that the cells are aligned along the coordinate lines.*
3. *Each face of a quadrilateral is cut at most once by the rigid body, cf. Remark 3.9.*

The opportunity to use structured meshes facilitates their generation which is an appreciable advantage of CutFEM. The third of the assumptions is made in order to

simplify the implementation when it comes to the integration over cut cells. For time-independent domains, this condition is always fulfilled, if the mesh size is chosen sufficiently small with respect to the size of the rigid body. For evolving domains, this condition can be violated, if $\Omega_r^t$ has a curved boundary. In our implementation we ensure, that the condition of Item 3 is always fulfilled, cf. Sec. 3.4 for details.

The time-dependent set of mesh cells $K \in \mathcal{T}_h$ that are subsets of the fluid domain $\Omega_f^t$ is denoted by $\mathcal{T}_{h,f}^t$, the mesh cells $K \in \mathcal{T}_h$ that are subsets of the rigid domain $\Omega_r^t$ is denoted by $\mathcal{T}_{h,r}^t$. The set of cut cells $K \in \mathcal{T}_h$ such that $K \cap \Omega_f^t \neq \emptyset$ and $K \cap \Omega_r^t \neq \emptyset$ is denoted by $\mathcal{T}_{h,c}^t$; cf. Fig. 3.2c.




(a) Computational domain $\Omega$ and background mesh $\mathcal{T}_h$.

(b) Fluid domain $\Omega_f^t$, rigid domain $\Omega_r^t$ and unfitted background mesh.

(c) Submesh of fluid cells $\mathcal{T}_{h,f}^t$, rigid cells $\mathcal{T}_{h,r}^t$ and cut cells $\mathcal{T}_{h,c}^t$.

Figure 3.2: CutFEM mesh topology with computational background mesh and submeshes of fluid, cut and rigid cells.

For the numerical experiments that are presented in Sec. 3.5 we used the Taylor–Hood family of inf-sup stable finite element pairs for the space discretization (see Chapter 2). These elements can be replaced by any other type of inf-sup stable elements (cf. [77]), when the proposed ghost penalty stabilization of Sec. 3.2.3 is applied.

### 3.2.2 Variational formulation of the continuous problem

A sufficiently regular solution of the Navier–Stokes system (3.9) satisfies the following variational problem that provides the basis for the space-time finite element discretizations.

**Problem 3.1** (Continuous problem)**.** *Let* $\boldsymbol{f} \in \boldsymbol{L}^2(\Omega_{f,I})$ *and* $\boldsymbol{v}_0 \in \boldsymbol{H}^1_{0,\Gamma_D^t}\left(\Omega_f^0\right) \cap \boldsymbol{V}^0_{div}$ *be given. Let* $\hat{\boldsymbol{g}} \in V_I^t \cap L_I^\infty$ *denote a prolongation of the boundary values on* $\Gamma_D^t$ *to* $\Omega_f^t$*,*

*such that* $\hat{\boldsymbol{g}} = \boldsymbol{g}$ *on* $\Gamma_D^t \times I$. *Find* $(\boldsymbol{v}, p) \in (V_I \cap L_I^\infty) \times L_{0,I}^2$, *with* $\boldsymbol{v} \in \hat{\boldsymbol{g}} + V_{0,I}$, *such that* $\boldsymbol{v}(0) = \boldsymbol{v}_0$ *and for all* $\boldsymbol{\phi} \coloneqq (\boldsymbol{\psi}, \xi) \in V_{0,I} \times L_{0,I}^2$,

$$\int_0^T \langle \partial_t \boldsymbol{v}, \boldsymbol{\psi} \rangle_{\Omega_f^t} + A_{\Omega_f^t}((\boldsymbol{v}, p), (\boldsymbol{\psi}, \xi)) \mathrm{d}t = \int_0^T L_{\Omega_f^t}(\boldsymbol{\psi}; \boldsymbol{f}) \mathrm{d}t \,. \tag{3.12}$$

**Remark 3.1.** *Regarding the well-posedness of Problem 3.1 we note the following.*

- *In [79, p. 156, Thm. 1], the existence and uniqueness of local in time weak solutions to Eqs.* (3.9a), (3.9b), (3.9g) *equipped with homogeneous Dirichlet boundary conditions is shown under sufficient regularity assumptions about the data and motion of the evolving domain. The technical details of the assumptions about the domain motion are skipped here. In particular,* $\boldsymbol{v} \in L_I^\infty$ *and* $\partial_t \boldsymbol{v} \in \boldsymbol{L}^2(\Omega_{f,I})$ *is ensured on a sufficiently small time interval. Further, in [79, p. 164, Cor.] the existence of weak solutions is proved under smallness assumptions about the data.*

- *For evolving domains, the functional analytical setting for proving the existence of weak solutions differs from the one used for time-independent domains. Stricter assumptions about the data and domain are required. Then, a velocity field with time derivative* $\partial_t \boldsymbol{v} \in \boldsymbol{L}^2(\Omega_{f,I})$ *instead of the distribution* $\partial_t \boldsymbol{v} \in L^2(I; H^{-1}(\Omega))$ *for time-independent domains* $\Omega$ *is obtained. For this reason, the stricter assumption* $\boldsymbol{f} \in \boldsymbol{L}^2(\Omega_{f,I})$ *about the right-hand side function* $\boldsymbol{f}$ *is made here. Further regularity conditions about* $\boldsymbol{f}$ *are supposed in [79] for the proof of existence and uniqueness.*

- *For* $\boldsymbol{v} \in V_I^t \cap L_I^\infty$ *and* $p \in L_{0,I}^2$ *the left-hand side terms in* (3.12) *are well-defined. The existence of the sufficiently smooth extension* $\hat{\boldsymbol{g}}$ *is tacitly assumed here. This requires smoothness assumptions about the evolving boundary of the fluid domain and assumptions about polygonal portions of the boundary. We skip the technical details of the assumptions since we focus on the discrete scheme here. On the discrete finite element level we avoid such extensions by employing Nitsche's method.*

### 3.2.3 Semidiscretization in space with weak enforcement of boundary conditions by Nitsche's method and ghost penalty stabilization

To incorporate Dirichlet boundary conditions on unfitted meshes we use Nitsche's method and follow [61], [85]. Here, the Dirichlet boundary conditions are enforced in a weak form by adding face integrals to the variational problem. Degrees of freedom assigned to Dirichlet portions of the boundary are now treated as unknowns of the variational problem. They are no longer enforced by the underlying function space and their implementation into the algebraic system, as it is usually done, with subsequent condensation of the algebraic equations.

Let $t \in [0,T]$. For the treatment of Dirichlet boundary conditions by Nitsche's method (cf. [61]) we introduce the bilinearform $B_{\Gamma_D^t} : \boldsymbol{H}^{1/2}(\Gamma_D^t) \times (\boldsymbol{V}_h \times Q_h) \to \mathbb{R}$ by

$$\begin{aligned} B_{\Gamma_D^t}(\boldsymbol{w},(\boldsymbol{\psi}_h,\xi_h)) := &- \langle \boldsymbol{w}, \nu\nabla\boldsymbol{\psi}_h\cdot\boldsymbol{n} + \xi_h\boldsymbol{n}\rangle_{\Gamma_D^t} \\ &+ \gamma_1\nu\langle h^{-1}\boldsymbol{w},\boldsymbol{\psi}_h\rangle_{\Gamma_D^t} + \gamma_2\langle h^{-1}\boldsymbol{w}\cdot\boldsymbol{n},\boldsymbol{\psi}_h\cdot\boldsymbol{n}\rangle_{\Gamma_D^t}\,, \end{aligned} \tag{3.13}$$

for $\boldsymbol{w} \in \boldsymbol{H}^{1/2}(\Gamma_D^t)$ and $(\boldsymbol{\psi}_h,\xi_h) \in \boldsymbol{V}_h \times Q_h$, where $\gamma_1 > 0$ and $\gamma_2 > 0$ are numerical (tuning) parameters for the penalization. In [28], [30], [86], it is demonstrated computationally that their choice in the interval $(10,100)$ leads to robust results. In our simulation presented in Sec. 3.5, we put $\gamma_1 = \gamma_2 = 35$. The discrete semilinear form $A_h : (\boldsymbol{V}_h \times Q_h) \times (\boldsymbol{V}_h \times Q_h) \to \mathbb{R}$ is then given by

$$A_h((\boldsymbol{v}_h,p_h),(\boldsymbol{\psi}_h,\xi_h)) := A_{\Omega_f^t}((\boldsymbol{v}_h,p_h),(\boldsymbol{\psi}_h,\xi_h)) - \langle \nu\nabla\boldsymbol{v}_h\cdot\boldsymbol{n} - p_h\boldsymbol{n},\boldsymbol{\psi}_h\rangle_{\Gamma_D^t} + B_{\Gamma_D^t}(\boldsymbol{v}_h,\boldsymbol{\phi}_h)\,, \tag{3.14}$$

for $(\boldsymbol{v}_h,p_h) \in \boldsymbol{V}_h \times Q_h$ and $(\boldsymbol{\psi}_h,\xi_h) \in \boldsymbol{V}_h \times Q_h$. The linear form $L_h : (\boldsymbol{V}_h \times Q_h) \to \mathbb{R}$ is defined by

$$L_h((\boldsymbol{\psi}_h,\xi_h);\boldsymbol{f},\boldsymbol{g}) := L_{\Omega_f^t}(\boldsymbol{\psi}_h;\boldsymbol{f}) + B_{\Gamma_D^t}(\boldsymbol{g},(\boldsymbol{\psi}_h,\xi_h))\,, \tag{3.15}$$

for $(\boldsymbol{\psi}_h,\xi_h) \in \boldsymbol{V}_h \times Q_h$. In (3.14) and (3.15), the forms $A_h$ and $L_h$ defined in (3.10) and (3.11), respectively, are extended naturally to test functions of $\boldsymbol{V}_h$ with nonhomogeneous Dirichlet conditions.

**Remark 3.2.**

- *We note that the forms* (3.13) *and* (3.14) *are introduced for (spatially) discrete functions that are defined on the time-independent background mesh* $\mathcal{T}_h$ *of the pipe* $\Omega$. *In* (3.13) *and* (3.14), *the integration is done over the evolving fluid domain* $\Omega_f^t$ *and its boundary part* $\Gamma_D^t$ *where Dirichlet boundary conditions are prescribed. The domain of the rigid body* $\Omega_r^t$ *is considered as a fictitious (ghost) flow domain. In the following, the discrete fluid velocity* $\boldsymbol{v}_h$ *and pressure* $p_h$ *are defined implicitly in the ghost domain* $\Omega_r^t$ *by the (semi-) discrete variational problem that is augmented by a ghost penalty stabilization exploiting an extension of the discrete functions to* $\Omega_r^t$.

- *We comment on the different boundary terms in the forms* (3.13) *and* (3.14). *The second term on the right-hand side of* (3.14) *reflects the natural boundary condition, making the weak imposition of the boundary conditions consistent. The first term on the right-hand side of* (3.13) *is introduced to preserve the symmetry properties of the continuous system. The last two terms are penalizations, that ensure the stability of the discrete system. In the inviscid limit* $\nu = 0$, *the last term amounts to a "no-penetration" condition. Thus, the form* (3.13) *provides a natural weighting between boundary terms corresponding to viscous effects* ($\boldsymbol{v} = \boldsymbol{g}$), *convective behavior* ($(\boldsymbol{v}\cdot\boldsymbol{n})^-\boldsymbol{v} = (\boldsymbol{g}\cdot\boldsymbol{n})^-\boldsymbol{g}$) *and inviscid behavior* ($\boldsymbol{v}\cdot\boldsymbol{n} = \boldsymbol{g}\cdot\boldsymbol{n}$).

To control irregular cuts of finite elements by the evolving domain and extend the fluid velocity and pressure functions from $\Omega_f^t$ to $\Omega_r^t$, and thereby to the entire, time-independent background domain $\Omega$, we introduce a discrete ghost penalty operator $S_h$. This stabilization then expands the semi-linear form (3.14) by additional terms. Our stabilization modifies an approach that was firstly introduced in [25] for convection-diffusion-reaction equations and, then, considered in [26] for Stokes problems to the Navier–Stokes system (3.9). This approach offers the appreciable advantage over further implementations of the ghost penalty method (cf. [25]) that no higher order spatial derivatives have to be computed. Thereby, it leads to reduced computational costs. The twofold motivation of the ghost penalty stabilization is sketched in Fig. 3.3 and Fig. 3.4, respectively. The first aim of the ghost stabilization $S_h$, defined in (3.17), is to extend the solution from the physical fluid domain $\Omega_f$ to the rigid domain $\Omega_r$. This is illustrated in Fig. 3.3. To assemble the algebraic system, i.e. the Jacobian matrix and right-hand-side function of the Newton linearization that is applied here, the discrete solution that is already computed for some time point $\tilde{t}_{n-1}$ in the corresponding fluid domain $\Omega_f^{\tilde{t}_{n-1}}$ needs to be evaluated in the fluid domain $\Omega_f^{\tilde{t}_n}$ at time $\tilde{t}_n$. For instance, in

the case of the lowest-order discontinuous Galerkin time discretization the time points $\tilde{t}_{n-1}$ and $\tilde{t}_n$ correspond to the midpoints of the subintervals $I_{n-1}$ and $I_n$ such that $\tilde{t}_{n-1} = (t_{n-2} + t_{n-1})/2$ and $\tilde{t}_n = (t_{n-1} + t_n)/2$. In the case of higher order discontinuous Galerkin time discretizations, more time nodes or degrees of freedom in time are involved, but the extension problem applies similarly. Due to the motion of the rigid body $\Omega_r^t$, the discrete velocities and pressure values at time $\tilde{t}_{n-1}$ are not necessarily defined in those parts of the rigid domain $\Omega_r^{\tilde{t}_{n-1}}$ that belong to the fluid domain $\Omega_f^{\tilde{t}_n}$ at time $\tilde{t}_n$ such that an appropriate extension of the solution at time $\tilde{t}_{n-1}$ to the rigid domain at time $\tilde{t}_{n-1}$ or to the entire time-independent domain $\Omega$, as done in this work, becomes indispensable. For this reason, we denote the domain of the rigid body as the fictitious or ghost domain of fluid flow, since discrete velocities and pressures are not defined here by means of a physical problem, but by some artificial extension only.

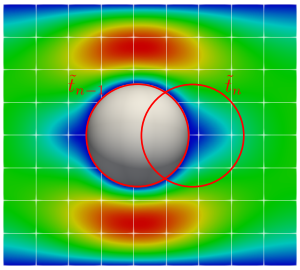
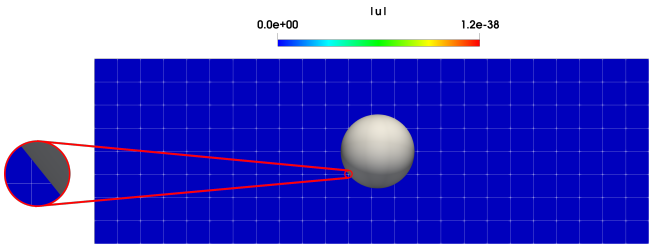


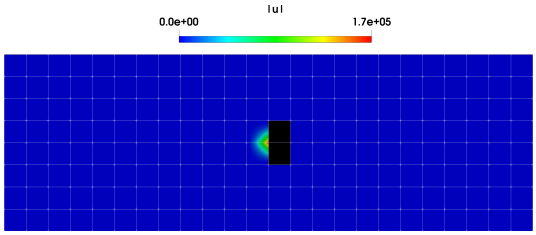
Figure 3.3: A moving rigid domain at two different time points $\tilde{t}_{n-1}$ and $\tilde{t}_n$ with computational background mesh.

The second aim of the stabilization is to reduce spurious and unphysical oscillations. They are due to unavoidable irregular or tiny cuts of finite elements by the moving rigid domain. This is illustrated in Fig. 3.4a. Here, we refer to the color range indicating the instability. The irregular cuts lead to steep gradients and oscillations in the computed solution profile. This is observation is related to the well-known challenges in the numerical approximation of convection-dominated transport. By steep gradients in the solution profile, the condition number of the corresponding linear algebraic system increases strongly which prevents their efficient solution by iterative methods. This problem of instability is sketched in Fig. 3.4b. For the sake of simplicity, flow around a fixed (non-moving) cylinder is simulated on a computational background mesh with arising cut cells. Here, no extension of the discrete solution to the rigid body is applied. In cut cells, the integration is done over the fluid part of the cell only. In Fig. 3.4b, an instability of the discrete solution in the cut cells is observed. By usage of ghost penalty stabilization that is proposed in the following, a smooth extension is obtained; cf. Fig. 3.4c. We explicitly note, that this extension admits no reasonable

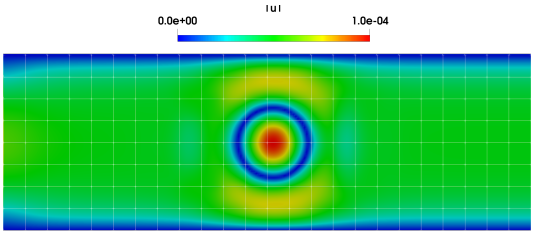
physical interpretation, since it is not based on any mathematical model but represents a numerically motivated approach.




(a) Irregular tiny cut cells of the background mesh due to the non body-fitted triangulation.




(b) Solution without ghost-penalty stabilization and extension. Cells $\in T_{h,r}^t$ are colored black and were inactive.




(c) Solution with ghost-penalty stabilization and fictitious domain extension to rigid body $\Omega_r^t$.

Figure 3.4: Effect of ghost-penalty stabilization and extension.

In the sequel, we extend the ghost penalty approach that is proposed in [25] for linear transport equations and extended in [26] to Stokes problems to the case of the Navier–Stokes system. However, our stabilization differs from the one that is suggested in [26]. In [26] the stabilization covers a boundary strip, i.e. the cut cells and cells of the fluid and the rigid domain adjacent to the boundary $\Gamma_r^t$. In the analysis, this is exploited to prove the error estimates. Here, we apply the combined stabilization and extension on the submeshes of the cut cells and the cells covering the rigid body (fictitious domain) $\Omega_r^t$ and, thus, to the entire domain $\Omega$ of our problem setting (cf. Fig. 3.1). This increases the efficiency of the parallel implementation (cf. Sec. 3.4). Our numerical experiments (cf. Sec. 3.5) indicate the admissibility of this modified ghost penalty stabilization and extension. For the problem setting of Fig. 3.1 we denote by the time-dependent submesh $\mathcal{T}_{h,s}^t$ the set of all cut cells and all cells that are entirely in the rigid domain,

$$\mathcal{T}_{h,s}^t := \mathcal{T}_{h,r}^t \cup \mathcal{T}_{h,c}^t = \left\{ K \in \mathcal{T}_h \mid K \subset \Omega_r^t \text{ or } K \cap \Omega_f^t \neq \emptyset \neq K \cap \Omega_r^t \right\} .$$

The set of all the faces that are common to two cells $K_1$ and $K_2$ of the submesh $\mathcal{T}_{h,s}^t$ is defined by

$$F_h^t := \left\{ \overline{K_1} \cap \overline{K_2} \mid K_1 \in \mathcal{T}_{h,s}^t \,,\ K_2 \in \mathcal{T}_{h,s}^t \,,\ K_1 \neq K_2 \,,\ \mathrm{meas}_1(\overline{K_1} \cap \overline{K_2}) > 0 \right\} . \tag{3.16}$$

For $F \in F_h^t$, we denote by $\omega_F = K_1 \cup K_2$ the patch of the two adjacent elements $K_1$ and $K_2$ with the common face $F$. Then we define the bilinear form $S_{F_h^t} : (\boldsymbol{V}_h \times Q_h) \times (\boldsymbol{V}_h \times Q_h) \to \mathbb{R}$ by

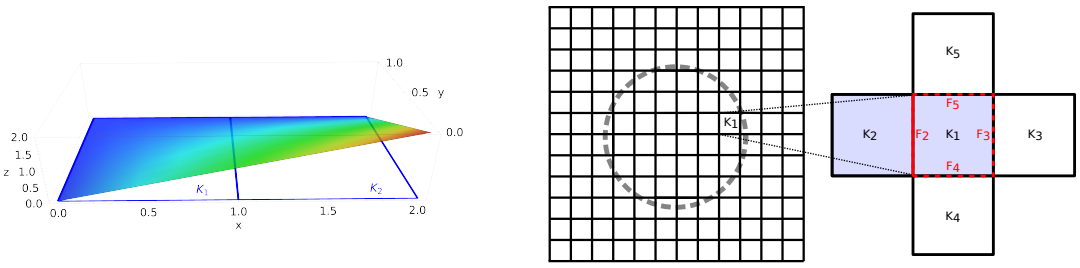
$$\begin{aligned} S_{F_h^t}((\boldsymbol{v}_h, p_h), (\psi_h, \xi_h)) = \sum_{F \in F_h^t} \gamma_v \left( \frac{1}{\nu} + \nu \right) \frac{1}{h^2} \left\langle (\mathcal{E}\boldsymbol{v}_{h|K_1} - \mathcal{E}\boldsymbol{v}_{h|K_2}), (\mathcal{E}\boldsymbol{\psi}_{h|K_1} - \mathcal{E}\boldsymbol{\psi}_{h|K_2}) \right\rangle_{\omega_F} \\ + \gamma_p \frac{1}{\nu} \left\langle (\mathcal{E}p_{h|K_1} - \mathcal{E}p_{h|K_2}), (\mathcal{E}\xi_{h|K_1} - \mathcal{E}\xi_{h|K_2}) \right\rangle_{\omega_F} , \end{aligned} \tag{3.17}$$

with numerical parameters $\gamma_v, \gamma_p > 0$. The stabilization (3.17) is based on [26]. However, in this work it is applied to the cut and rigid body cells given by $\mathcal{T}_{h,s}^t$. This approach differs from [26] which is motivated by our aim to use a time-independent background mesh for computational efficiency. The performed computations (cf. Sec. 3.5) illustrate the robustness of the extension and stabilization. In Eq. (3.17), $\mathcal{E}$ is the canonical extension of a polynomial function of degree $r$ in each variable, that is defined on one element of the face patch $\omega_F$, to the whole patch; cf. [25]. Thus, we have that

$$\mathcal{E} : \mathbb{Q}_r(K_i) \to \mathbb{Q}_r(\omega_F) \,, \quad \text{for } i \in \{1, 2\} \,, \tag{3.18}$$

with $(\mathcal{E}u)_{|K_i} = u$ for $u \in \mathbb{Q}_r(K_i)$. In the case of vector-valued functions, the extension (3.18) is applied component wise. To illustrate the extension (3.18), Fig. 3.5a shows the extended function $\mathcal{E}v_1$ of a scalar valued Lagrange $\mathcal{Q}_1$ finite element Lagrangian basis function $v_1$ that is defined on the quadrilateral $K_1$ and has the value 1 in the grid node $(1, 0)^\top$. In Fig. 3.5b, we illustrate the application of the stabilization operator (3.17) for the cut cell $K_1$. This cell has four faces, that all belong to the set $F_h^t$. For each of these faces we apply the stabilization (3.17) along with the extension (3.18). We start with the face $F_2$, marked by a full red line. The corresponding face patch, built from the cells $K_1$ and $K_2$, i.e. $\omega_F = K_1 \cup K_2$, is colored in purple in Fig. 3.5b. After evaluating the extension (3.17) on this first face patch we continue with the remaining three face patches, that are respectively built from the faces marked by the dashed red line. Proceeding in this way with all faces in $F_h^t$, we implicitly extend a discrete velocity $\boldsymbol{v}_h$ or pressure $p_h$ to $\Omega_r^t$ by adding the stabilization (3.17) to the space discretization. The implicit definition of the extension of the discrete functions comes through the

fact that the function values in the extended (ghost) domain $\Omega_r^t$ are obtained by the solution of the algebraic system and is not explicitly prescribed.




(a) Canonical extension $\mathcal{E}v_1$ of a Lagrange basis function from $K_1$ to $\omega_F = K_1 \cup K_2$.

(b) Patchwise extension to a fictitous domain for a cut cell $K_1$, with tiny, irregular cut.

Figure 3.5: Patchwise extension of ghost penalty stabilization.

Finally, we define the stabilized semilinear form $A_h^s : (\boldsymbol{V}_h \times Q_h) \times (\boldsymbol{V}_h \times Q_h) \to \mathbb{R}$ by

$$A_h^s((\boldsymbol{v}_h, p_h), (\boldsymbol{\psi}_h, \xi_h)) := A_h((\boldsymbol{v}_h, p_h), (\boldsymbol{\psi}_h, \xi_h)) + S_{F_h^t}((\boldsymbol{v}_h, p_h), (\psi_h, \xi_h)) \,, \tag{3.19}$$

for $(\boldsymbol{v}_h, p_h) \in \boldsymbol{V}_h \times Q_h$ and $(\boldsymbol{\psi}_h, \xi_h) \in \boldsymbol{V}_h \times Q_h$.

**Remark 3.3.** *We note that $A_h^s$ is defined on the time-independent bulk space $\boldsymbol{V}_h \times Q_h$. The volume integrals of $A_h$ (cf. (3.14)) in (3.19) and of $L_h$ in (3.15) are computed over the time-dependent fluid domain $\Omega_f^t$. Thus, an integration over the fluid portion of the cut cells has to be provided. This is done in Sec. 3.4. The penalty stabilization $S_h$ (cf. (3.17)) is computed over time-dependent patches that are subsets either of the fluid and the rigid domain or of the rigid one only.*

**Remark 3.4.** *Since we extend the solution of the physical fluid domain $\Omega_f$ to the whole computational domain $\Omega$, the choice of the time step size is in principle not restricted by a geometric CFL condition as it arises in [22]. This is confirmed by numerical experiments. Nevertheless, the choice of the time step size affects the performance and robustness of the ghost penalty stabilization. Its sensitivity depends on the geometrical and flow parameters like the diameter of the rigid domain and the fluid viscosity such that the time step size should be chosen reasonably.*

We are now in a position to define a semidiscrete approximation of the system (3.9).

**Problem 3.2.** *Let* $\boldsymbol{f} \in L^2(\Omega_{f,I})$ *and* $\boldsymbol{v}_{0,h} \in \boldsymbol{V}_h^{div}$ *be given. Find* $(\boldsymbol{v}_h, p_h) \in V_{I,h} \times L^2_{0,I,h}$, *such that* $\boldsymbol{v}_h(0) = \boldsymbol{v}_{0,h}$ *and for all* $(\boldsymbol{\psi}, \xi) \in L^2(I; \boldsymbol{V}_h) \times L^2_{0,I,h}$,

$$\int_0^T \langle \partial_t \boldsymbol{v}_h, \boldsymbol{\psi}_h \rangle_{\Omega_f^t} + A_h^s((\boldsymbol{v}_h, p_h), (\boldsymbol{\psi}_h, \xi_h)) \mathrm{d}t = \int_0^T L_h(\psi_h; \boldsymbol{f}, \boldsymbol{g}) \mathrm{d}t \,. \tag{3.20}$$

### 3.2.4 Discretization in time with discontinuous Galerkin methods

For the discretization in time we use the discontinuous Galerkin method with piecewise polynomials in time of order $k \in \mathbb{N}_0$. We note that continuous in time (or even continuously differentiable in time) variational discretizations of partial differential equations are known to be more efficient if the number degrees of freedom in time is measured versus the convergence rate of the time discretization if the underlying basis functions in time and quadrature formulas are chosen properly. For this observation we refer to, e.g. [54], [87], [88]. The superiority of the continuous in time families is then due the fact that some degrees of freedom in time are directly obtained by continuity constraints and do not have to be computed as parts of the algebraic system; cf. Chapter 5. However, if the Stokes or Navier–Stokes system is considered, the application of continuous in time variational discretization becomes more involved. This is due to the fact that no initial value for the pressure variable is given by the mathematical model. However, such initial value is required for the unique definition of the pressure trajectory as long as equal order in time discretizations of the velocity and pressure variable are desired. Applying the discontinuous in time Galerkin method, frees us from an initial value for the pressure. Moreover, stronger stability properties of the discretization are ensured since discontinuous Galerkin methods are known to be strongly $A$-stable.

In order to keep this work self-contained, we briefly present a formal derivation of the discontinuous Galerkin discretization for the abstract evolution problem

$$\partial_t \boldsymbol{v} + \boldsymbol{A}\boldsymbol{v} = \boldsymbol{f} \,, \qquad \boldsymbol{v}(0) = \boldsymbol{v}_0 \,, \tag{3.21}$$

as an equality in the dual space $V^*(t)$ of a time-dependent Hilbert space $V(t)$ of functions defined on an evolving domain $\Omega^t$ for $t \in [0, T]$; cf. [84]. In the formal derivation we tacitly assume that the solution of (3.21) satisfies all the additional conditions that are required such that the arising terms are well-defined. For the derivation of the

discontinuous Galerkin method we use the Reynold's transport theorem that reads as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega^t} \boldsymbol{v}\cdot\boldsymbol{\psi}\,\mathrm{d}\boldsymbol{x} = \int_{\Omega^t} \partial_t\boldsymbol{v}\cdot\boldsymbol{\psi}\,\mathrm{d}\boldsymbol{x} + \int_{\Omega^t} \boldsymbol{v}\cdot\partial_t\boldsymbol{\psi}\,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega^t} (\boldsymbol{v}\cdot\boldsymbol{\psi})(\boldsymbol{w}\cdot\boldsymbol{n})\,\mathrm{d}s\,. \tag{3.22}$$

Here, the vector field $\boldsymbol{w}(\boldsymbol{x},t)$ is the material velocity of the particles from the boundary $\partial\Omega^t$, and $\boldsymbol{n}$ is the outer unit normal vector. Substituting (3.21) into (3.22) yields that

$$\int_0^T \frac{\mathrm{d}}{\mathrm{d}t}\langle\boldsymbol{v},\boldsymbol{\psi}\rangle_{\Omega^t} - \langle\boldsymbol{v},\partial_t\boldsymbol{\psi}\rangle_{\Omega^t} - \big\langle\boldsymbol{v}\cdot\boldsymbol{\psi},\boldsymbol{w}\cdot\boldsymbol{n}\big\rangle_{\partial\Omega^t} + \langle\boldsymbol{A}\boldsymbol{v},\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t = \int_0^T \langle\boldsymbol{f},\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t\,. \tag{3.23}$$

With the fundamental theorem of calculus and for test functions $\psi$ with $\boldsymbol{\psi}(T)=\boldsymbol{0}$ we get that

$$-\langle\boldsymbol{v}(0),\boldsymbol{\psi}(0)\rangle_{\Omega^0} - \int_0^T \langle\boldsymbol{v},\partial_t\boldsymbol{\psi}\rangle_{\Omega^t} + \big\langle\boldsymbol{v}\cdot\boldsymbol{\psi},\boldsymbol{w}\cdot\boldsymbol{n}\big\rangle_{\partial\Omega^t} - \langle\boldsymbol{A}\boldsymbol{v},\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t = \int_0^T \langle\boldsymbol{f},\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t\,. \tag{3.24}$$

Rewriting the integrals in (3.24) as a sum over the subintervals of the time mesh $\mathcal{M}_\tau$, we get that

$$-\langle\boldsymbol{v}(0),\boldsymbol{\psi}(0)\rangle_{\Omega^0} - \sum_{n=1}^N \int_{t_{n-1}}^{t_n} \langle\boldsymbol{v},\partial_t\boldsymbol{\psi}\rangle_{\Omega^t} + \big\langle\boldsymbol{v}\cdot\boldsymbol{\psi},\boldsymbol{w}\cdot\boldsymbol{n}\big\rangle_{\partial\Omega^t} - \langle\boldsymbol{A}\boldsymbol{u},\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t = \sum_{n=1}^N \int_{t_{n-1}}^{t_n} \langle\boldsymbol{f},\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t\,. \tag{3.25}$$

From (3.22) we derive that

$$-\int_{t_{n-1}}^{t_n} \langle\boldsymbol{v},\partial_t\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t = -\int_{t_{n-1}}^{t_n} \frac{\mathrm{d}}{\mathrm{d}t}\langle\boldsymbol{v},\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t + \int_{t_{n-1}}^{t_n} \langle\partial_t\boldsymbol{v},\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t + \int_{t_{n-1}}^{t_n} \big\langle\boldsymbol{v}\cdot\boldsymbol{\psi},\boldsymbol{w}\cdot\boldsymbol{n}\big\rangle_{\partial\Omega^t}\mathrm{d}t\,.$$

Applying the fundamental theorem of calculus to the first term on the right-hand side of this identity and substituting the resulting equation into (3.25) shows that

$$\begin{aligned}\sum_{n=1}^N \int_{t_{n-1}}^{t_n} \langle\partial_t\boldsymbol{v},\boldsymbol{\psi}\rangle_{\Omega^t} + \langle A\boldsymbol{v},\boldsymbol{\psi}\rangle_{\Omega^t}\mathrm{d}t + \sum_{n=1}^{N-1} \Big[\big\langle\boldsymbol{v},\boldsymbol{\psi}\big\rangle_{\Omega^t}\Big]_n + \big\langle\boldsymbol{v}(0^+),\boldsymbol{\psi}(0)\big\rangle_{\Omega^0} \\ = \sum_{n=1}^N \int_{t_{n-1}}^{t_n} \langle\boldsymbol{\phi};\boldsymbol{f}\rangle_{\Omega^t}\mathrm{d}t + \langle\boldsymbol{v}_0,\boldsymbol{\psi}(0)\rangle_{\Omega^0}\,,\end{aligned} \tag{3.26}$$

with the jump operator $[\cdot]_n$ at $t_n$, defined as $[g]_n := g(t_n^+) - g(t_n^-)$ with the one-sided limits $g(t_n^\pm) = \lim_{t\to t_n^\pm} g(t)$. For a smoothly evolving domain $\Omega^t$, that is assumed here, the one-sided limits of $\Omega^t$ at the time nodes $t_n$ coincide. We note that (3.26) continues to be well-defined for functions that are differentiable piecewise in time (with respect to

the time mesh $\mathcal{M}_\tau$) only. Thus, (3.26) can be solved within the space (2) of piecewise polynomials in time.

Applying this concept of discontinuous Galerkin time discretization to the semidiscrete Problem 3.2 yields the following fully discrete problem.

**Problem 3.3.** *Let $\boldsymbol{f} \in \boldsymbol{L}^2(\Omega_{f,I})$ and $\boldsymbol{v}_{0,h} \in \boldsymbol{V}_h^{div}$ be given. For $n = 1, \dots, N$, and given $\boldsymbol{v}_{\tau,h|I_{n-1}} \in \mathbb{P}_k(I_{n-1}; \boldsymbol{V}_h)$ for $n > 1$ and $\boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1}^-) := \boldsymbol{v}_{0,h}$ for $n = 1$, find $(\boldsymbol{v}_{\tau,h}, p_{\tau,h}) \in \mathbb{P}_k(I_n; \boldsymbol{V}_h) \times \mathbb{P}_k(I_n; Q_h)$, such that for all $(\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}) \in \mathbb{P}_k(I_n; \boldsymbol{V}_h) \times \mathbb{P}_k(I_n; Q_h)$,*

$$\int_{t_{n-1}}^{t_n} \langle \partial_t \boldsymbol{v}_{\tau,h}, \boldsymbol{\psi}_{\tau,h} \rangle_{\Omega_f^t} + A_h^s((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))\mathrm{d}t + \langle \boldsymbol{v}_{\tau,h}(t_{n-1}^+), \boldsymbol{\psi}_{\tau,h}(t_{n-1}^+) \rangle_{\Omega^{t_{n-1}}}$$
$$= \int_{t_{n-1}}^{t_n} L_h(\psi_{\tau,h}; \boldsymbol{f}, \boldsymbol{g})\mathrm{d}t + \langle \boldsymbol{v}_{\tau,h}(t_{n-1}^-), \boldsymbol{\psi}_{\tau,h}(t_{n-1}^+) \rangle_{\Omega^{t_{n-1}}} . \quad (3.27)$$

**Remark 3.5.**

- *In Problem 3.3 a smoothly evolving domain is assumed such that $\Omega^{t_{n-1}^+} = \Omega^{t_{n-1}^-}$ is satisfied. All volume integrals that arise in the forms of Eq. (3.27) are computed over the fluid domain $\Omega_f^t$.*
- *To solve the algebraic counterpart of Eq. (3.27) we use an inexact Newton method, which is presented in detail in Sec. 3.3.*
- *To solve the linear system of the Newton iteration in this chapter, we use the parallel, sparse direct solver SuperLU_DIST [89]. The development of an efficient geometric multigrid preconditioner for the CutFEM approach is addressed in Sec. 4.6.*

### 3.2.5 Algebraic in time formulation

In this section we derive a semi-algebraic formulation of Problem 3.3. The algebraic counterpart of Eq. (3.27) is presented with respect to the time variable only. The restriction to the time-discretization is done due to the challenges related to the evolving domain. The presentation of the fully algebraic counterpart of Eq. (3.27) is skipped here for brevity. The transformation of the space discretization into an algebraic form follows the usual steps of finite element methods.

In Problem 3.3, numerical integration in time by the $m$-point Gauss quadrature is still applied. For $g \in \{f : (0,T) \to L^2(\Omega) \mid f \in C(I_n; L^2(\Omega)) \; \forall I_n \in \mathcal{M}_\tau\}$ this formula is given by

$$Q_n(g) = \sum_{\mu=1}^{m} w_\mu g(t_{n,\mu}) \approx \int_{I_n} g(t)\mathrm{d}t \,, \tag{3.28}$$

where $t_{n,\mu} \in I_n$, for $\mu = 1, \ldots, m$, are the integration points and $w_\mu > 0$ the weights. The number $m \in \mathbb{N}$ of quadrature points, and thereby the order of exactness $2m-1$ the quadrature formula (3.28), is chosen such that $m \geq (3k+1)/2$ is satisfied. Then, the integration in time of the discrete semilinearform $A_h((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))$ on the left-hand side of (3.27) is done exactly. In Fig. 3.6 we illustrate the distribution of the space-time integration points for a one-dimensional setting in space and a single cell. Two quadrature nodes are used for the time domain. This corresponds to the application of the dG(1) scheme in time. Six quadrature nodes are used for spatial integration, which arises in the spatial approximation by piecewise quadratic functions. Further, we define a temporal basis $\{\chi_l\}_{l=0}^k \subset \mathbb{P}_k(I_n; \mathbb{R})$ by the conditions

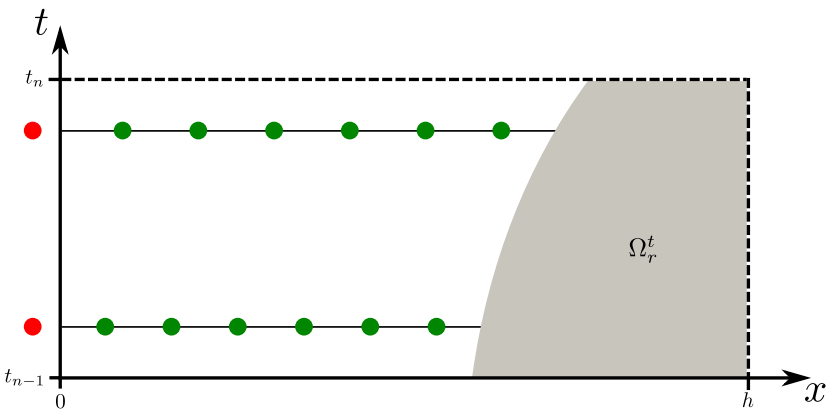



Figure 3.6: Space-time quadrature nodes for a one-dimensional problem and a single cut cell with a rigid domain $\Omega_r^t$.

$$\chi_l\left(t_{n,\mu}^{\mathrm{GR}}\right) = \delta_{l,\mu} \,, \quad \text{for } \; l, \mu = 0, \ldots, k \,,$$

for the Gauss–Radau quadrature nodes $\{t_{n,\mu}^{\mathrm{GR}}\}_{\mu=0}^k$ of the subinterval $I_n$. Expanding the discrete solution $(\boldsymbol{v}_{\tau|I_n}, p_{\tau|I_n}) \in \mathbb{P}_k(I_n; \boldsymbol{V_h}) \times \mathbb{P}_k(I_n; Q_h)$ on $I_n$ in terms of the basis functions yields

$$\begin{aligned} \boldsymbol{v}_{\tau,h|I_n}(\boldsymbol{x}, t) &= \sum_{l=0}^{k} \boldsymbol{v}_{n,l}(\boldsymbol{x})\chi_l(t) \,, \\ p_{\tau,h|I_n}(\boldsymbol{x}, t) &= \sum_{l=0}^{k} p_{n,l}(\boldsymbol{x})\chi_l(t) \,, \end{aligned} \tag{3.29}$$

for $t \in I_n$ and with coefficient functions $\boldsymbol{v}_{n,l} \in \boldsymbol{V}_h$ and $p_{n,l} \in Q_h$. Appreciable advantage of the usage of the Gauss–Radau quadrature points for the construction of the temporal basis is that the quantity $\boldsymbol{v}_{\tau,h|I_{n-1}}(\boldsymbol{x}, t_{n-1}^-)$ in (3.27), that is due to the discontinuous Galerkin scheme, is a degree of freedom of the time discretization and, thus, is available without further computational costs.
Then, Problem 3.3 can be recovered in the following form.

**Problem 3.4.** *Let $\boldsymbol{f} \in \boldsymbol{L}^2(\Omega_{f,I})$ and $\boldsymbol{v}_{0,h} \in \boldsymbol{V}_h^{div}$ be given. For $n = 1, \dots, N$, and given $\boldsymbol{v}_{\tau,h|I_{n-1}} \in \mathbb{P}_k(I_{n-1}; \boldsymbol{V}_h)$ for $n > 1$ and $\boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1}^-) := \boldsymbol{v}_{0,h}$ for $n = 1$, find $(\boldsymbol{v}_{\tau,h}, p_{\tau,h}) \in \mathbb{P}_k(I_n; \boldsymbol{V}_h) \times \mathbb{P}_k(I_n; Q_h)$, such that for all $j \in \{0 \dots k\}$ and $(\boldsymbol{\psi}_h, \xi_h) \in \boldsymbol{V}_h \times Q_h$,*

$$\begin{aligned}
&\sum_{\mu=1}^{m} w_\mu \Big[ \langle \partial_t \boldsymbol{v}_{\tau,h|I_n}(t_{n,\mu}), \boldsymbol{\psi}_h \, \chi_j(t_{n,\mu}) \rangle_{\Omega_f^{t_{n,\mu}}} \\
&\qquad + A_h^s \left( \left( \boldsymbol{v}_{\tau,h|I_n}(t_{n,\mu}), p_{\tau,h|I_n}(t_{n,\mu}) \right), (\boldsymbol{\psi}_h \, \chi_j(t_{n,\mu}), \xi_h \, \chi_j(t_{n,\mu})) \right) \Big] \\
&\qquad + \left\langle \boldsymbol{v}_{\tau,h|I_n}(t_{n-1}^+), \boldsymbol{\psi}_h \, \chi_j(t_{n-1}^+) \right\rangle_{\Omega^{t_{n-1}}} \\
&= \sum_{\mu=1}^{m} w_\mu \left[ (L_h(\boldsymbol{\psi}_h \, \chi_j(t_{n,\mu}); \boldsymbol{f}, \boldsymbol{g})) \right] + \left\langle \boldsymbol{v}_{\tau,h|I_{n-1}}(\boldsymbol{x}, t_{n-1}^-), \boldsymbol{\psi}_h(\boldsymbol{x}) \chi_j(t_{n-1}^+) \right\rangle_{\Omega^{t_{n-1}}} . \quad (3.30)
\end{aligned}$$

To illustrate the structure of Eq. (3.30), we review the terms in Eq. (3.30) more in detail. Exemplarily, this is done for some of them. Using the expansions given in (3.29), we obtain that

$$\langle \partial_t \boldsymbol{v}_{\tau,h|I_n}(t_{n,\mu}), \boldsymbol{\psi}_h \, \chi_j(t_{n,\mu}) \rangle_{\Omega_f^{t_{n,\mu}}} = \sum_{l=0}^{k} \langle \boldsymbol{v}_{n,l} \, \partial_t \chi_l(t_{n,\mu}), \boldsymbol{\psi}_h \, \chi_j(t_{n,\mu}) \rangle_{\Omega_f^{t_{n,\mu}}} , \quad (3.31)$$

$$\langle \nabla \boldsymbol{v}_{\tau,h|I_n}(t_{n,\mu}), \nabla \boldsymbol{\psi}_h \, \chi_j(t_{n,\mu}) \rangle_{\Omega_f^{t_{n,\mu}}} = \sum_{l=0}^{k} \langle \nabla \boldsymbol{v}_{n,l} \, \chi_l(t_{n,\mu}), \nabla \boldsymbol{\psi}_h \, \chi_j(t_{n,\mu}) \rangle_{\Omega_f^{t_{n,\mu}}} , \quad (3.32)$$

$$\langle p_{\tau,h|I_n}(t_{n,\mu}), \nabla \cdot \boldsymbol{\psi}_h \, \chi_j(t_{n,\mu}) \rangle_{\Omega_f^{t_{n,\mu}}} = \sum_{l=0}^{k} \langle p_{n,l} \, \chi_l(t), \nabla \cdot \boldsymbol{\psi}_h \, \chi_j(t_{n,\mu}) \rangle_{\Omega_f^{t_{n,\mu}}} . \quad (3.33)$$

Thus, Problem 3.4 aims at computing the coefficient functions $\{\boldsymbol{v}_{n,l}, p_{n,l}\}_{l=0}^k$, with $\{\boldsymbol{v}_{n,l}, p_{n,l}\} \in \boldsymbol{V}_h \times Q_h$ for $l = 0, \dots, k$. Clearly, the pair $\{\boldsymbol{v}_{n,l}, p_{n,l}\} \in \boldsymbol{V}_h \times Q_h$ yields the fully discrete approximation in the Gauss–Radau quadrature node $t_{n,l} \in I_n$. The coefficient functions are elements of the bulk finite element spaces $\boldsymbol{V}_h$ and $Q_h$, respectively, and are thus defined on the computational background mesh $\mathcal{T}_h$ of $\Omega$. In

the rigid body domain they are defined implicitly by means of the stabilization and extension operator $S_{F_h^t}$ introduced in Eq. (3.17).

## 3.3 Newton Linearization

In this section we apply Newton's method to our nonlinear problem in Sec. 3.3.1 and show two ways of globalizing the resulting scheme in Sec. 3.3.2.

### 3.3.1 Space–time discrete system

Problem 3.3 is a nonlinear equation for $\boldsymbol{u}_{\tau,h} = \{\boldsymbol{v}_{\tau,h}, p_{\tau,h}\}$. To solve it there essentially exist two different ways: a fixed point iteration or applying a Newton scheme [38], [77]. The "Picard iteration" is probably the most prominent representative of a fixed point iteration scheme. Its advantage is, that it is a robust and well-studied scheme, but this comes at the cost of just having a linear convergence order. The Newton scheme in contrast is locally quadratic convergent but needs a good initial guess to converge.

In this work a Newton scheme is applied to solve Problem 3.3. To globalize it we have on the one hand implemented a classical backtracking approach, in which we apply a linesearch to damp the length of a Newton step. On the other hand we have implemented a "dogleg method" (see e.g. [90]), which belongs to the class of trust-region methods and whose advantage is, that also the search direction, not just it's length, might vary. Both schemes just need the Jacobian matrix of the system in Problem 3.3, but in the dogleg method we have to compute multiple Jacobian–vector products. Since we don't use a "matrix free" approach in this paper and have the Jacobian stored as sparse matrix, this matrix–vector product can be evaluated at low computational costs.

To derive the (non-damped) Newton iteration we first subtract the right-hand-side of (3.27) and let this new left hand side define our nonlinear operator $\boldsymbol{q}(\boldsymbol{u_{\tau,h}}, \boldsymbol{\phi}_{\tau,h})$:

$$\begin{aligned}\boldsymbol{q} \coloneqq \int_{t_{n-1}}^{t_n} \langle \partial_t \boldsymbol{v}_{\tau,h}, \boldsymbol{\psi}_{\tau,h} \rangle_{\Omega_f^t} + A_h^s((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))\mathrm{d}t - L_h(\psi_{\tau,h}; \boldsymbol{f}, \boldsymbol{g})\,\mathrm{d}t \\ + \langle \boldsymbol{v}_{\tau,h}(t_{n-1}^+), \boldsymbol{\psi}_{\tau,h}(t_{n-1}^+) \rangle_{\Omega^{t_{n-1}}} - \langle \boldsymbol{v}_{\tau,h}(t_{n-1}^-), \boldsymbol{\psi}_{\tau,h}(t_{n-1}^+) \rangle_{\Omega^{t_{n-1}}}\,, \quad (3.34)\end{aligned}$$

with $\boldsymbol{\phi}_{\tau,h} = \{\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}\}$. The equation, that has to be solved now reads as:

$$\boldsymbol{q}(\boldsymbol{u}_{\boldsymbol{\tau},\boldsymbol{h}}, \boldsymbol{\phi}_{\tau,h}) = \mathbf{0}\,. \tag{3.35}$$

The Newton scheme for step $m+1$ then reads as:

$$\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}^m_{\tau,h}} \boldsymbol{d}^{m+1}_{\tau,h} = -\boldsymbol{q}(\boldsymbol{u}^m_{\tau,h}, \boldsymbol{\phi}_{\tau,h})\,. \tag{3.36}$$

for the correction $\boldsymbol{d}^{m+1}_{\tau,h} = (\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}}, d^{m+1}_{p_{\tau,h}})^\top \coloneqq \boldsymbol{u}^{m+1}_{\tau,h} - \boldsymbol{u}^m_{\tau,h}$. The left hand side can be interpreted as directional derivative of $\boldsymbol{q}$ along $\boldsymbol{d}^{m+1}_{\tau,h}$ at $\boldsymbol{u}^m_{\tau,h}$ given by

$$\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}^m_{\tau,h}} \boldsymbol{d}^{m+1}_{\tau,h} = \lim_{\epsilon\to 0} \frac{1}{\epsilon}\Big(\boldsymbol{q}(\boldsymbol{u}^m_{\tau,h} + \epsilon \boldsymbol{d}^m_{\tau,h}) - \boldsymbol{q}(\boldsymbol{u}^m_{\tau,h})\Big)\,. \tag{3.37}$$

We introduce a linearized operator $A_{lin}$ at the m-th Newton iteration as:

$$\begin{aligned} A_{lin}(\boldsymbol{d}^{m+1}_{\tau,h}, \boldsymbol{\phi}_{\tau,h}) = A_{lin}((\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}}, d^{m+1}_{p_{\tau,h}}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h})) \coloneqq \\ \langle(\boldsymbol{v}^m_{\tau,h}\cdot\boldsymbol{\nabla})\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}}, \boldsymbol{\psi}_{\tau,h}\rangle_{\Omega^t_f} + \langle(\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}}\cdot\boldsymbol{\nabla})\boldsymbol{v}^m_{\tau,h}, \boldsymbol{\psi}_{\tau,h}\rangle_{\Omega^t_f} + \nu\langle\nabla\boldsymbol{v}^{m+1}_{\tau,h}, \nabla\boldsymbol{\psi}_{\tau,h}\rangle_{\Omega^t_f} \\ - \langle d^{m+1}_{p_{\tau,h}}, \nabla\cdot\boldsymbol{\psi}_{\tau,h}\rangle_{\Omega^t_f} + \langle\boldsymbol{\nabla}\cdot\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}}, \xi\rangle_{\Omega^t_f}\,. \end{aligned} \tag{3.38}$$

We further introduce an operator $A^s_{lin}((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))$, which is formed by replacing $A_h((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))$ in Eq. (3.19) with $A_{lin}(\boldsymbol{d}^{m+1}_{\tau,h}, \boldsymbol{\phi}_{\tau,h})$ of Eq. (3.38).

$$A^s_{lin}(\boldsymbol{d}^{m+1}_{\tau,h}, \boldsymbol{\phi}_{\tau,h}) \coloneqq A_{lin}(\boldsymbol{d}^{m+1}_{\tau,h}, \boldsymbol{\phi}_{\tau,h}) + S_{F^t_h}(\boldsymbol{d}^{m+1}_{\tau,h}, \boldsymbol{\phi}_{\tau,h})\,.$$

With this we can write the directional gradient as:

$$\begin{aligned} \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}^m_{\tau,h}} \boldsymbol{d}^{m+1}_{\tau,h} = \int_{t_{n-1}}^{t_n} \langle\partial_t \boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}}, \boldsymbol{\psi}_{\tau,h}\rangle_{\Omega^t_f} + A^s_{lin}((\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}}, d^{m+1}_{p_{\tau,h}}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))\mathrm{d}t \\ + \left\langle \boldsymbol{d}^{m+1,n-1,+}_{\boldsymbol{v}_{\tau,h}}, \boldsymbol{\psi}^{n-1,+}_{\tau,h}\right\rangle_{\Omega_f(t_{n-1})}, \end{aligned} \tag{3.39}$$

which forms the left hand side of Eq. (3.36).

For the algebraic representation of Eq. (3.39), the unknown discrete functions $(\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}}, d^{m+1}_{p_{\tau,h}}) \in \mathbb{P}_k(I_n; \boldsymbol{V}_h) \times \mathbb{P}_k(I_n; Q_h)$ in a temporal basis $\{\chi_l\}_{l=0}^k$ of $\mathbb{P}_k(I_n; \mathbb{R})$ by means of

$$d^{m+1}_{v_{\tau,h,i|I_n}}(\boldsymbol{x}, t) = \sum_{l=0}^{k} d^{n,l}_{v_i}(\boldsymbol{x}) \chi_{n,l}(t)\,, \quad \text{for } i \in \{1, \dots, \dim\}\,, \tag{3.40}$$

$$d^{m+1}_{p_{\tau,h|I_n}}(\boldsymbol{x}, t) = \sum_{l=0}^{k} d^{n,l}_{p}(\boldsymbol{x}) \chi_{n,l}(t)\,, \tag{3.41}$$

with coefficient functions $\boldsymbol{d}^{n,l}_{\boldsymbol{v}} = (d^{n,l}_{v_1}, \dots, d^{n,l}_{v_d})^\top \in \boldsymbol{V}_h$ and $d^{n,l}_p \in Q_h$, where $\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}} = (\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h,1}}, \dots, \boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h,d}})^\top$ for $t \in I_n$. For the basis $\{\chi_l\}_{l=0}^k$ we choose the Lagrange interpolants with respect to the $k+1$ Gauss–Radau quadrature nodes of $I_n$. Appreciable of the Gauss–Radau quadrature formula is that the end point of the subinterval $I_n$ is a quadrature node, which simplifies the evaluation of the second term on the right-hand side of Eq. (4.9). Letting

$$H^r_h = \operatorname{span}\{\psi_1, \dots, \psi_R\} \quad \text{and} \quad Q_h = \operatorname{span}\{\xi_1, \dots, \xi_S\}\,, \tag{3.42}$$

the coefficient functions $\boldsymbol{d}^{n,l}_{\boldsymbol{v}} \in \boldsymbol{V}_h$ and $d^{n,l}_p \in \boldsymbol{Q}_h$ of Eq. (3.40) admit the representation

$$d^{n,l}_{v_i}(\boldsymbol{x}) = \sum_{r=1}^{R} d^{n,l}_{v_{i,r}}\, \psi_r(\boldsymbol{x}) \quad \text{and} \quad d^{n,l}_{p}(\boldsymbol{x}) = \sum_{s=1}^{S} d^{n,l}_{p_s}\, \xi_s(\boldsymbol{x})\,,$$

with the vectors of unknown coefficients

$$\boldsymbol{d}^{n,l}_{\boldsymbol{v}_i} = (d^{n,l}_{v_{i,1}}, \dots, d^{n,l}_{v_{i,R}})^\top \in \mathbb{R}^R \quad \text{and} \quad \boldsymbol{d}^{n,l}_{\boldsymbol{p}} \boldsymbol{p}^{n,l} = (d^{n,l}_{p,1}, \dots, d^{n,l}_{p,S})^\top \in \mathbb{R}^S\,,$$

for all degrees of freedom in time in $I_n$ with $l = 0, \dots, k$. Clearly, the vectors $\boldsymbol{d}^{n,l}_{\boldsymbol{v}_i}$ denote the coefficients of the velocity component functions $d^{n,l}_i$, with $i = 1, \dots, d$, with respect to the spacial basis $\{\psi_r\}_{r=1}^R$. For brevity the scalar valued basis of the velocity field is also combined to a canonical, vectorial basis and expressed as:

$$\boldsymbol{d}^{n,l}_{\boldsymbol{v}}(\boldsymbol{x}) = \sum_{r=1}^{J} \boldsymbol{d}^{n,l}_{\boldsymbol{v}_r}\, \boldsymbol{\psi}_r(\boldsymbol{x})\,,$$

where we used the abbreviation $J = R \cdot \dim$.

Next the vector $\boldsymbol{X}_n \in \mathbb{R}^{(k+1)\times(J+S)}$ of unknown coefficients for the solution of Problem 3.3 in the subinterval $I_n$ is defined by

$$\boldsymbol{X}_n = (\boldsymbol{v}_1^{n,0}, \ldots, \boldsymbol{v}_d^{n,0}, \boldsymbol{p}^{n,0}, \ldots, \boldsymbol{v}_1^{n,k}, \ldots, \boldsymbol{v}_d^{n,k}, \boldsymbol{p}^{n,k})^\top \in \mathbb{R}^{(k+1)\times(\dim\cdot R+S)}\,. \tag{3.43}$$

With this Eq. (3.35) can be recovered in an algebraic form as

$$\boldsymbol{Q}_n(\boldsymbol{X}_n) = \boldsymbol{0}\,, \tag{3.44}$$

for $\boldsymbol{Q}_n : \mathbb{R}^{(k+1)\times(J+S)} \to \mathbb{R}^{(k+1)\times(J+S)}$. With the vector $\boldsymbol{D}_n \in \mathbb{R}^{(k+1)\times(J+S)}$ of corrections in the Newton iterate m:

$$\boldsymbol{D}_n^{m+1} = (\boldsymbol{d}_{\boldsymbol{v}_1}^{n,0}, \ldots, \boldsymbol{d}_{\boldsymbol{v}_d}^{n,0}, \boldsymbol{d}_{\boldsymbol{p}}^{n,0}, \ldots, \boldsymbol{d}_{\boldsymbol{v}_1}^{n,k}, \ldots, \boldsymbol{d}_{\boldsymbol{v}_d}^{n,k}, \boldsymbol{d}_{\boldsymbol{p}}^{n,k})^\top \in \mathbb{R}^{(k+1)\times(J+S)}\,,$$

Eq. (3.36) can be recovered as the linear system

$$\boldsymbol{J}_n^m \boldsymbol{D}_n^{m+1} = \boldsymbol{Q}_n^m\,, \tag{3.45}$$

with the Jacobian matrix and right-hand side vector

$$\boldsymbol{J}_n^m = \frac{\partial \boldsymbol{Q}_n}{\partial \boldsymbol{X}_n}(\boldsymbol{X}_n^m) \quad \text{and} \quad \boldsymbol{Q}_n^m = \boldsymbol{Q}_n(\boldsymbol{X}_n^m)\,. \tag{3.46}$$

Eq. (3.45) has to be solved in each Newton iteration for the new iterate

$$\boldsymbol{X}_n^{m+1} = \boldsymbol{X}_n^m + \boldsymbol{D}_n^{m+1}\,.$$

The Jacobian matrix, $\boldsymbol{J}_n^m$ consists of $(k+1)^2$ blocks of type of the discretized Stokes operator, that have the following saddle point structure:

$$\begin{pmatrix} \boldsymbol{F} & \boldsymbol{B}^\top \\ -\boldsymbol{B} & \boldsymbol{C} \end{pmatrix}. \tag{3.47}$$

For a discontinuous Galerkin of order $k$ time discretization scheme the resulting Jacobian matrix has the following structure:

$$\boldsymbol{J}_n^m = \begin{pmatrix} \boldsymbol{F}_{0,0} & \boldsymbol{B}_{0,0}^\top & \dots & \boldsymbol{F}_{0,k} & \boldsymbol{B}_{0,k}^\top \\ -\boldsymbol{B}_{0,0} & \boldsymbol{C}_{0,0} & \dots & -\boldsymbol{B}_{0,k} & \boldsymbol{C}_{0,k} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{F}_{k,0} & \boldsymbol{B}_{k,0}^\top & \dots & \boldsymbol{F}_{k,k} & \boldsymbol{B}_{k,k}^\top \\ -\boldsymbol{B}_{k,0} & \boldsymbol{C}_{k,0} & \dots & -\boldsymbol{B}_{k,k} & \boldsymbol{C}_{k,k} \end{pmatrix}. \tag{3.48}$$

Here the partitioning of the vector of unknowns $\boldsymbol{D}_n^{m+1}$ of the corresponding system Eq. (3.45) is given by

$$\boldsymbol{D}_n^{m+1} = (\boldsymbol{d_v}^{n,0}, \boldsymbol{d_p}^{n,0}, \dots, \boldsymbol{d_v}^{n,k}, \boldsymbol{d_p}^{n,k})^\top \in \mathbb{R}^{(k+1)(d\cdot R+S)},$$

where $\boldsymbol{d_v}^{n,l}$ and $\boldsymbol{d_p}^{n,l}$, with $l \in \{0, k\}$, denote the components of $\boldsymbol{D}_n^{m+1}$ related to velocity and pressure unknowns, respectively. For $a, b \in [0, k]$ and $i, j \in [1, J]$ the explicit representation of the $\boldsymbol{F}$ blocks can be expressed as:

$$\begin{aligned} \boldsymbol{F}_{a,b} = \int_{t_{n-1}}^{t_n} & \langle \partial_t \chi_{n,a} \boldsymbol{\psi}_i, \chi_{n,b} \boldsymbol{\psi}_j \rangle_{\Omega_f^t} + \langle (\boldsymbol{v}_{\tau,h}^m \cdot \boldsymbol{\nabla}) \chi_{n,a} \boldsymbol{\psi}_i, \chi_{n,b} \boldsymbol{\psi}_j \rangle_{\Omega_f^t} \\ & + \langle (\chi_{n,a} \boldsymbol{\psi}_i \cdot \boldsymbol{\nabla}) \boldsymbol{v}_{\tau,h}^m, \chi_{n,b} \boldsymbol{\psi}_j \rangle_{\Omega_f^t} + \nu \langle \nabla \chi_{n,a} \boldsymbol{\psi}_i, \nabla \chi_{n,b} \boldsymbol{\psi}_j \rangle_{\Omega_f^t} \\ & + \langle \nu \chi_{n,a} \nabla \boldsymbol{\psi}_i \cdot \boldsymbol{n}, \chi_{n,b} \boldsymbol{\psi}_j \rangle_{\Gamma_D^t} - \langle \chi_{n,a} \boldsymbol{\psi}_i, \nu \chi_{n,b} \nabla \boldsymbol{\psi}_j \cdot \boldsymbol{n} \rangle_{\Gamma_D^t} \\ & + \gamma_1 h^{-1} \nu \langle \chi_{n,a} \boldsymbol{\psi}_i, \chi_{n,b} \boldsymbol{\psi}_j \rangle_{\Gamma_D^t} + \gamma_2 h^{-1} \langle \chi_{n,a} \boldsymbol{\psi}_i \cdot \boldsymbol{n}, \boldsymbol{\chi}_{n,b} \boldsymbol{\psi}_j \cdot \boldsymbol{n} \rangle_{\Gamma_D^t} \\ & + \sum_{F \in F_h^t} \gamma_v \left( \nu^{-1} + \nu \right) h^{-2} \left\langle \chi_{n,a} (\mathcal{E} \boldsymbol{\psi}_{i|K_1} - \mathcal{E} \boldsymbol{\psi}_{i|K_2}), \chi_{n,b} (\mathcal{E} \boldsymbol{\psi}_{j|K_1} - \mathcal{E} \boldsymbol{\psi}_{j|K_2}) \right\rangle_{\omega_F} \mathrm{d}t \\ & + \left\langle \chi_{n,a}((n-1)^+) \boldsymbol{\psi}_i, \chi_{n,b}((n-1)^+) \boldsymbol{\psi}_j \right\rangle_{\Omega_f(t_{n-1})}. \end{aligned} \tag{3.49}$$

These blocks represent the couplings of velocity trial and test functions.

The $\boldsymbol{B}$ blocks represent the pressure-velocity coupling. For $a, b \in [0, k]$, $i \in [1, J]$ and $j \in [1, S]$ these blocks can be expressed as:

$$\boldsymbol{B}_{a,b}^\top = \int_{t_{n-1}}^{t_n} \langle \chi_{n,1} \nabla \cdot \boldsymbol{\psi}_i, \chi_{n,1} \xi_j \rangle_{\Omega_f^t} - \langle \chi_{n,1} \nabla \boldsymbol{\psi}_i, \chi_{n,1} \xi_j \boldsymbol{n} \rangle_{\Gamma_D^t} \, \mathrm{d}t. \tag{3.50}$$

The $\boldsymbol{C}$ blocks denote the pressure-pressure coupling, that arises of the ghost stabilization in Sec. 3.2.3. For $a, b \in [0, k]$ and $i, j \in [1, S]$ it can be expressed as:

$$\boldsymbol{C}_{a,b} = \int_{t_{n-1}}^{t_n} \sum_{F \in F_h^t} \gamma_p \frac{1}{\nu} \left\langle \chi_{n,1}(\mathcal{E}\xi_{i|K_1} - \mathcal{E}\xi_{i|K_2}), \chi_{n,1}(\mathcal{E}\xi_{j|K_1} - \mathcal{E}\xi_{j|K_2}) \right\rangle_{\omega_F} \mathrm{d}t \,. \tag{3.51}$$

To enhance the range of convergence of Newton's method, a damped version using an additional linesearch technique is applied for solving Eq. (3.44). Alternatively, a "dogleg approach" (cf., e.g. [90]), that belongs to the class of trust-region methods and offers the advantage that also the search direction, not just its length, can be adapted to the nonlinear solution process, was implemented and tested. Both schemes require the computation of the Jacobian matrix of the algebraic counterpart of Eq. (4.9). In the dogleg method multiple matrix-vector products with the Jacobian matrix have to be computed. Since the Jacobian matrix is stored as a sparse matrix, the products can be computed at low computational costs. From the point of view of convergence, both methods yield a superlinear convergence behavior. In our numerical examples of Sec. 4.4, both modifications of Newton's method lead to comparable results. In our computational studies, we did not observe any convergence problems for these nonlinear solvers.

### 3.3.2 Globalization

In this subsection two methods to increase to convergence radius of the Newton scheme are presented. In Sec. 3.3.2.1 a classical backtracking line search and in Sec. 3.3.2.2 a dogleg method, which is a combination of Newton's method with the method of the steepest descent, is proposed. One can switch between them in the underlying code, with the change of a simple parameter. In the numerical examples in this paper, there occurred no proplems, regarding the convergence behavior of the Newton scheme and the iteration rates were comparable for both schemes. Since the backtracking scheme of Sec. 3.3.2.1 is slightly computationally cheaper, all benchmarks, especially in Chapter 4, were performed using this method.

#### 3.3.2.1 By Backtracking line search

The idea of a line search algorithm is simple to adjust the length of one Newton step while keeping its' direction, given by the solution of Eq. (3.36), to increase the

convergence radius of the Newton scheme. So in each Newton step at first Eq. (3.36) is solved and an initial increment $\boldsymbol{d}_{\tau,h}^{m+1}$ is determined. Afterwards this initial increment is adjusted until a solution is found, that results in an "acceptable" increment $\boldsymbol{d}_{\tau,h}^{m+1}$.

The algorithm for applying a backtracking line search on Eq. (3.36) is as follows (cf. [91], [92]):

**Algorithm 1:** Backtracking line search

Given $\alpha \in (0, \frac{1}{2}), \quad \lambda_k = 1$ ;

**while** $\left\|\boldsymbol{q}(\boldsymbol{u}_{\tau,h}^k + \lambda_m \boldsymbol{d}_{\tau,h}^{m+1}, \boldsymbol{\phi}_{\tau,h})\right\| > \left\|\boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h})\right\| + \alpha\lambda_k \left\|\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m} \boldsymbol{d}_{\tau,h}^{m+1}\right\|$ **do**

Update $\lambda_k$:

$$\lambda_m = \frac{\left\|\boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h})\right\|}{2*\left(\lambda_m \left\|\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m} \boldsymbol{d}_{\tau,h}^{m+1}\right\| + \left\|\boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h})\right\| - \left\|\boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m + \lambda_m \boldsymbol{d}_{\tau,h}^{m+1}, \boldsymbol{\phi}_{\tau,h})\right\|\right)}$$

**end**

Update $\boldsymbol{d}_{\tau,h}^{m+1}$:

$\boldsymbol{d}_{\tau,h}^{m+1} = \lambda_m \boldsymbol{d}_{\tau,h}^{m+1}$

In this algorithm the condition in the while-loop ensures, that the average rate of decrease from $\boldsymbol{u}_{\tau,h}^m$ to $\boldsymbol{u}_{\tau,h}^{m+1}$ is at least some prescribed fraction (depending on $\alpha$) of the initial rate of decrease in that direction and makes the scheme more robust ([91, p. 118]). In the implementation used for this thesis $\alpha$ was set to $\alpha = 1 \times 10^{-4}$. The scaling factor for the step length $\lambda_m$ is updated by finding the argument of the quadratic (one dimensional) polynomial $p(\lambda_m)$ that minimizes it. The polynomial coefficients are derived by a kind of Hermite interpolation regarding the following conditions:

$$\begin{aligned} p(0) &= \frac{1}{2}\left\|\boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h})\right\|^2, \qquad p(1) = \frac{1}{2}\left\|\boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m + \lambda_m \boldsymbol{d}_{\tau,h}^{m+1}, \boldsymbol{\phi}_{\tau,h})\right\|^2, \\ p'(0) &= \frac{1}{2}\lambda_m \left\|\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m} \boldsymbol{d}_{\tau,h}^{m+1}\right\|^2. \end{aligned} \tag{3.52}$$

We note, that if Algorithm 1 does not end after the first update, it is possible to model $p(\lambda_m)$ as a cubic polynomial at no additional costs, since the value of $\boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m + \lambda_m \boldsymbol{d}_{\tau,h}^{m+1}, \boldsymbol{\phi}_{\tau,h})$ is available at the position of the first iteration. According to [91, p. 128] this might be advantageous in scenarios where $\boldsymbol{q}$ has negative curvature. For the numerical examples in this work Algorithm 1 converge very quickly and such a cubic approach was not implemented.

#### 3.3.2.2 By Dogleg method

If in Algorithm 1 the full Newton step is refused, because our linear model is unsatisfactory, we look for a new step into the same direction but with a reduced step length, by building a quadratic model, based on the function and gradient information in the Newton direction ([91, p. 130]). The dogleg method combines the solution of the method of the steepest descent with the solution of the Newton scheme (of the linear model) to also vary the search direction, if the full Newton step is unsatisfactory. The method belongs to the kind of "trust region methods", where one tries to find a minimizer of the norm of the linearized problem within a given radius around a starting point. Typically a first initial guess for this radius is the $\|\cdot\|_2$ of the right-hand-side.

The direction of the steepest descent of the linear model in Eq. (3.36) at $\boldsymbol{u}_{\tau,h}^m$ is given by:

$$\boldsymbol{s}_{SD}^m = -\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m}^\top \boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h})\,. \tag{3.53}$$

The resulting minimizing problem reads as:

**Problem 3.5.** *Find $\lambda$ so that*

$$\left\| \boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h}) + \lambda \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m} \boldsymbol{q}(\boldsymbol{s}_{SD}^m - \boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h}) \right\| \stackrel{!}{=} min\,. \tag{3.54}$$

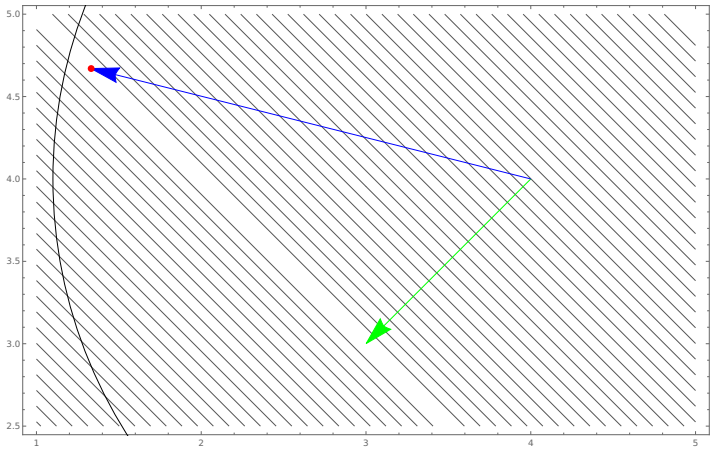
According to [91, p. 140] this problem has the unique solution:

$$\lambda^\star = \frac{\left\| \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m}^\top \boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h}) \right\|^2}{\boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h})^\top \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m} \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m}^\top \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m} \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}_{\tau,h}^m}^\top \boldsymbol{q}(\boldsymbol{u}_{\tau,h}^m, \boldsymbol{\phi}_{\tau,h})}\,. \tag{3.55}$$

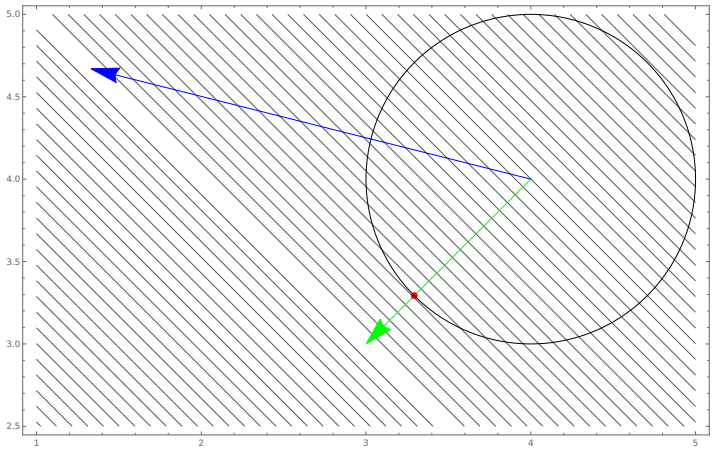
The corresponding solution is called "Cauchy step". There can occur three different scenarios, depending of the relative positions of the solutions, given by the Newton step and the Cauchy step. These scenarios are sketched in Fig. 3.7. The basic algorithm that is executed then is:

a If the length of the Newton step is smaller than the trust region size, so if $\left\|\boldsymbol{d}_{\tau,h}^{m+1}\right\| \leq \delta$, then the Newton step is taken directly. This is sketched in Fig. 3.7a.

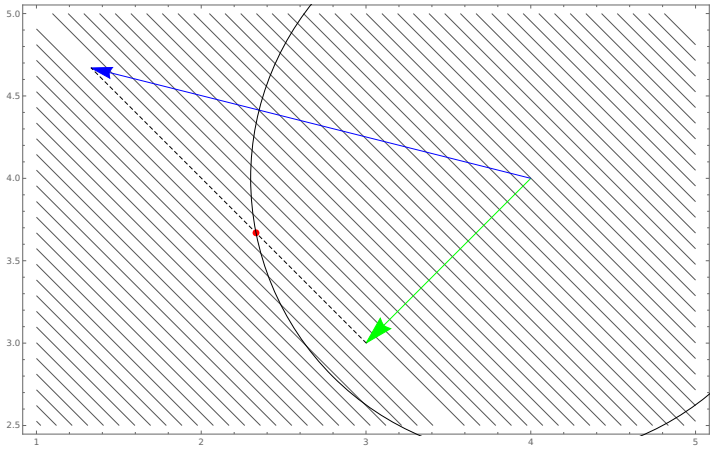
b If both, the length of the Newton step $\left\|\boldsymbol{d}_{\tau,h}^{m+1}\right\|$ and the length of the Cauchy step $\|\lambda^\star \boldsymbol{s}_{SD}^m\|$ are bigger than the trust region size $\delta$, then set the step length to $\delta$ and go into the steepest descent direction, so set $\boldsymbol{u}_{\tau,h}^{m+1} = \boldsymbol{u}_{\tau,h}^m + \frac{\delta}{\|\boldsymbol{s}_{SD}^m\|}\boldsymbol{s}_{SD}^m$. This is sketched in Fig. 3.7b.




(a) Newton step within trust region.




(b) Newton and Cauchy step outside trust region.




(c) Newton step outside, Cauchy step within trust region.

Figure 3.7: Three cases that can occur in the dogleg algorithm. The radius of the trust region is indicated by the circle, the Newton step is marked with a blue arrow and the Cauchy step with a green arrow. The chosen point is marked with a red dot.

c If just the length of the Newton step $\left\|\boldsymbol{d}_{\tau,h}^{m+1}\right\|$ is bigger than the trust region size $\delta$, linearly combine the Newton and the Cauchy step, so that the resulting step lies on the line, connecting the Newton and the Cauchy step with distance $\delta$ to $\boldsymbol{u}_{\tau,h}^{m}$, see Fig. 3.7c. This can be written as $\boldsymbol{u}_{\tau,h}^{m+1} = \boldsymbol{u}_{\tau,h}^{m} + \lambda^{\star}\boldsymbol{s}_{SD}^{m}(1-\eta) + \eta\boldsymbol{d}_{\tau,h}^{m+1}$, where $\eta$ is chosen such, that $\delta = \left\|\boldsymbol{u}_{\tau,h}^{m} + \lambda^{\star}\boldsymbol{s}_{SD}^{m} + \eta(\boldsymbol{d}_{\tau,h}^{m+1} - \lambda^{\star}\boldsymbol{s}_{SD}^{m})\right\|$ is fulfilled, see Remark 3.6.

**Remark 3.6.** *To calculate $\eta$, the quadratic equation*

$$\delta^2 = \left\|\boldsymbol{u}_{\tau,h}^{m} + \lambda^{\star}\boldsymbol{s}_{SD}^{m} + \eta(\boldsymbol{d}_{\tau,h}^{m+1} - \lambda^{\star}\boldsymbol{s}_{SD}^{m})\right\|^2$$

*is solved for the positive root (see [91, p. 142]). With*

$$\boldsymbol{a} \coloneqq \boldsymbol{u}_{\tau,h}^{m} + \lambda^{\star}\boldsymbol{s}_{SD}^{m}\,, \qquad\qquad \boldsymbol{b} \coloneqq \boldsymbol{d}_{\tau,h}^{m+1} - \lambda^{\star}\boldsymbol{s}_{SD}^{m}$$

*this problem can be rewritten as: $\delta^2 = \|\boldsymbol{a} + \eta\boldsymbol{b}\|^2$. With the components $a_i$ and $b_i$ of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ this can be expressed as*

$$\begin{aligned}\delta^2 &= \sum_i (a_i + \eta b_i)^2 \\ &= \eta^2 \sum_i b_i^2 + \eta \sum_i 2a_i b_i + \sum_i a_i^2 \\ &= \eta^2 \langle \boldsymbol{b}, \boldsymbol{b}\rangle + \eta 2 \langle \boldsymbol{a}, \boldsymbol{b}\rangle + \langle \boldsymbol{a}, \boldsymbol{a}\rangle \ .\end{aligned}$$

*The positive solution $\eta^{\star}$ is then given by:*

$$\eta^{\star} = \frac{-2\cdot\langle \boldsymbol{a}, \boldsymbol{b}\rangle + \sqrt{4\cdot\langle \boldsymbol{a}, \boldsymbol{b}\rangle^2 - 4\cdot\langle \boldsymbol{b}, \boldsymbol{b}\rangle\cdot(\langle \boldsymbol{a}, \boldsymbol{a}\rangle - \delta^2)}}{2\cdot\langle \boldsymbol{b}, \boldsymbol{b}\rangle}\,.$$

**Remark 3.7.** *In the productive code the underlying algorithm is slightly more involved and assures convergence by also dynamically adjusting the size of the trust region $\delta$. We refer to [93, p. 10ff.].*

## 3.4 Implementational aspects

In this section we address key aspects of the implementation of the presented CutFEM higher order space-time approach with arbitrary polynomial degree in time and space.

We use a software architecture that is built upon the C++ library *deal.II* [57] combined with the linear algebra package *Trilinos* [94] under a Message Passing Interface (MPI) parallelization. All routines for the assembly of the Newton-linearized algebraic system and the linear algebraic solver are parallelized and can be run on standard multicore/-processor architectures. Here, the mesh is partitioned into the number of MPI processes and each of these processes owns the cells and corresponding matrix and vector entries of its partition (and some additional overhead). Basically, the simulation can be run with an arbitrary number of MPI processes. For the simulations of this work we distributed exactly one MPI process to one physical CPU core.

Now we firstly address the integration over cut cells which is an important ingredient of the practical realization of the CutFEM approach. Then, key blocks of the code are discussed.

### 3.4.1 Integration over cut cells

In Problem 3.3 or Problem 3.4, respectively, the stabilized discrete form $A_h^s$, that is defined in Eq. (3.19) along with Eqs. (3.10) and (3.14), evokes the computation of integrals over the fluid domain. Due to the usage of unfitted meshes, integrals over the portions of the cut cells, that are filled with fluid, have thus to be evaluated. Fig. 3.8 illustrates schematically the types of cell intersections that can be induced by the motion of the rigid body. For quadrilateral finite elements, that we use in our implementation, the fluid cell portions can be pentagons, general quadrilaterals or triangles. We recall that we have assumed (cf. Assumption 3.2), that each face of the quadrilateral is cut at most once by the boundary of the rigid body (cf. Fig. 3.8). The type of the cut cell is determined by evaluating the levelset function, for $t \in (0, T)$,

$$\theta(\boldsymbol{x}, t) \begin{cases} < 0 & \forall\, \boldsymbol{x} \in \Omega_r^t\,, \\ = 0 & \forall\, \boldsymbol{x} \in \partial\Omega_r^t\,, \\ > 0 & \forall\, \boldsymbol{x} \in \Omega_f^t \end{cases} \tag{3.56}$$

in the corners of each cell.

In order to integrate over such cut cells we apply iterated one-dimensional numerical integration schemes. This is illustrated in Fig. 3.9 and reads as

$$\iint_{K_{\text{fluid}}} f(x_1, x_2)\mathrm{d}(x_1, x_2) \approx \text{meas}_2(K_{\text{fluid}}) \sum_{\mu_1=1}^{M_1} \sum_{\mu_2=1}^{M_2} \omega_{\mu_1}\omega_{\mu_2} f(x_{1,\mu_1}, x_{2,\mu_2})\,, \tag{3.57}$$

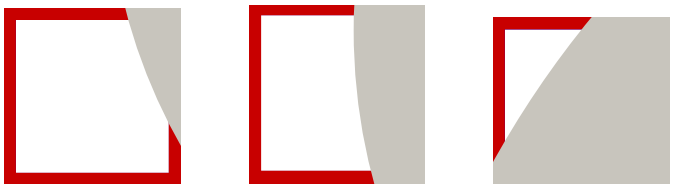



Figure 3.8: Types of arising cut cells with gray shaded rigid body and white shaded fluid domain.

where $\omega_{\mu_1}$ and $x_{1,\mu_1}$, for $\mu_1 = 1, \ldots, M_1$, as well as $\omega_{\mu_2}$ and $x_{2,\mu_2}$, for $\mu_2 = 1, \ldots, M_2$, denote the weights and quadrature nodes of the quadrature formula for the respective coordinate direction and $K_{\text{fluid}}$ is the fluid portion of the cut cell $K$. Here, we use the Gauss quadrature formula. The number $M_1$ and $M_2$ of quadrature points in the coordinate directions, and thus the degree of exactness of the quadrature rules, can be chosen independently of each other, the number $M_2$ of quadrature nodes in $x_2$ direction can even depend on $\mu_1$, i.e., on the quadrature node $x_{1,\mu_1}$ in $x_1$ direction. We note that the integration by numerical quadrature is restricted to the fluid portions of the cut cells by adapting the interval lengths in either coordinate direction. The flexible choice of the degree of exactness of the iterated integration formulas then allows an accurate integration over cut cells. In the implementation, the direction $x_1$ and $x_2$ of the iterated integration is adapted to the respective element. Precisely, the direction of the largest element face (i.e., longest side in two space dimensions) bounding the fluid portion of the cut cell (cf. Fig. 3.9), is chosen for the outer summation in Eq. (3.57). For this we recall that in this work the assumption was made that the evolving domain and thus the position of the moving boundary $\Gamma_r^t$ (cf. Fig. 3.1) is prescribed explicitly and, therefore, is computable; cf. Assumption 3.1. Further, a structured grid of elements aligned to the coordinate lines was supposed (cf. Sec. 3.2.1). In our implementation, the quadrature points of each cut cell are computed in each time step of Problem 3.3 before the assembly of the algebraic system is done. In the assembly routine, these points are then used to build a custom quadrature rule using *deal.II* routines.

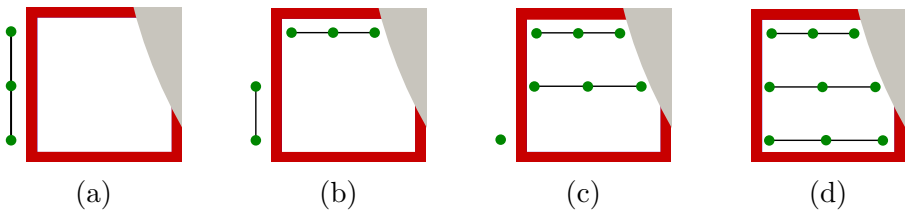



Figure 3.9: Quadrature nodes of iterated numerical integration over cut cells by Gauss quadrature formulas.

Regarding the accuracy of the presented iterated integration we note the following.

**Remark 3.8.** *The numerical results presented in Sec. 3.5 show that the space-time convergence behavior of the CutFEM approach is not deteriorated by the application of the iterated numerical integration on the cut cells. This also holds if irregular (tiny) cuts are present.*

**Remark 3.9.** *For rigid domains $\Omega_r^t$ with curved boundary, a cut scenario as sketched in Fig. 3.10 can occur.*

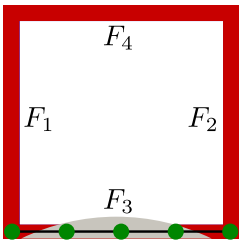



Figure 3.10: One face cut twice.

*Here, the face $F_3$ of the cell is cut twice. If $\Omega_r$ is time-independent, such a scenario is avoided by using a sufficiently fine mesh. Nevertheless, when the rigid domain $\Omega_r^t$ evolves in time, such cell intersections might arise. In our implementation, it is avoided by the following algorithm. We allow only the types of cell cuts that are illustrated in Fig. 3.8. To ensure the conformity of the arising cuts with these patterns, we evaluate the levelset function Eq. (3.56) not only in the corners of each cell, but also in the $n$ Gauss-Lobatto points on each face of the corresponding cell. This is sketched in Fig. 3.10 for $n = 5$, which we use in all of the simulations presented in Sec. Sec. 3.5. If the levelset function coincides in both corners of the face, the values in the interior Gauss-Lobatto nodes of this face are checked to coincide with these value as well. If this condition is fulfilled, it is assumed that this face is not cut. Otherwise, the face is supposed to be cut more than once. For time-independent domains, either a mesh refinement or repositioning of the rigid domain $\Omega_r$ is performed. For evolving domains, a simple algorithm that slightly adjusts the time step size for this single time step in order to avoid such a scenario is applied. We explicitly note that cut scenarios as illustrated in Fig. 3.10 are very unlikely to occur, especially when the mesh size is chosen sufficiently fine.*

**Remark 3.10.** *The integration over the boundary $\Gamma_r^t$ is performed using a standard parametrization of the circular rigid domain $\Omega_r^t$. With this we transform the surface*

*integral into a 1D integral that we numerically evaluate using a Gaussian quadrature rule.*

### 3.4.2 Key code blocks

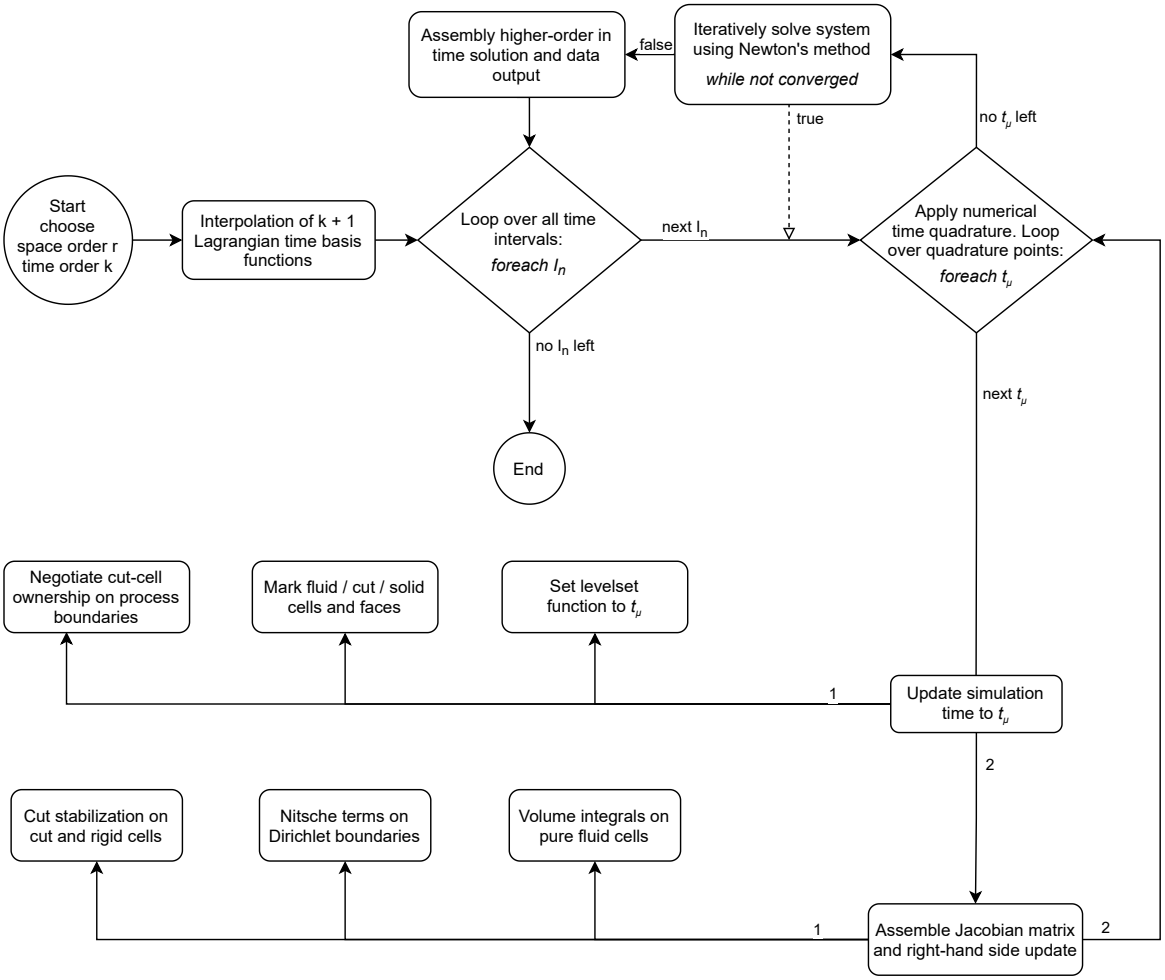



Figure 3.11: Key blocks of the CutFEM solver with arbitrary polynomial order in space and time.

Here, we describe the key blocks of the implementation of our CutFEM approach. A flowchart of the code is given in Fig. 3.11. After execution of the program and before the loop of the time marching process of Problem 3.3 is started, a nodal Lagrangian time basis $\{\chi_l\}_{l=0}^k$ is computed with respect to the $(k+1)$ Gauss–Radau quadrature nodes by solving the corresponding interpolation problem. Actually, this is done on the reference interval $\hat{I} = [0,1]$, such that $\{\hat{\chi}_l\}_{l=0}^k$, with $\hat{\chi}_l \in \mathbb{P}_k(\hat{I};\mathbb{R})$, satisfies

$$\hat{\chi}_l(\hat{t}_\mu) = \delta_{l,\mu}\,, \quad l,\mu = 0\ldots,k\,,$$

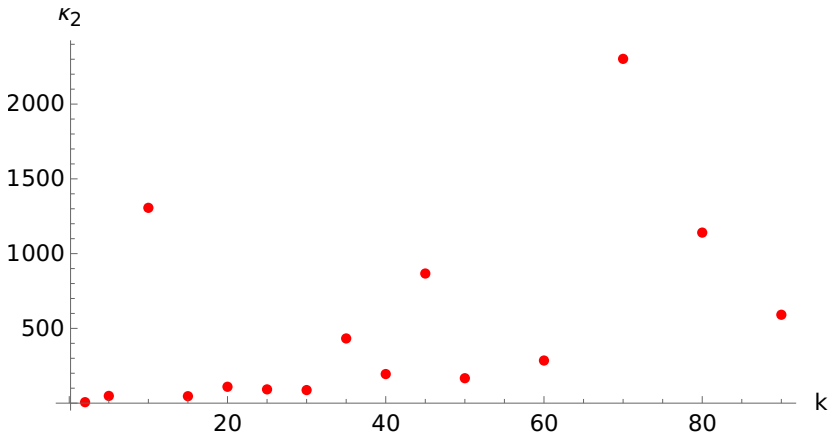



Figure 3.12: Spectral condition number $\kappa_2$ for the computation of the nodal $dG(k)$ time basis.

with the Kronecker symbol $\delta_{l,\mu}$ and the $(k+1)$ Gauss–Radau quadrature nodes $\{\hat{t}_\mu\}_{\mu=0}^{l}$ of $\hat{I}$. The resulting linear system is solved by using Linear Algebra PACKage (LAPACK) routines. Since the support points are non-equally distributed, the spectral condition number $\kappa_2$ of the resulting Vandermonde matrix is feasible even for high values of $k$. This is illustrated in Fig. 3.12. In each subinterval of the time marching process of Problem 3.3, numerical quadrature is applied for the integration in the time and space domain. Firstly, the subinterval and its quadrature nodes are incremented to $I_n$ and $t_{n,\mu}$, for $\mu = 0, \ldots, k$. For the representation of the rigid domain $\Omega_r^t$ we use the levelset function $\theta(\boldsymbol{x}, t)$, defined in Eq. (3.56). By evaluating this function we identify whether a finite element cell belongs at time $t_{n,\mu}$ to the set of fluid cells $\mathcal{T}_{h,f}^{t_{n,\mu}}$, rigid cells $\mathcal{T}_{h,r}^{t_{n,\mu}}$ or cut cells $\mathcal{T}_{h,c}^{t_{n,\mu}}$. In subroutines, we mark each cell of the computational background grid by evaluating the levelset function in the edges of each cell. For cut cells, we pre-compute the quadrature points and weights for the (fluid) volume and surface integrals (line integrals in two space dimensions) and store them into lookup tables. In the MPI based implementation these tasks scale perfectly with the number of available processes.

**Remark 3.11.** *We note that the levelset function Eq. (3.56) is not discretized or computed by the approximation of a suitable transport equation. The direct evaluation of Eq. (3.56) becomes feasible due to the Assumption 3.1 that the motion of the rigid body is prescribed. Thereby, further discretization errors are avoided.*

However, the parallelized code needs some mechanism to prevent that the ghost penalty operator defined in Eq. (3.17) is applied twice in the interface zone of MPI process boundaries of the mesh partition. For this we use a simple master-slave approach and

let the process with the higher process number assemble the ghost penalty operator over patches $\omega_F$ at process partition boundaries. This problem is sketched in Fig. 3.13 for the patch $\omega_F$, built from the cells $K_1$ and $K_2$. Cell $K_1$ of the patch belongs to process 0 and cell 2 to process 1. Without a control of the parallel assembly of the ghost penalty stabilization, both processes would assemble the contributions of Eq. (3.17), such that the stabilization would be applied twice. To avoid this, the process with the higher process number (process 1 in this case) assembles the ghost penalty stabilization.

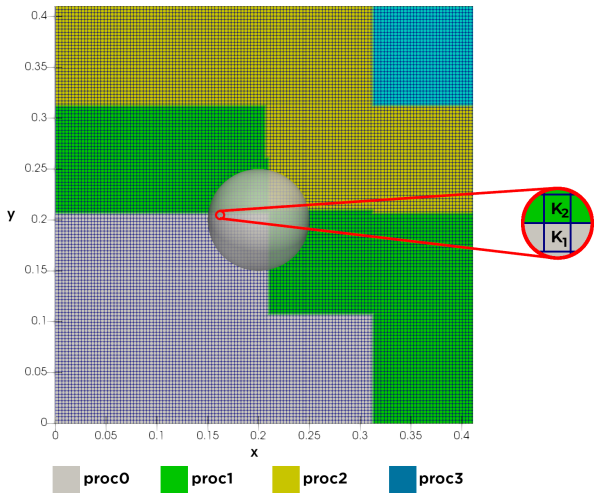



Figure 3.13: Assembly of the ghost penalty stabilization at partition boundaries with computational mesh $\mathcal{T}_h$ and colored processor ownerships of mesh cells.

In the next block the linear system of the Newton iteration (cf. Remark 3.5) is assembled. For the integration over fluid cut cells iterated integration, as sketched in Sec. 3.4.1, is applied. Then, the resulting linear system is solved by the parallel, sparse direct solver SuperLU_DIST [89] and, unless the stopping criteria is satisfied, the Newton iteration is continued by reassembling the linear system. After convergence of the Newton iteration, a parallel assembly and data output of the computed higher order solution for the subinterval $I_n$ terminates the time step.

## 3.5 Numerical experiments

In this section we present the results of four numerical experiments to illustrate the performance and accuracy properties of the proposed CutFEM approach. We start with a space-time convergence test with a moving $\Omega_r^t$, to validate all aspects of the implementation. In the second experiment, the popular DFG benchmark of flow around a cylinder (cf. [46]) is considered. A careful comparative study of the CutFEM approach

on background meshes with the results obtained for body fitted meshes is done. In the last two test cases we investigate the quality of the ghost penalty stabilization and extension of Eq. (3.17) on an evolving domain and demonstrate it's application in a realistic example. Throughout all simulations we set the viscosity to $\nu = 0.001$.

In all the examples, the Newton solver uses an absolute value of the residual smaller than $1 \times 10^{-10}$ as the stopping criteria. In Sec. 3.5.4, it is checked additionally if the initial residual of a time step is decreased by a factor of 1000. For all our numerical experiments we set the Nitsche penalty parameters of Eq. (3.13) to $\gamma_1 = \gamma_2 = 35$, which is motivated by the numerical experiments in [28], [30]. The ghost penalty parameters of Eq. (3.17) are set to $\gamma_v = \gamma_p = 10^{-2}$, except in Sec. 3.5.6, where we set $\gamma_v = \gamma_p = 1$.

For the convergence tests, we define the errors

$$\boldsymbol{e}^{\boldsymbol{v}}(t) \coloneqq \boldsymbol{v}(t) - \boldsymbol{v}_{\tau,h}(t)\,, \qquad e^p(t) \coloneqq p(t) - p_{\tau,h}(,t)\,. \tag{3.58}$$

We study the error $(\boldsymbol{e}^{\boldsymbol{v}}, e^p)$ on the fluid domain $\Omega_f^t$ with respect to the norms

$$\|e^w\|_{L^\infty(L^2)} \coloneqq \max_{t\in I} \left( \int_{\Omega_f^t} \|e^w\|^2 \mathrm{d}\boldsymbol{x} \right)^{\frac{1}{2}}, \quad \|e^w\|_{L^2(L^2)} \coloneqq \left( \int_I \int_{\Omega_f^t} \|e^w(t)\|^2 \mathrm{d}\boldsymbol{x}\,\mathrm{d}t \right)^{\frac{1}{2}}, \tag{3.59}$$

where $w \in (\boldsymbol{v}, p)$. The $L^\infty$-norm in time is computed on the discrete time grid

$$I = \left\{ t_n^d \mid t_n^d = t_{n-1} + d \cdot k_n \cdot \tau_n, \quad k_n = 0.001\,,\ d = 0, \ldots, 999\,,\ n = 1, \ldots, N \right\}. \tag{3.60}$$

### 3.5.1 Experimental order of convergence, time-independent domain

To validate our implementation with respect to the space-time discretization and application of Nitsche's method, we start with a standard convergence test. We prescribe the initial value $\boldsymbol{v}(\boldsymbol{x}, 0)$ and right-hand side function $\boldsymbol{f}$ on $\Omega_f \times I = (0,1)^2 \times (0,1]$ in such a way, that the exact solution of the Navier–Stokes system (3.9) is given by

$$\begin{aligned} \boldsymbol{v}_e(\boldsymbol{x}, t) &\coloneqq \begin{pmatrix} \cos(x_2\pi) \cdot \sin(t) \cdot \sin(x_1\pi)^2 \cdot \sin(x_2\pi) \\ -\cos(x_1\pi) \cdot \sin(t) \cdot \sin(x_2\pi)^2 \cdot \sin(x_1\pi) \end{pmatrix}, \\ p_e(\boldsymbol{x}, t) &\coloneqq \cos(x_2\pi) \cdot \sin(t) \cdot \sin(x_1\pi) \cdot \cos(x_1\pi) \cdot \sin(x_2\pi)\,. \end{aligned} \tag{3.61}$$

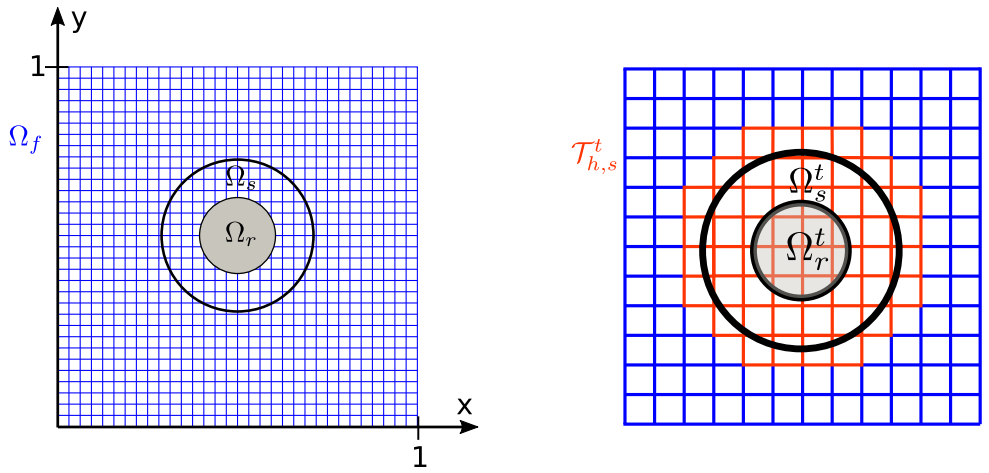
Thus, homogeneous Dirichlet boundary condition are prescribed on $\partial\Omega_f$. Table 3.1 shows the computed errors and experimental orders of convergence for different polynomial degrees in space and time. For the definition of the discrete function spaces we refer to Chapter 2. The expected convergence of optimal order is observed.

Table 3.1: Experimental order of convergence for different $dG(k)$ schemes

| $\tau$ | $h$ | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 2.789e−5 | – | 1.360e−4 | – | 4.020e−4 | – | 1.641e−3 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 8.802e−7 | 4.99 | 7.385e−6 | 4.20 | 3.178e−5 | 3.66 | 2.178e−4 | 2.91 |
| $\tau_0/2^2$ | $h_0/2^2$ | 2.755e−8 | 5.00 | 4.422e−7 | 4.06 | 3.122e−6 | 3.35 | 2.973e−5 | 2.87 |
| $\tau_0/2^2$ | $h_0/2^2$ | 8.610e−9 | 5.00 | 2.745e−8 | 4.01 | 3.556e−7 | 3.13 | 3.882e−6 | 2.94 |
| elements$_{\|I_n}$ | | $(\mathbb{P}_4(I_n;H_h^4))^2 \times \mathbb{P}_4(I_n;H_h^3)$ | | | | $(\mathbb{P}_2(I_n;H_h^3))^2 \times \mathbb{P}_2(I_n;H_h^2)$ | | | |

## 3.5.2 Influence of the size of the stabilization zone

In the second numerical experiment the effect of the cut cell integration along with the ghost stabilization is computationally analyzed. Further, analytical results of e.g. [26] showed, that the size of the area, where the ghost penalty operator of Eq. (3.19) is applied influences the quality of the extension. This effect is also computationally studied in this example. Fig. 3.14 shows the setup of this test, which contains a time-




(a) Problem setting. (b) Submesh $\mathcal{T}^t_{h,s}$ of stabilization.

Figure 3.14: Problem setting, domain $\Omega^t_s$ and submesh $\mathcal{T}^t_{h,s}$ of stabilization.

independent domain $\Omega_r$. One aim of this experiment is to study the impact of the width of the domain of stabilization $\Omega^t_s$ on the convergence of the scheme proposed in Sec. 3.2. Precisely, if a wider overlapping of the fluid domain $\Omega^t_f$ by $\Omega^t_s$ is required to

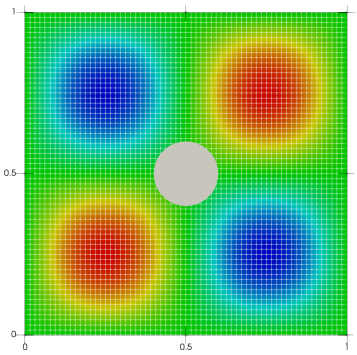



Figure 3.15: Pressure field at $t = 1$ for $h = h_0/2^4$.

ensure convergence of optimal order. We put $\Omega \times I = (0,1)^2 \times (0,1]$. The midpoint of the circular rigid body with radius $r_{\Omega_r} = 0.1$ is located in the center of $\Omega$. We prescribe the initial value $\boldsymbol{v}_0$ and right-hand side function $\boldsymbol{f}$ on $\Omega_f \times I$ in such a way, that the solution of the Navier–Stokes system on $\Omega_f$ is given by Eq. (3.61). On the inner fluid boundary we prescribe the condition $\boldsymbol{v} = \boldsymbol{v}_e$, on the outer boundary we prescribe a homogeneous Dirichlet condition. Table 3.2 shows the computed errors and experimental orders of convergence (EOC) for two different combinations of space-time finite elements, based on the Taylor–Hood family $(H_h^r)^2 \times H_h^{r-1}$, $r \geq 2$ in space, and two different choices of the radius $r_{\Omega_s}$ of the domain $\Omega_s$ where the stabilization $S_{F_h^t}$ is applied, precisely $r_{\Omega_s} = 1 \cdot r_{\Omega_r}$ (*bottom*) and $r_{\Omega_s} = 2 \cdot r_{\Omega_r}$ (*top*).

Table 3.2: Errors and experimental order of convergence for varying radius $r_{\Omega_s}$ of the domain $\Omega_s$ for $\tau_0 = 1.0$ and $h_0 = 1/(2\sqrt{2})$.

| $\tau$ | $h$ | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 1.514e−2 | – | 3.719e−2 | – | 1.581e−3 | – | 8.851e−3 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 3.764e−3 | 2.01 | 9.213e−3 | 2.01 | 2.234e−4 | 2.82 | 1.174e−3 | 2.92 |
| $\tau_0/2^2$ | $h_0/2^2$ | 9.100e−4 | 2.05 | 2.156e−3 | 2.09 | 2.972e−5 | 2.91 | 1.531e−4 | 2.94 |
| $\tau_0/2^3$ | $h_0/2^3$ | 2.340e−4 | 1.96 | 5.524e−4 | 1.96 | 3.673e−6 | 3.02 | 1.962e−5 | 2.96 |
| $\tau_0/2^4$ | $h_0/2^4$ | 5.792e−5 | 2.01 | 1.381e−4 | 2.00 | 4.666e−7 | 2.98 | 2.479e−6 | 2.98 |
| $\tau_0/2^5$ | $h_0/2^5$ | 1.448e−5 | 2.00 | 3.477e−5 | 1.99 | 5.833e−8 | 3.00 | 3.120e−7 | 2.99 |
| elements$_{\|I_n}$ | | $(\mathbb{P}_1(I_n;H_h^2))^2 \times \mathbb{P}_1(I_n;H_h^1)$, $r_{\Omega_s} = 2 \cdot r_{\Omega_r}$ | | | | $(\mathbb{P}_2(I_n;H_h^3))^2 \times \mathbb{P}_2(I_n;H_h^2)$, $r_{\Omega_s} = 2 \cdot r_{\Omega_r}$ | | | |

| $\tau$ | $h$ | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 1.540e−2 | – | 3.045e−2 | – | 1.552e−3 | – | 8.794e−3 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 5.762e−3 | 1.42 | 9.503e−3 | 1.68 | 2.191e−4 | 2.83 | 1.925e−3 | 2.19 |
| $\tau_0/2^2$ | $h_0/2^2$ | 1.812e−3 | 1.67 | 2.248e−3 | 2.08 | 2.194e−5 | 3.32 | 1.743e−4 | 3.46 |
| $\tau_0/2^3$ | $h_0/2^3$ | 5.004e−4 | 1.86 | 5.580e−4 | 2.01 | 5.293e−6 | 2.05 | 4.302e−5 | 2.03 |
| $\tau_0/2^4$ | $h_0/2^4$ | 1.316e−4 | 1.93 | 1.444e−4 | 1.95 | 7.240e−7 | 2.87 | 5.723e−6 | 2.91 |
| $\tau_0/2^5$ | $h_0/2^5$ | 3.290e−5 | 2.00 | 3.636e−5 | 1.99 | 9.240e−8 | 2.97 | 7.203e−7 | 2.99 |
| elements$_{\|I_n}$ | | $(\mathbb{P}_1(I_n;H_h^2))^2 \times \mathbb{P}_1(I_n;H_h^1)$, $r_{\Omega_s} = 1 \cdot r_{\Omega_r}$ | | | | $(\mathbb{P}_2(I_n;H_h^3))^2 \times \mathbb{P}_2(I_n;H_h^2)$, $r_{\Omega_s} = 1 \cdot r_{\Omega_r}$ | | | |

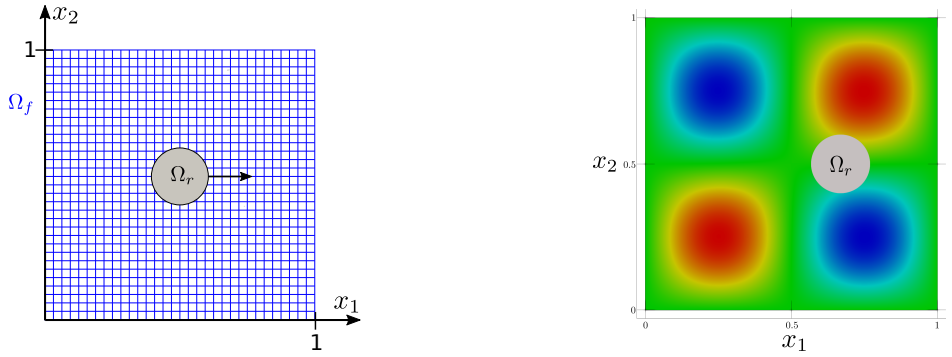
Even though the absolute errors are smaller for $r_{\Omega_s} = 2 \cdot r_{\Omega_r}$ (*top*), our results indicate that the EOC does not depend on the diameter of the region where the stabilization $S_{F_h^t}$ is applied. We note that in our experiments cut cells with non-convex boundary segments due to intersections with $\Omega_r^t$ occur, cf. Fig. 3.14. The convergence does not suffer from these cells.

### 3.5.3 Experimental order of convergence, time-dependent domain

The aim of the third numerical experiment is to verify all components of our approach, in particular the integration over cut cells as well as the ghost penalty stabilization on a time dependent domain in a space-time convergence study. We consider the problem setting sketched in Fig. 3.16a with a time-dependent domain $\Omega_r^t$. We put $\Omega \times I = (0,1)^2 \times (0,1]$. The rigid body is modeled by a circular domain of radius of $r = 0.1$ and an initial position of its center at $\boldsymbol{x}_r(0) = (0.5, 0.5)^\top$. The motion of the center is prescribed by

$$\boldsymbol{x}_r(t) = \boldsymbol{x}_r(0) + (A \cdot \sin(\omega \cdot t), 0)^\top \,, \tag{3.62}$$

with $A = 0.2$ and $\omega = 1$. We prescribe the initial value $\boldsymbol{v}_0$ and right-hand side function $\boldsymbol{f}$ on $\Omega_f \times I$ in such a way, that the exact solution of the Navier–Stokes system $\boldsymbol{v}_e$ on $\Omega_f$ is given by Eq. (3.61). On the inner fluid boundary we prescribe the condition $\boldsymbol{g}_r = \boldsymbol{v}_e$ (see Eq. (3.61)), which is an artificial condition and not physically motivated. On the outer boundary we use a homogeneous Dirichlet condition. Table 3.3 shows the computed errors and experimental orders of convergence (EOC) for two different combinations of space-time finite elements, based on the Taylor–Hood family in space. In the first experiment, the polynomial order in time is adapted to the spatial approximation of the prerssure variable. It is chosen to $k = 1$ such that optimal second order of convergence in time and space is obtained for the pressure variable with discrete values in $H_h^1$. In the second experiment, the polynomial order in time is put to $k = 2$ such that optimal third order of convergence in time and space is obtained for the pressure variable with discrete values in $H_h^2 \times H_h^2$. For this we recall that a non-equal order, inf-sup stable discretization in space by the Taylor–Hood family of elements is used here. For the definition of the discrete function spaces we refer to Eqs. (2.5) and (2.9), respectively. In Fig. 3.16, the expected convergence of optimal order is documented for both experiments.




(a) Initial configuration of the space-time convergence test.

(b) Solution of the pressure field at $t = T$ with $h = \frac{h_0}{2^4}$.

Figure 3.16: Problem setup and pressure field at the end of the simulation.

Table 3.3: Errors and experimental order of convergence for $\tau_0 = 1.0$ and $h_0 = 1/(2\sqrt{2})$.

| $\tau$ | $h$ | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 4.314e−2 | – | 1.476e−2 | – | 3.420e−2 | – | 1.340e−2 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 8.602e−3 | 2.33 | 3.047e−3 | 2.28 | 7.833e−3 | 2.13 | 2.590e−3 | 2.37 |
| $\tau_0/2^2$ | $h_0/2^2$ | 2.063e−3 | 2.06 | 6.964e−4 | 2.13 | 9.459e−4 | 3.05 | 5.605e−4 | 2.21 |
| $\tau_0/2^3$ | $h_0/2^3$ | 4.882e−3 | 2.08 | 1.853e−4 | 1.91 | 1.151e−4 | 3.04 | 1.354e−4 | 2.05 |
| $\tau_0/2^4$ | $h_0/2^4$ | 1.221e−4 | 2.00 | 4.600e−5 | 2.01 | 1.428e−5 | 3.01 | 3.385e−5 | 2.00 |
| elements$_{\|I_n}$ | | $(\mathbb{P}_1(I_n;H_h^2))^2 \times \mathbb{P}_1(I_n;H_h^1)$ | | | | $(\mathbb{P}_2(I_n;H_h^2))^2 \times \mathbb{P}_2(I_n;H_h^1)$ | | | |

**Remark 3.12.** *The setting of this example was chosen in such a way, that the propagation of the boundary of $\Gamma_r^t$ is less than one cell in each time step, in order to preserve a CFL like condition. We didn't observe any decrease in the convergence rate by violating this condition. For instance, choosing $h_0 = 1/(4\sqrt{2})$, leads to comparable EOCs, but violates the assumption that the boundary of $\Gamma_r^t$ propagates over a distance less than one cell within a time step.*

### 3.5.4 Time periodic flow around a cylinder

The aim of the second numerical example is to analyze the effects of usage of cut cells for the space discretization on background meshes along with the extension to a rigid domain $\Omega_r$ and the application of the stabilization introduced in Eq. (3.17). For this, we use the well-known DFG benchmark setting of flow around a cylinder, defined in [46]. The spatial setup is shown in Fig. 3.17. Although the domain is non-evolving, evaluating the performance properties of the approach for this flow benchmark is of high interest. We compare the numerical results with the ones obtained for simulations on a body-fitted meshes that is highly pre-adapted to the cylinder. Quantities of interest

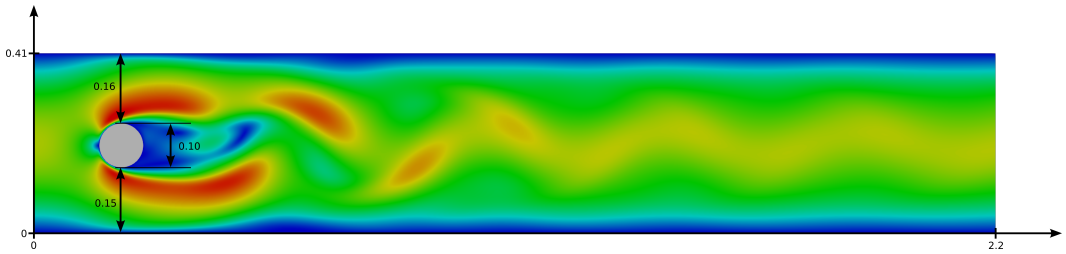



Figure 3.17: Geometrical setting of the 2d DFG benchmark with computed velocity profile of the fully developed flow.

and comparison in the simulations are the drag and lift coefficient of the flow on the circular cross section (cf. [46]). With the drag and lift forces $F_D$ and $F_L$ on the rigid circle $S$ given by

$$F_D = \int_S \left( \nu \frac{\partial \boldsymbol{v}_t}{\partial \boldsymbol{n}} n_y - P n_x \right) \mathrm{d}S \,, \qquad F_L = -\int_S \left( \nu \frac{\partial \boldsymbol{v}_t}{\partial \boldsymbol{n}} n_x - P n_y \right) \mathrm{d}S \,, \tag{3.63}$$

where $\boldsymbol{n}$ is the normal vector on $S$, pointing outside $S$. The tangential direction is defined as $\boldsymbol{t} = (n_y, -n_x)^\top$ and $\boldsymbol{v}_t$ is the velocity $\boldsymbol{v}$ projected into the direction of $t$. The drag and lift coefficient $c_D, c_L$ are defined by means of

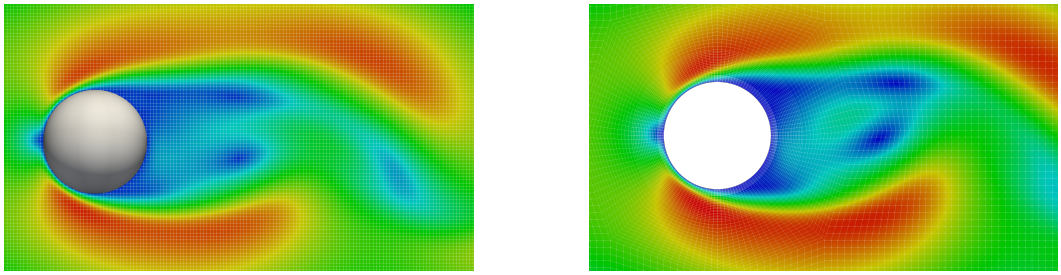
$$c_D = \frac{2}{\bar{U}^2 L} F_D \,, \qquad c_L = \frac{2}{\bar{U}^2 L} F_L \,. \tag{3.64}$$

**Remark 3.13.** *The integration of the surface integrals in Eq. (3.63) is performed on the surface $S$ of the cylinder, using a Gaussian quadrature rule, see Eq. (3.28). The number $m \in \mathbb{N}$ of quadrature points is chosen such that $m \geq (k+1)/2$ is satisfied.*
*As described in [50], [95]–[97], there exist ways compute the drag and lift coefficients by replacing the surface integrals with volume integrals. The idea is basically to test the momentum equation (see Eq. (3.3)) with non-conforming test functions. Afterwards an integration over $\Gamma_f^t$ and integration by parts is performed. This results in a volume integral based formulation for $c_D$ and $c_L$. Thereby, as shown in [96], the order of accuracy can be increased. In [95] it is further mentioned, that performing the surface integration in Eq. (3.63) over a discretized surface $S_h$ might influence the result.*
*As pointed out in Remark 3.10, in our CutFEM simulations we use an exact parametrization of the cylinder. Thereby we avoid making such a discretization error and perform the integration over the surface of the cylinder.*

According to [46], we set the boundary condition on the inflow boundary $\Gamma_i$ as

$$\boldsymbol{g}_i(x, y, t) = \left( \frac{4 \cdot 1.5 \cdot y(0.41 - y)}{0.41^2}, 0 \right)^\top .$$

This leads to a Reynolds number of $Re = 100$ and a time-periodic flow behavior. The space–time discretization is done in the discrete spaces $\mathbb{P}_1(I_n; H_h^2))^2 \times \mathbb{P}_1(I_n; H_h^1)$ on each time interval $I_n$. For the CutFEM approach we study a sequence of successive mesh refinements in space. For the developed flow and the computation of the drag and lift coefficient as quantities of physical interest we use the time step size $\tau = 0.005$.



(a) $\text{Cut}_7$ configuration with computational mesh.

(b) Fitted mesh, pre-adapted to the rigid body.

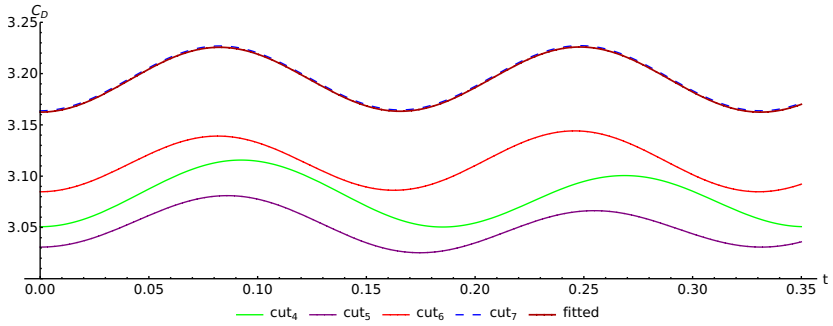
Figure 3.18: Portion of problem setting and computed velocity profile for CutFEM approach (Fig. 3.18a) and fitted mesh approximation (Fig. 3.18b).

Fig. 3.18 shows the computed flow profile and spatial mesh of a CutFEM simulation and a computation done on a highly pre-adapted, body-fitted mesh. In the latter case, high accuracy for the drag and lift coefficient is obtained (cf. [46]). The CutFEM simulation is done for a sequence of successive refinement steps of the entire background mesh (cf. Table 3.4). Table 3.4 shows the computed drag and lift coefficient as well as their frequency. Precisely, the latter denotes the frequency of the oscillation of the lift coefficient $c_L$, which is computed at a time instant $t_0$ where the lift coefficient $c_L$ is smallest and ends at a time instant $t_1 = t_0 + \frac{1}{f}$ when $c_L$ is smallest again. Exemplarily, Fig. 3.19 illustrates such a cycle for the drag and the lift coefficient. To simplify the interpretation of the graphs, the starting point of the monitoring cycle is set to $t = 0$.

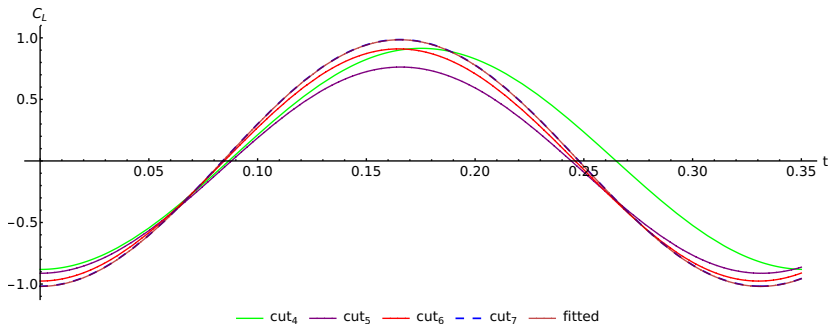
Comparing the results of the either approaches, CutFEM on background meshes versus adapted body-fitted mesh, we observe that the goal quantities of the CutFEM simulations nicely converge to the results computed on a manually pre-adapted and body-fitted mesh. Whereas the CutFEM background mesh is refined uniformly, the

Table 3.4: Computed minimum and maximum drag–lift coefficients and frequency for a sequence of successively refined CutFEM background meshes and an adapted, body-fitted mesh. Reference values: $c_{D_{max}} \in [3.2200, 3.2400]$ and $c_{L_{max}} \in [0.9900, 1.0100]$, see [46].

| Method | $\text{DoFs}_{\lvert I_n}$ | $\min(c_L)$ | $\max(c_L)$ | $\min(c_D)$ | $\max(c_D)$ | $f_L$ |
|---|---|---|---|---|---|---|
| $\text{cut}_4$ | 24 000 | −0.8814 | 0.9147 | 3.0507 | 3.1157 | 2.8433 |
| $\text{cut}_5$ | 94 086 | −0.9127 | 0.7626 | 3.0308 | 3.0810 | 3.0146 |
| $\text{cut}_6$ | 372 486 | −0.9759 | 0.9101 | 3.0848 | 3.1391 | 3.0274 |
| $\text{cut}_7$ | 1 482 246 | −1.0177 | 0.9856 | 3.1638 | 3.2272 | 3.0183 |
| Adapted, body-fitted | 502 464 | −1.0190 | 0.9843 | 3.1625 | 3.2259 | 3.0184 |




(a) Drag coefficients $c_D$.




(b) Lift coefficients $c_L$.

Figure 3.19: Computed drag and lift coefficients for the example of Sec. 3.5.4 and different time discretization schemes with basic time step size $\tau = 0.005$.

body-fitted mesh is highly adapted to the problem setting and flow profile. In particular, the region around the cylinder is strongly resolved by the spatial mesh whereas the CutFEM mesh remains much coarser in this region, even on the finest refinement level. In Fig. 3.18 both meshes are shown. The diameter of a quadratic cut cell of the $\text{Cut}_7$ configuration is 0.004 529 9. The diameter of a rectangular cell of the body-fitted

triangulation in the neighborhood of $\Omega_r$ is 0.002 495 03. Therefore, the superiority of the body-fitted approach for this test case with non-evolving domain is obvious. Nevertheless, Table 3.4 and Fig. 3.19 nicely show the numerical convergence of the CutFEM simulations for a successive refinement of the background mesh to the expected solution in terms of the confirmed drag and lift coefficient and their frequency (cf. [46]). This confirms the accuracy of the proposed CutFEM approach in the case of non-evolving domains.

### 3.5.5 Dynamic Poiseuille flow

In this numerical experiment the accuracy of the presented CutFEM approach is analyzed for an evolving domain. Precisely, we study the perturbation of a Poiseuille flow profile by the motion of an enclosed moving rigid body (cf. Fig. 3.1) and whether application of the ghost penalty operator leads to a disturbance of the solution. The test configuration is chosen in such a way, that the Poiseuille profile is preserved in the pipe, even though a moving rigid body is present. We than analyze if the Poiseuille flow is recovered by the CutFEM approach.

The computational domain is $\Omega = (0, l_x) \times (-l_y, l_y)$, with $l_x = 3, l_y = 0.5$, we let $I = (0, 40]$. A parabolic Poiseuille flow profile is prescribed at the left inflow boundary, that is given by

$$\boldsymbol{v}_p(x, y) = U_{\text{in}}(l_y^2 - y^2, 0)^\top , \tag{3.65}$$

with $U_{\text{in}} = 1$. The initial value is $\boldsymbol{v}_0 = \mathbf{0}$. The extension of Eq. (3.65) to the entire pipe is the Poiseuille flow and satisfies the Navier–Stokes equations along with the pressure $p_p = -2\nu U_{\text{in}}(x - l_x)$. The rigid body is modeled by a circular domain of radius of $r = 0.2$ and an initial position of its center at $\boldsymbol{x}_r(0) = (1.545, 0)^\top$. The motion of the ball's center is prescribed by Eq. (3.62), with $A = 0.8$ and $\omega = 0.2$. On the boundary of the rigid body the Dirichlet condition

$$\boldsymbol{v} = \boldsymbol{g}_r \coloneqq \boldsymbol{v}_p \tag{3.66}$$

is prescribed. After a transition from the zero initial state, the Poiseuille flow profile is expected to develop, and it is not perturbed by the rigid body motion due to the choice of the boundary condition Eq. (3.66). Clearly, even though the rigid body is moving inside the fluid by means of Eq. (3.62), its impact on the fluid flow is hidden by means of the boundary condition in Eq. (3.66).

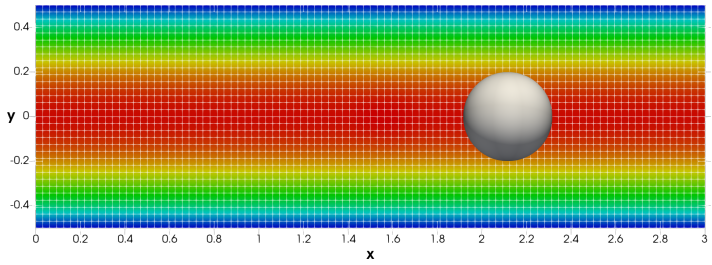



Figure 3.20: Problem setting and flow profile at $T = 40$.

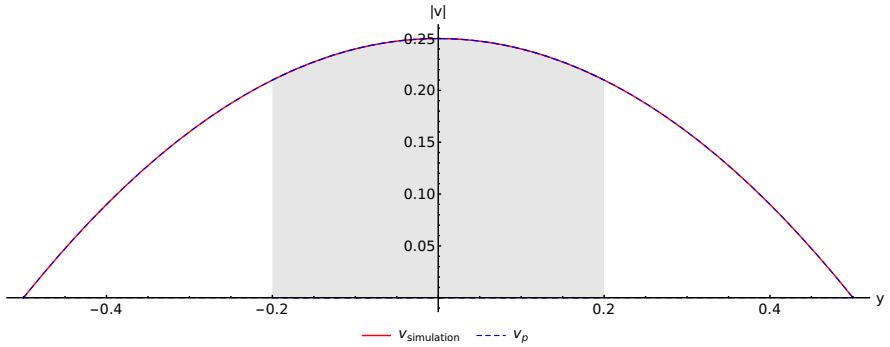
We compute fully discrete approximations in $(\mathbb{P}_1(I_n; H_h^2))^2 \times \mathbb{P}_1(I_n; H_h^1)$, for $n = 1, \ldots, N$, for the time step size $\tau = 0.1$ and a background mesh with 223 750 space-time degrees of freedom in each time step. The mesh and the computed flow profile at the final simulation time $T = 40$ are illustrated in Fig. 3.20. This solution is fully converged. In fact, the Poiseuille profile is nicely recovered by the simulation. Fig. 3.21 shows the cross sections plots in $y$ and $x$ coordinate directions, respectively, of the flow and pressure profile computed by CutFEM. The position of the rigid body domain is gray-shaded. In this domain the velocity and pressure values are computed by the extension and ghost penalty stabilization operator defined in Eq. (3.17). The computed profiles and the Poiseuille profile nicely coincide which clearly demonstrates the accuracy of the suggested CutFEM approach and efficiency of the combined extension and stabilization in Eq. (3.17).

### 3.5.6 Dynamic flow around moving cylinder

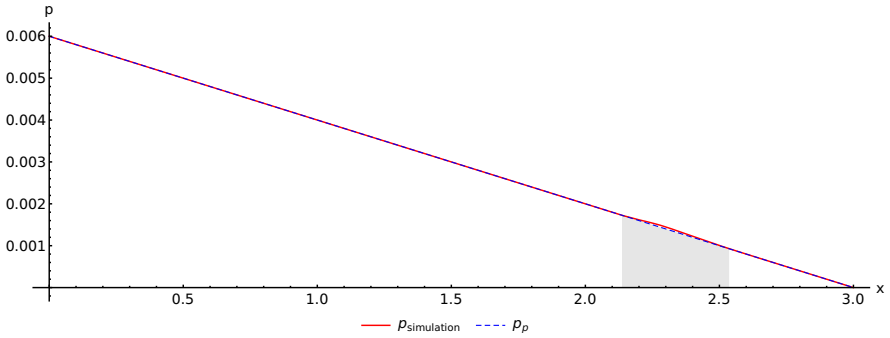
In the last numerical example we illustrate the stability and performance properties of the proposed CutFEM for a problem of higher interest in practice. Precisely, we simulate dynamic flow around a moving (two-dimensional) ball. The background domain is $\Omega = (0, 3) \times (0, 1)$, and we let $I = (0, 29]$. At the left inflow boundary we prescribe the parabolic inflow profile

$$\boldsymbol{g}_i(\boldsymbol{x}, t) = \begin{cases} (6ty(1-y), 0)^\top, & t \leq 1|, \\ (6y(1-y), 0)^\top, & t > 1\,. \end{cases} \tag{3.67}$$

For the rigid body, a ball with a radius $r = 0.2$ and an initial position of its center at $\boldsymbol{x}_r(0) = (1.545, 0.6)^\top$ is chosen. For a non-moving ball, the flow setting would result in a Reynolds number of $Re = 400$. The motion of the ball's center is prescribed by Eq. (3.62), with $A = 0.8$ and $\omega = 0.5$. On the boundary $\Gamma_r^t$ of the ball, the no-slip condition $\boldsymbol{v} = \boldsymbol{v}_r = (A\omega\cos(\omega t), 0)^\top$ for a moving, rigid domain is applied, such that



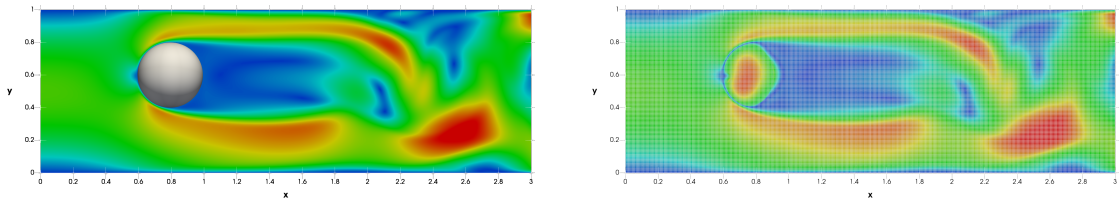

(a) Velocity along $y$-direction at $x = 2.34$.




(b) Pressure along $x$-direction at $y = 0$.

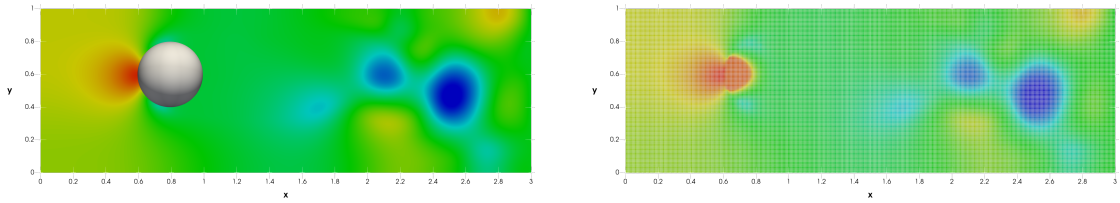
Figure 3.21: Cross section plots of velocity in $y$-direction and pressure in $x$-direction and Poiseuille profile at $T = 40$ with gray-shaded rigid body domain.

the fluid and ball velocity coincide on their interface, cf. [97, p. 47]. We use the time step size $\tau = 0.01$ and a uniform structured background mesh with $h = \frac{1}{64\sqrt{2}}$, which results in 889 862 degrees of freedom in each time interval. With this setup the rigid interface crosses at max one spatial mesh cell. Discrete solutions are computed in $(\mathbb{P}_1(I_n; H_h^2))^2 \times \mathbb{P}_1(I_n; H_h^1)$. Fig. 3.22 and Fig. 3.23 illustrate the computed profiles of velocity and the pressure at time $t = 10.12$ and $t = T$, respectively. The figures show that by the application of the ghost penalty stabilization and extension of the discrete solution to the ghost rigid domain, defined by Eq. (3.17), lead to stable approximations of the flow problem on a background mesh. The color legend of Fig. 3.23 shows that reasonable values for the velocity and pressure variable are obtained in the ghost subdomain of the rigid ball. We note that all kind of cuts (cf. Sec. 3.4.1), including irregular ones with small portions, arise in the simulation. Unphysical oscillations that are due to irregular cuts of finite elements or insufficient extensions to the rigid domain, are strongly reduced and do not perturb the ambient fluid flow.



(a) Velocity field with overlay of moving rigid domain.

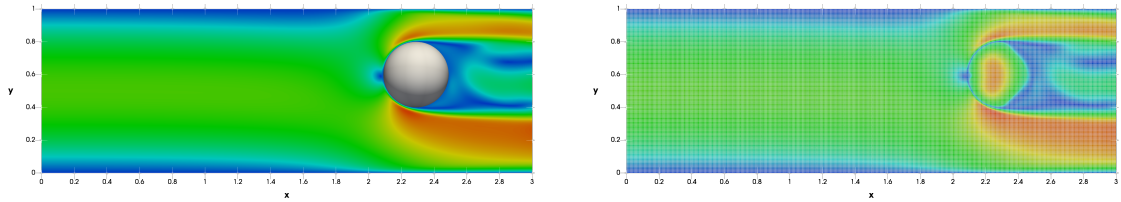
(b) Velocity field with computational background mesh.



(c) Pressure field with overlay of moving rigid body.

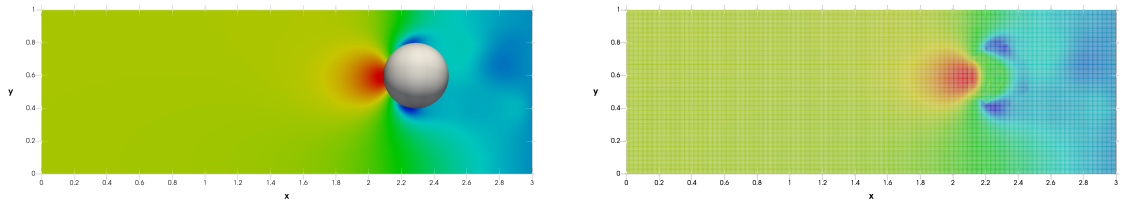
(d) Pressure field with computational background mesh.

Figure 3.22: Solution of the CutFEM simulation at time $t = 10.12$.



(a) Velocity field with overlay of moving rigid body.

(b) Velocity field with computational background mesh.



(c) Pressure field with overlay of rigid body.

(d) Pressure field with computational background mesh.


(e) Scaling for the velocity and pressure plots.

Figure 3.23: Solution of the CutFEM simulation at $t = T$.

## 3.6 Summary

In this chapter a CutFEM approach for the numerical simulation of fluid flow on evolving domain was presented. The physical domain was embedded into a fixed com-

putational background mesh. Discontinuous Galerkin methods and inf-sup stable pairs of finite element spaces were used for the discretization of the time and space variables, respectively. Ghost penalty stabilization for the treatment of irregular cuts along with an extension of the discrete solution to the ghost rigid domain was suggested further. The numerical performance properties of the parallel implementation of the family of schemes were illustrated. Thereby, the accuracy and stability of the approach was confirmed. Here, a parallel direct linear solver was applied for the sake of simplicity. In Sec. 4.6 results of embedding the CutFEM approach of this chapter into a geometric multigrid environment are presented and challenges for future research are identified and outlined.

# 4 A geometric multigrid preconditioner for space–time flow discretizations

Often, the linear solver represents the limiting factor for the level of mesh resolution and its number of degrees of freedom. In Chapter 3 the usage of a direct solver is one of the main drawbacks, that prohibits the application of the method to more sophisticated problems. Here, a geometric multigrid (GMG) method that is used as a preconditioner for generalized minimal residual (GMRES) iterations is proposed and analyzed computationally for higher order space–time finite element discretizations of the Navier–Stokes equations in two and three space dimensions. Its parallel implementation in the *deal.II* library [57] is also addressed.

As pointed out in Sec. 1.3, discretizing the Navier–Stokes system by inf-sup stable pairs of finite elements leads to linear systems of equations with saddle point structure. Higher order variational time discretizations, that we implement as time marching schemes by the choice of a discontinuous in time test basis, generate linear block systems, where one can again identify sub-blocks, which have again saddle point structure. As it can be seen in Eq. (3.48), such systems consist of several temporal degrees of freedom in time within each time step.

In this work we propose a GMG approach based on a local (cell-based) Vanka smoother for higher order discontinuous Galerkin approximations in time. We apply this GMG preconditioner to all the space–time degrees of freedom in each time step. An inf-sup stable pair of finite element spaces with discontinuous pressure approximation is used for the discretization in space. This GMG method is built in the state of the art, multi-purpose finite element toolbox deal.II (cf. [57] and [98], [99]) along with the linear algebra package Trilinos [94]. Efficient data structures are provided. The deal.II library is enhanced in such a way that a parallel assembly and application of a cell-based Vanka smoother become feasible. We note that some geometric multigrid support, that was used in [100] to implement a GMG based preconditioner using $H^{\mathrm{div}}$-conforming elements for the Stokes problem, was already provided in deal.II. The robustness and

efficiency of our GMG method in terms of a grid-independent convergence of the preconditioned GMRES iterations is studied computationally for the popular benchmark problems of flow around a cylinder in two and three space dimensions. For this, we explicitly note that parallel multigrid iterations for challenging three-dimensional flow problems do by far not meet a standard nowadays. This is underlined by the fact that the three-dimensional benchmark problem of flow around a cylinder [46] continues to be an open one. Confirmed numbers for the goal quantities of the simulation are not available yet.

This chapter is organized as follows. In Sec. 4.1, the prototype model problem is introduced. Afterwards, in Sec. 4.2, our space–time finite element approach for simulating the Navier–Stokes system, as well as the structure of the resulting underlying system matrix, are presented. In Sec. 4.3, we briefly recall the geometric multigrid algorithm as well as the local Vanka smoother, used in this work. We address practical aspects of the parallel implementation of the algorithm by using the deal.II library and the linear algebra package Trilinos. Then, in Sec. 4.4, we present numerical results and measure the performance properties of our proposed algorithms for the 2d and 3d DFG benchmark of flow around a cylinder [46].

## 4.1 Time-independent domains' Navier–Stokes system

At first we introduce a GMG preconditioner for problems that are defined on time-independent domains. Therefore, we briefly introduce the prototype setting of incompressible viscous flow around an obstacle in a rectangular two- or three-dimensional domain, that is time-independent. The two-dimensional problem configuration along with the notation of the geometrical setting is sketched in Fig. 4.1. We evaluate the performance properties of the proposed GMG solver for this benchmark problem (cf.[46]) that is interesting enough. Further, the notation overhead is reduced by the restriction to this setting. We consider solving the Navier–Stokes equations

$$\partial_t \boldsymbol{v} + (\boldsymbol{v}\cdot\nabla)\boldsymbol{v} - \nu\Delta\boldsymbol{v} + \nabla p = \boldsymbol{f} \quad \text{in } \Omega\times I\,, \tag{4.1a}$$

$$\nabla\cdot\boldsymbol{v} = 0 \quad \text{in } \Omega\times I\,, \tag{4.1b}$$

$$\boldsymbol{v} = \boldsymbol{g} \quad \text{on } \Gamma_D\times I\,, \tag{4.1c}$$

$$\nu\nabla\boldsymbol{v}\cdot\boldsymbol{n} - \boldsymbol{n}p = \boldsymbol{0} \quad \text{on } \Gamma_o\times I\,, \tag{4.1d}$$

$$\boldsymbol{v}(0) = \boldsymbol{v}_0 \quad \text{in } \Omega\,. \tag{4.1e}$$

In (4.1), $\Omega \subset \mathbb{R}^d$, with $d = 2$ or $d = 3$, is the open domain filled with fluid. We put $I = (0, T]$ for some final time $T > 0$. The velocity field $\boldsymbol{v}$ and the pressure $p$ are the unknown variables. In (4.1a), the parameter $\nu > 0$ denotes the fluid's viscosity and the right-hand side function $\boldsymbol{f}$ is a given external force. The union of the Dirichlet boundary segments is denoted by $\Gamma_D$, such that $\Gamma_D := \Gamma_i \cup \Gamma_w$. On $\Gamma_D$ we prescribe the fluid velocity by a function $\boldsymbol{g}$, that prescribes an inflow profile on $\Gamma_i$ and a no slip condition on $\Gamma_w$. $\Gamma_o$ represents an outflow boundary that is modeled by the do-nothing boundary condition (4.1d); cf. [78]. In (4.1d), the field $\boldsymbol{n}$ is the outer unit normal vector. In (4.1e), the function $\boldsymbol{v}_0$ denotes the prescribed initial velocity. In our numerical experiments presented in Sec. 4.4, the (time-independent) rigid domain $\Omega_r$ is a sphere in two space-dimensions or a cylinder in three space-dimensions.

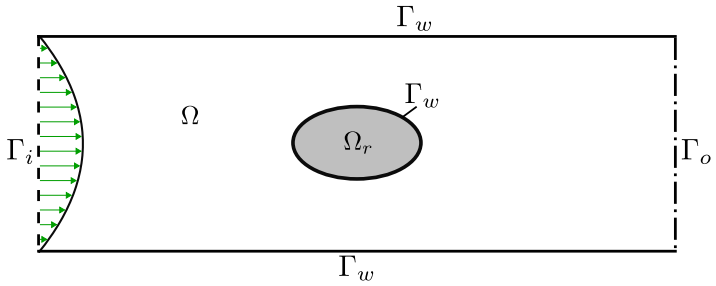



Figure 4.1: Problem setting for two space dimensions and corresponding notation.

For the weak problem formulation we introduce the semi-linear form $A : (\boldsymbol{H}^1(\Omega) \times L_0^2(\Omega)) \times (\boldsymbol{H}^1_{0,\Gamma_D}(\Omega) \times L_0^2(\Omega)) \to \mathbb{R}$ by

$$A((\boldsymbol{v}, p), (\boldsymbol{\psi}, \xi)) := \langle (\boldsymbol{v} \cdot \boldsymbol{\nabla})\boldsymbol{v}, \boldsymbol{\psi} \rangle + \nu \langle \nabla \boldsymbol{v}, \nabla \boldsymbol{\psi} \rangle - \langle p, \nabla \cdot \boldsymbol{\psi} \rangle + \langle \boldsymbol{\nabla} \cdot \boldsymbol{v}, \xi \rangle \,, \tag{4.2}$$

for $(\boldsymbol{v}, p) \in \boldsymbol{H}^1(\Omega) \times L_0^2(\Omega)$ and $(\boldsymbol{\psi}, \xi) \in \boldsymbol{H}^1_{0,\Gamma_D}(\Omega) \times L_0^2(\Omega)$. For given $\boldsymbol{f} \in \boldsymbol{L}^2(\Omega)$ we introduce the linear form $L : \boldsymbol{H}^1_{0,\Gamma_D}(\Omega) \to \mathbb{R}$ by

$$L(\boldsymbol{\psi}; \boldsymbol{f}) := \langle \boldsymbol{f}, \boldsymbol{\psi} \rangle \,, \tag{4.3}$$

for $\boldsymbol{\psi} \in \boldsymbol{H}^1_{0,\Gamma_D}(\Omega)$.

## 4.2 Space–time finite element discretization

Here we introduce our approximation of the Navier–Stokes system (4.1) by space-finite element methods. Firstly, we briefly recall the space–time variational formulation of the Navier–Stokes system (4.1) on fixed grids and present its discretization in space and

time, similar to Sec. 3.2. Finally, the block structure of the Newton linearized algebraic system for each time slab is illustrated. To solve the linear systems of the Newton iteration, we use a GMG preconditioner for the outer flexible GMRES iterations. Its presentation in Sec. 4.3 is the key contribution of this chapter.

### 4.2.1 Variational formulation of the continuous problem

A sufficiently regular solution of the Navier–Stokes system (4.1) satisfies the following variational problem that provides the basis for the space–time finite element discretizations.

**Problem 4.1** (Continuous variational space–time problem)**.** *Let* $\boldsymbol{f} \in L^2(I; \boldsymbol{H}^{-1}(\Omega))$ *and* $\boldsymbol{v}_0 \in \boldsymbol{L}^2_{div,0,\Gamma_D}(\Omega)$ *be given. Let* $\hat{\boldsymbol{g}} \in L^2\left(I; H^1(\Omega)\right) \cap L^\infty_I$ *denote a prolongation of the boundary values on* $\Gamma_D$ *to* $\Omega$*, such that* $\hat{\boldsymbol{g}} = \boldsymbol{g}$ *on* $\Gamma_D \times I$*. Find* $(\boldsymbol{v}, p) \in (V_I \cap L^\infty_I) \times L^2_{0,I}$*, with* $\boldsymbol{v} \in \hat{\boldsymbol{g}} + V_{0,I}$*, such that* $\boldsymbol{v}(0) = \boldsymbol{v}_0$ *and*

$$\int_0^T \langle \partial_t \boldsymbol{v}, \boldsymbol{\psi} \rangle + A((\boldsymbol{v}, p), (\boldsymbol{\psi}, \xi)) \mathrm{d}t = \int_0^T L(\boldsymbol{\psi}; \boldsymbol{f}) \mathrm{d}t \,, \tag{4.4}$$

*for all* $(\boldsymbol{\psi}, \xi) \in V_{0,I} \times L^2_{0,I}$*.*

**Remark 4.1.** *In the sequel, on the discrete level we only use a Nitsche-type incorporation of the Dirichlet boundary values, similiarly to Chapter 3.*

### 4.2.2 Semidiscretization in space with weak enforcement of boundary conditions by Nitsche's method

To incorporate Dirichlet boundary conditions we use Nitsche's method, as in Sec. 3.2.

Let $t \in [0, T]$. For the treatment of Dirichlet boundary conditions by Nitsche's method (cf. [61]) we introduce the bilinearform $B_{\Gamma_D} : \boldsymbol{H}^{1/2}(\Gamma_D) \times (\boldsymbol{V}_h \times Q_{h,\mathrm{disc}}) \to \mathbb{R}$ by

$$\begin{aligned} B_{\Gamma_D}(\boldsymbol{w}, (\boldsymbol{\psi}_h, \xi_h)) := &- \langle \boldsymbol{w}, \nu \nabla \boldsymbol{\psi}_h \cdot \boldsymbol{n} + \xi_h \boldsymbol{n} \rangle_{\Gamma_D} \\ &+ \gamma_1 \nu \langle h^{-1} \boldsymbol{w}, \boldsymbol{\psi}_h \rangle_{\Gamma_D} + \gamma_2 \langle h^{-1} \boldsymbol{w} \cdot \boldsymbol{n}, \boldsymbol{\psi}_h \cdot \boldsymbol{n} \rangle_{\Gamma_D} \,, \end{aligned} \tag{4.5}$$

for $\boldsymbol{w} \in \boldsymbol{H}^{1/2}(\Gamma_D)$ and $(\boldsymbol{\psi}_h, \xi_h) \in \boldsymbol{V}_h \times Q_{h,\mathrm{disc}}$, where $\gamma_1 > 0$ and $\gamma_2 > 0$ are numerical (tuning) parameters, see Sec. 3.2.3. The discrete semilinear form $A_h : (\boldsymbol{V}_h \times Q_{h,\mathrm{disc}}) \times (\boldsymbol{V}_h \times Q_{h,\mathrm{disc}}) \to \mathbb{R}$ is then given by

$$A_h((\boldsymbol{v}_h, p_h), (\boldsymbol{\psi}_h, \xi_h)) \coloneqq A((\boldsymbol{v}_h, p_h), (\boldsymbol{\psi}_h, \xi_h)) - \langle \nu \nabla \boldsymbol{v}_h \cdot \boldsymbol{n} - p_h \boldsymbol{n}, \boldsymbol{\psi}_h \rangle_{\Gamma_D} + B_{\Gamma_D}(\boldsymbol{v}_h, \boldsymbol{\phi}_h)\,, \tag{4.6}$$

for $(\boldsymbol{v}_h, p_h) \in \boldsymbol{V}_h \times Q_{h,\mathrm{disc}}$ and $(\boldsymbol{\psi}_h, \xi_h) \in \boldsymbol{V}_h \times Q_{h,\mathrm{disc}}$. The linear form $L_h : (\boldsymbol{V}_h \times Q_{h,\mathrm{disc}}) \to \mathbb{R}$ is defined by

$$L_h((\boldsymbol{\psi}_h, \xi_h); \boldsymbol{f}, \boldsymbol{g}) \coloneqq L(\boldsymbol{\psi}_h; \boldsymbol{f}) + B_{\Gamma_D}(\boldsymbol{g}, (\boldsymbol{\psi}_h, \xi_h))\,, \tag{4.7}$$

for $(\boldsymbol{\psi}_h, \xi_h) \in \boldsymbol{V}_h \times Q_{h,\mathrm{disc}}$.

We are now in a position to define a semidiscrete approximation of Problem 4.1.

**Problem 4.2.** *Let $\boldsymbol{f} \in L^2(I; \boldsymbol{H}^{-1}(\Omega))$ and $\boldsymbol{v}_{0,h} \in \boldsymbol{V}_h^{div}$ be given. Find $(\boldsymbol{v}_h, p_h) \in V_{I,h} \times L^2_{0,I,h}$, such that $\boldsymbol{v}_h(0) = \boldsymbol{v}_{0,h}$ and*

$$\int_0^T \langle \partial_t \boldsymbol{v}_h, \boldsymbol{\psi}_h \rangle + A_h((\boldsymbol{v}_h, p_h), (\boldsymbol{\psi}_h, \xi_h))\mathrm{d}t = \int_0^T L_h(\psi_h; \boldsymbol{f}, \boldsymbol{g})\mathrm{d}t\,, \tag{4.8}$$

*for all $(\boldsymbol{\psi}, \xi) \in L^2(I; \boldsymbol{V}_h) \times L^2_{0,I,h}$.*

### 4.2.3 Fully discrete problem

Applying the discontinuous Galerkin scheme of Sec. 3.2.4 to Problem 4.2 and using a discontinuous local test basis, yields the following time marching scheme (see also Sec. 3.2.4):

**Problem 4.3.** *Let $\boldsymbol{f} \in L^2(I; \boldsymbol{H}^{-1}(\Omega))$ and $\boldsymbol{v}_{0,h} \in \boldsymbol{V}_h^{div}$ be given. For $n = 1, \ldots, N$, and given $\boldsymbol{v}_{\tau,h|I_{n-1}} \in \mathbb{P}_k(I_{n-1}; \boldsymbol{V}_h)$ for $n > 1$ and $\boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1}^-) \coloneqq \boldsymbol{v}_{0,h}$ for $n = 1$, find $(\boldsymbol{v}_{\tau,h}, p_{\tau,h}) \in \mathbb{P}_k(I_n; \boldsymbol{V}_h) \times \mathbb{P}_k(I_n; Q_h)$, such that*

$$\begin{aligned}\int_{t_{n-1}}^{t_n} \langle \partial_t \boldsymbol{v}_{\tau,h}, \boldsymbol{\psi}_{\tau,h} \rangle + A_h((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))\mathrm{d}t + \langle \boldsymbol{v}_{\tau,h}(t_{n-1}^+), \boldsymbol{\psi}_{\tau,h}(t_{n-1}^+) \rangle \\ = \int_{t_{n-1}}^{t_n} L_h(\psi_{\tau,h}; \boldsymbol{f}, \boldsymbol{g})\mathrm{d}t + \langle \boldsymbol{v}_{\tau,h}(t_{n-1}^-), \boldsymbol{\psi}_{\tau,h}(t_{n-1}^+) \rangle\,,\end{aligned} \tag{4.9}$$

*for all $(\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}) \in \mathbb{P}_k(I_n; \boldsymbol{V}_h) \times \mathbb{P}_k(I_n; Q_h)$.*

For the construction of the GMG method in Sec. 4.3.2 we discuss the algebraic counterpart of Eq. (4.9) more thoroughly. Firstly, we represent the unknown discrete functions $(\boldsymbol{v}_{\tau,h}, p_{\tau,h}) \in \mathbb{P}_k(I_n; \boldsymbol{V}_h) \times \mathbb{P}_k(I_n; Q_h)$ in a temporal basis $\{\chi_l\}_{l=0}^k$ of $\mathbb{P}_k(I_n; \mathbb{R})$ by means of

$$v_{\tau,h,i|I_n}(\boldsymbol{x}, t) = \sum_{l=0}^{k} v_i^{n,l}(\boldsymbol{x})\chi_{n,l}(t)\,, \quad \text{for } i \in \{1, \dots, d\}\,, \quad p_{\tau,h|I_n}(\boldsymbol{x}, t) = \sum_{l=0}^{k} p^{n,l}(\boldsymbol{x})\chi_{n,l}(t)\,, \tag{4.10}$$

with coefficient functions $\boldsymbol{v}^{n,l} = (v_1^{n,l}, \dots, v_d^{n,l})^\top \in \boldsymbol{V}_h$ and $p^{l,n} \in Q_h$, where $\boldsymbol{v}_{\tau,h} = (\boldsymbol{v}_{\tau,h,1}, \dots, \boldsymbol{v}_{\tau,h,d})^\top$ for $t \in I_n$. For the basis $\{\chi_l\}_{l=0}^k$ we choose the Lagrange interpolants with respect to the $k+1$ Gauss–Radau quadrature nodes of $I_n$, with the same motivation as in Sec. 3.3. Letting

$$H_h^r = \operatorname{span}\{\psi_1, \dots, \psi_R\} \quad \text{and} \quad Q_h = \operatorname{span}\{\xi_1, \dots, \xi_S\}\,, \tag{4.11}$$

the coefficient functions $\boldsymbol{v}^{n,l} \in \boldsymbol{V}_h$ and $p^{n,l} \in Q_{h,\mathrm{disc}}$ of (4.10) admit the representation

$$v_i^{n,l}(\boldsymbol{x}) = \sum_{r=1}^{R} v_{i,r}^{n,l}\, \psi_r(\boldsymbol{x}) \quad \text{and} \quad p^{n,l}(\boldsymbol{x}) = \sum_{s=1}^{S} p_s^{n,l}\, \xi_s(\boldsymbol{x})\,,$$

with the vectors of unknown coefficients

$$\boldsymbol{v}_i^{n,l} = (v_{i,1}^{n,l}, \dots, v_{i,R}^{n,l})^\top \in \mathbb{R}^R \quad \text{and} \quad \boldsymbol{p}^{n,l} = (p_1^{n,l}, \dots, p_S^{n,l})^\top \in \mathbb{R}^S\,,$$

for all degrees of freedom in time in $I_n$ with $l = 0, \dots, k$, see Sec. 3.3. The coupling of the coefficient functions, defined by (4.10), with the test basis, given by (4.11), in the algebraic formulation of Eq. (4.9) is illustrated in Table 4.1.

Applying now a Newton linearization as in Sec. 3.3 to Eq. (4.9) results on an algebraic level in the same linearized system as in Eq. (3.45).
The block structure of the Jacobian matrix $\boldsymbol{J}_n^m$ can be deduced from the couplings illustrated in Table 4.1. For brevity, an explicit form of the Jacobian matrix $\boldsymbol{J}_n^m$ is not given here. For the sake of clearness, we restrict ourselves to presenting the block

Table 4.1: Coupling of the coefficient functions of (4.10) with the spacial test basis of (4.11) in the fully discrete space–time system (4.9) on $I_n$ for two space dimensions with $d = 2$.

| | $(\psi_1, 0)^\top$ | $(0, \psi_1)^\top$ | $\cdots$ | $(\psi_R, 0)^\top$ | $(0, \psi_R)^\top$ | $\xi_1$ | $\cdots$ | $\xi_S$ |
|---|---|---|---|---|---|---|---|---|
| $v_1^{n,0}$ | • | • | | • | • | • | | • |
| $v_2^{n,0}$ | • | • | | • | • | • | | • |
| $p^{n,0}$ | • | • | | • | • | | | |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $v_1^{n,k}$ | • | • | | • | • | • | | • |
| $v_2^{n,k}$ | • | • | | • | • | • | | • |
| $p^{n,k}$ | • | • | | • | • | | | |

structure of $\boldsymbol{J}_n^m$ for the polynomial order in time $k = 1$ only. This corresponds to the dG(1) method for the time discretization. For this, we get that

$$\boldsymbol{J}_n^m = \begin{pmatrix} \boldsymbol{F}_1 & \boldsymbol{B}_1^\top & \boldsymbol{F}_2 & \boldsymbol{B}_2^\top \\ -\boldsymbol{B}_1 & \boldsymbol{0} & -\boldsymbol{B}_2 & \boldsymbol{0} \\ \boldsymbol{F}_3 & \boldsymbol{B}_3^\top & \boldsymbol{F}_4 & \boldsymbol{B}_4^\top \\ -\boldsymbol{B}_3 & \boldsymbol{0} & -\boldsymbol{B}_4 & \boldsymbol{0} \end{pmatrix}. \tag{4.12}$$

In (4.12), the partitioning of the vector of unknowns $\boldsymbol{D}_n^m$ of the corresponding system Eq. (3.45) is then given by

$$\boldsymbol{D}_n^m = (\boldsymbol{d_v}^{n,0}, \boldsymbol{d_p}^{n,0}, \boldsymbol{d_v}^{n,1}, \boldsymbol{d_p}^{n,1})^\top \in \mathbb{R}^{2(d \cdot R + S)},$$

where $\boldsymbol{d_v}^{n,l}$ and $\boldsymbol{d_p}^{n,l}$, with $l \in \{0, 1\}$, denote the components of $\boldsymbol{D}_n^m$ related to velocity and pressure unknowns, respectively. In contrast to the matrix of (3.48), we have no $\boldsymbol{C}$ blocks here, that arise of the ghost penalty terms in Eq. (3.45).

To enhance the range of convergence of Newton's method, we also apply the linesearch technique of Sec. 3.3.2.1. To solve the linear systems of the Newton iteration, we use a flexible GMRES Krylov subspace method [39] with a GMG preconditioner based on a local Vanka smoother [40]. The GMG approach is presented in the next section.

## 4.3 A parallel geometric multigrid preconditioner

During the last decades numerous methods for solving the algebraic linear systems resulting from the discretization of the Navier–Stokes equations have been developed and studied. GMG methods seem to be among the best classes of solvers that are currently available; cf., e.g., [47]. Space–time finite element methods have recently attracted researchers' interest strongly. Their application puts an additional complexity to the solution of the linear systems due to their more complex block structure; cf. Table 4.1 and, e.g., [1], [101]. Here, we use the GMG method as a preconditioner for Krylov subspace iterations, which is a standard concept for the efficient solution of high-dimensional linear systems arising from the discretization of partial differential equations. The core of GMG methods is the smoother. We propose a cell-based Vanka smoother that is adapted to the space–time finite element approach.

Even though the basic concepts of GMG methods have become standard in the meantime, their efficient implementation continues to be a challenging task. In particular, this holds if the computational power of modern parallel computer architectures has to be fully exploited. In this case, the definition of data structures and the memory usage become of utmost importance. Moreover, trends like adaptive space–time finite element methods (cf. [102]) further complicate their implementation. These issues induce an ongoing research about GMG methods; cf., e.g., [98]. For our simulations we use the *deal.II* finite element toolbox [57]. Details of our implementation of the GMG approach in this platform are addressed in the sequel as well. The concepts are flexible enough and can be transferred to similar software tools.

### 4.3.1 Key idea of the geometric multigrid method

To sketch briefly the basic principles of GMG iterations and fix our notation, we consider the linear system Eq. (3.45), that is rewritten in the simpler, index-free form

$$\boldsymbol{J}\boldsymbol{d} = \boldsymbol{q}\,, \tag{4.13}$$

with a the right-hand side vector $\boldsymbol{r}$ and the Jacobian matrix $\boldsymbol{J}$. The key idea of the GMG method, that is sketched in Fig. 4.2, is to construct a hierarchical sequence of finite element spaces $V_h^l$, with $l = 1, \ldots, L$, that are embedded into each other, such that $V_h^1 \subset V_h^2 \subset \ldots \subset V_h^L$, and correspond to different grid levels $\mathcal{T}_{h_l}$ with mesh sizes $h_l$, for $l = 1, \ldots, L$ , of decompositions of the domain $\Omega$. Instead of solving the linear system

(4.13) on the finest grid level $\mathcal{T}_{h_L}$ entirely, the idea is to smooth only high frequency errors of an initial guess to the solution $\boldsymbol{d} = \boldsymbol{d}_L$ of (4.13) on the finest grid level $\mathcal{T}_{h_l}$ with $l = L$. Clearly, on level $l = L$, the right-hand side vector $\boldsymbol{r} = \boldsymbol{r}_L$ corresponds to the right-hand side vector $\boldsymbol{Q}_n^k$ of the Newton system Eq. (3.45). Now, smoothing is done by the application of the local Vanka operator $\boldsymbol{S}_L$. Then, the resulting residual $\boldsymbol{r}_L$ of (4.13) for the computed approximation of $\boldsymbol{d}_L$ is restricted to next coarser mesh level $\mathcal{T}_{h_l}$, with $l = L - 1$, which yields the right-hand side vector $\boldsymbol{r} = \boldsymbol{r}_{L-1}$. On level $L - 1$, the high frequency errors of an initial guess (given by the null vector $\mathbf{0}$) to the solution's correction $\boldsymbol{d}_{L-1}$ on $\mathcal{T}_{h_l}$, with $l = L - 1$, is smoothened by the application of the local Vanka operator again. These operations of restricting the residual to the next coarser grid and smoothing on this level the error in the solution of the defect equation is recursively repeated until the coarsest mesh level $\mathcal{T}_{h_l}$, with $l = 1$, is reached. On this level, typically a direct solver is used to compute the corresponding defect correction $\boldsymbol{d}_1$. Afterwards the computed defect correction of the coarsest level is prolongated to the next finer grid level $\mathcal{T}_{h_l}$, with $l = 2$, and used to update the defect correction $\boldsymbol{d}_2$. On this level, the defect correction $\boldsymbol{d}_2$ is then smoothed again and, finally, prolongated to the next coarser mesh level $\mathcal{T}_{h_l}$, with $l = 3$. These operations of prolongating successively the coarse grid correction and smoothing the modified defect correction are continued until the finest grid level $\mathcal{T}_{h_l}$, with $l = L$, is reached, where after the final smoothing an updated solution is obtained. This GMG approach is summarized in Algorithm 2.

**Algorithm 2:** Recursive algorithm Multigrid$(\boldsymbol{d}_l, \boldsymbol{r}_l, l)$

**if** $l = 0$ **then**
  $\boldsymbol{d}_1 = \boldsymbol{J}_1^{-1}\boldsymbol{r}_1$ ; `// Direct coarse solver`
**end**
**else**
  $\boldsymbol{d}_l = \boldsymbol{S}(\boldsymbol{d}_l, \boldsymbol{r}_l)$ ; `// Pre-smooth`
  $\boldsymbol{r}_l \coloneqq \boldsymbol{J}\boldsymbol{d}_l - \boldsymbol{r}_l$ ; `// Compute residuum`
  $\boldsymbol{r}_{l-1} = \boldsymbol{R}(\boldsymbol{r}_l)$ ; `// Restrict residuum`
  Multigrid$(\mathbf{0}, \boldsymbol{r}_{l-1}, l-1)$ ; `// Recursively call this function`
  $\boldsymbol{c}_l = \boldsymbol{P}(\boldsymbol{d}_l)$ ; `// Prolongate correction`
  $\boldsymbol{d}_l \coloneqq \boldsymbol{d}_l + \boldsymbol{c}_l$ ; `// Correct solution`
  $\boldsymbol{d}_l = \boldsymbol{S}(\boldsymbol{d}_l, \boldsymbol{r}_l)$ ; `// Post-smooth`
**end**

For our implementation of the GMG approach and the simulations presented in Sec. 4.4, we use the *deal.II* finite element toolbox [57] along with the direct, parallel

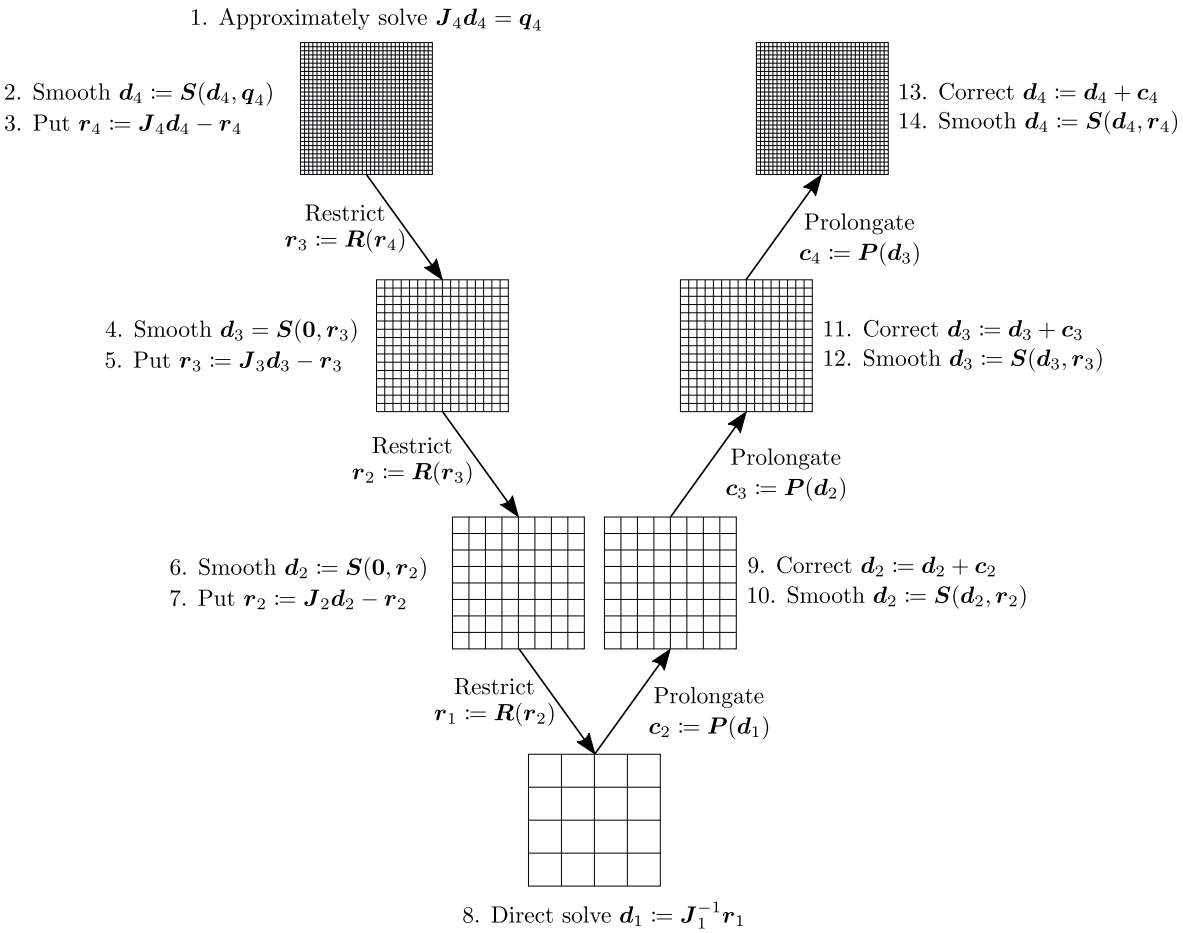



Figure 4.2: Structure of a single multigrid V-cycle for a hierarchy of four grid levels.

*SuperLU_Dist* solver [89]. Our code is based on the contributions of [98] to this open source framework and expands their work by a parallel, cell-based Vanka smoother. For the restriction and prolongation steps in parallel computations, the deal.II classes *MultiGrid* and *MGTransferPrebuilt* are used. The latter implements the prolongation between grids by interpolation and applies the transpose operator for the restriction. The core of our GMG approach is the smoother. This operator has to be efficient in smoothing high frequency errors. Further, since the smoother is applied frequently (cf. Fig. 4.2), this demands for its performant and scalable implementation in the multi processor such that the hardware's potential is fully exploited. Our implementation of the smoother is presented more in detail in the sequel.

### 4.3.2 A parallel, cell-based Vanka smoother

The Newton linearized system (4.13) of the fully discrete problem (4.9) has a generalized saddle-point structure; cf. Eq. (4.12). The generalization comes through the

application of the higher order discontinuous Galerkin time discretization with $k+1$ temporal degrees of freedom, cf. Eq. (4.10) in time for the velocity and pressure variable within each subinterval $I_n$. Thereby, blocks of saddle point subsystems arise; cf. Eq. (4.12). Standard smoothers, like the Gauss-Seidel or Jacobi method, that are often used in GMG methods, are not applicable to such systems; cf. [103]. Vanka smoothers, that can be traced back to [104], offer the potential to to damp high frequency errors in the approximation of solutions to saddle point problems. In [47], [105], Vanka-type smoothers have demonstrated excellent performance properties for systems with weak velocity-pressure couplings. In this work, we adapt the principle of a cell-based, full Vanka smoother of [47, p. 460] and extend the definition of the Vanka smoother to our higher-order space–time finite element approximation of the Navier–Stokes system. Since the numerical results, reported for instance in [49], show that a strong velocity-pressure coupling leads to a local violation of the continuity constraint, (4.1b) we only use a discontinuous finite element space for the pressure variable, defined in Chapter 2. This results in the ability to use local test functions, that are defined on a single cell. Therefore, the mass conservation is fulfilled locally, cf. [97], [106]. Fig. 4.3 illustrates the position of the underlying degrees of freedom for a pair of continuous/discontinuous finite elements for the velocity/pressure variables, corresponding to the case $r = 2$ in the definitions of (2.8).

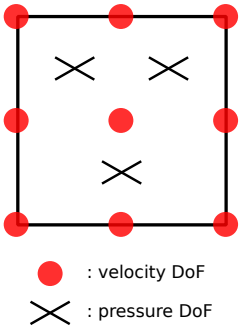



Figure 4.3: Degrees of freedom for the continuous/discontinuous $\mathbb{Q}_2$–$\mathbb{P}_1$ pair of elements, see [107].

**Remark 4.2.** *In the deal.II finite element library, we use the* FE_DGP⟨ ⟩ *class to form a basis of* $\mathbb{P}_r$. *This basis is constructed using a set of polynomials of complete degree* $r$ *that form a Legendre basis on the unit square, i. e. they are* $L^2$ *orthogonal and normalized on the reference cell. Noteworthy, this element is not a Lagrangian one, so it is not defined by finding shape functions within the given function space that interpolate a particular set of points. Therefore, in Fig. 4.3 the pressure DoF positions*

*symbolically just stand for the number of basis functions on each element and not for the corresponding position of nodal interpolation points.*

Now, we define the cell-based Vanka smoother for the proposed space–time finite element approximation. This part generalizes previous work on the local Vanka smoother to the higher order time discretization. On the mesh level $l$ (cf. Fig. 4.2) and in a single iteration step, the cell-based Vanka smoother is applied to all $I_n$-coefficient subvectors of the spatial degrees of freedom corresponding to the respective mesh cell $T$. On grid level $l$, with $l \in \{2, \dots, L\}$, we let the solution vector $\boldsymbol{d}$ of (4.13) be subdivided in terms of velocity and pressure subvectors according to the structure of the solution vector $\boldsymbol{X}_n$, defined in (3.43), of the nonlinear system (3.44), such that

$$\boldsymbol{d} = (\boldsymbol{d}_1^0, \dots, \boldsymbol{d}_d^0, \boldsymbol{q}^0, \dots, \boldsymbol{d}_1^k, \dots, \boldsymbol{d}_d^k, \boldsymbol{q}^k)^\top \in \mathbb{R}^{(k+1)\times(d\cdot R_l + S_l)} . \tag{4.14}$$

Here, $R_l$ and $S_l$ denote the number of (global) degrees of freedom for the velocity and pressure variable on grid level $l$, where $\boldsymbol{X}_n$ is defined for the finest mesh level $\mathcal{T}_L$. The subvectors $\boldsymbol{d}_1^l, \dots, \boldsymbol{d}_d^l$ for $l = 0, \dots, k$, correspond to the velocity values (or their corrections, respectively) and the subvectors $\boldsymbol{q}^l$ for $l = 0, \dots, k$, to the pressure values (or their corrections, respectively) on the grid level $l$. With the amount $n_p$ of the local pressure degrees of freedom on each element, $n_p = \binom{d+r}{r}$, we then denote by $\boldsymbol{d}_T$ the subvector of $\boldsymbol{d}$ that is built from the degrees of freedom in $\boldsymbol{d}$ that are associated with the element $T$, such that

$$\boldsymbol{d}_T = (\boldsymbol{d}_{1,T}^0, \dots, \boldsymbol{d}_{d,T}^0, \boldsymbol{q}_T^0, \dots, \boldsymbol{d}_{1,T}^k, \dots, \boldsymbol{d}_{d,T}^k, \boldsymbol{q}_T^k)^\top \in \mathbb{R}^{(k+1)\times(d\cdot(r+1)^d + n_p)} . \tag{4.15}$$

Here, the subvectors $\boldsymbol{d}_{1,T}^l, \dots, \boldsymbol{d}_{d,T}^l$ for $l = 0, \dots, k$, correspond to the velocity values on the element $T$ and the subvectors $\boldsymbol{q}_T^l$ for $l = 0, \dots, k$, to the pressure values. Further, for right-hand side vector $\boldsymbol{r}$ of (4.13) on grid level $l$ we let

$$\boldsymbol{r}_0 = (\boldsymbol{r}_1^0, \dots, \boldsymbol{r}_d^0, \boldsymbol{r}_p^0, \dots, \boldsymbol{r}_1^k, \dots, \boldsymbol{r}_d^k, \boldsymbol{r}_p^k)^\top \in \mathbb{R}^{(k+1)\times(d\cdot R_l + S_l)} , \tag{4.16}$$

with the partition of $\boldsymbol{r}$ into subvectors induced by (4.14). On a single cell $T$, the local Jacobian matrix $\boldsymbol{J}_T$ is defined as

$$\boldsymbol{J}_T = \frac{\partial \boldsymbol{Q}_T}{\partial \boldsymbol{X}_T}(\boldsymbol{X}_T) ,$$

where, in contrast to (3.46), we skipped the index of the time interval $n$ and the Newton step $m$ for brevity. On such a cell $T$, the smoothing operator $\boldsymbol{S}_T(\boldsymbol{d}, \boldsymbol{r})$ is then defined by

$$\boldsymbol{S}_T(\boldsymbol{d}, \boldsymbol{r}) := \boldsymbol{d}_T + \boldsymbol{J}_T^{-1}(\boldsymbol{r}_0 - \boldsymbol{J}\boldsymbol{d})_T \,. \tag{4.17}$$

In (4.17), the vector $(\boldsymbol{r}_0 - \boldsymbol{J}\boldsymbol{d})_T \in \mathbb{R}^{(k+1)\times(d\cdot(r+1)^d+n_p)}$ denotes the local subvector of $(\boldsymbol{r}_0 - \boldsymbol{J}\boldsymbol{d}) \in \mathbb{R}^{(k+1)\times(d\cdot R_l+S_l)}$, that is obtained by condensing $(\boldsymbol{r}_0 - \boldsymbol{J}\boldsymbol{d})$ to its components corresponding to the mesh cell $T$, similarly to (4.15). The full application of the smoother $\boldsymbol{S}(\boldsymbol{d}, \boldsymbol{r})$ then comes through running over all cells of the corresponding mesh level and applying the local smoother $\boldsymbol{S}_T(\boldsymbol{d}, \boldsymbol{r})$ to each of the elements.

The appreciable advantage of the Vanka smoother is that the system, that has to be solved on each cell, or the inverse of the local Jacobian matrix $\boldsymbol{J}_T^{-1}$, respectively, is small compared to the global system with system matrix $\boldsymbol{J}$. This will be addressed further in the next subsection. The efficiency of the application of the Vanka smoother in complex simulations with a high number of mesh cells depends on two ingredients:

1. The efficient application of $\boldsymbol{J}_T^{-1}$: How are the local systems defined by (4.17) solved?

2. The efficient data exchange in the parallel environment: How are the data for computing $\boldsymbol{J}_T$ or $\boldsymbol{J}_T^{-1}$, respectively, assembled?

These two issues are discussed in the following.

### 4.3.3 Efficient application of $\boldsymbol{J}_T^{-1}$

The implementation of the operator $\boldsymbol{J}_T^{-1}$ is an important ingredient for the efficiency of the GMG approach in computations, since the Vanka smoother is applied . We recall that the GMG method is used as a preconditioner in GMRES iterations for solving the Newton linearized system of each subinterval $I_n$. We also refer to Fig. 4.2 illustrating the usage of smoothing steps on the grid levels of a GMG V-cycle. In our implementation of the GMG method, inverses of the element-wise Jacobian matrices $\boldsymbol{J}_T^{-1}$, for all $T \in \mathcal{T}_{h_l}$ with $l = 1, \ldots, L$, are pre-computed after each update of the Jacobian matrix $\boldsymbol{J}$. For this, we use LAPACK routines to pre-compute the matrices $\boldsymbol{J}_T^{-1}$ and store them in a hashed *unordered_map*. If $\boldsymbol{J}_T^{-1}$ has to be applied on a cell $T$ according to (4.17), the costs for looking up the corresponding inverse is an operation with a complexity of order $\mathcal{O}(1)$. The costs in terms of memory for storing each inverse $\boldsymbol{J}_T^{-1}$ is, for instance, $85 \cdot 85 \cdot 64\,\text{bit} = 57.8\,\text{kB}$ or $0.0578\,\text{MB}$ for a three-dimensional problem

for the spatial approximation by the $\mathbb{Q}_2$–$\mathbb{P}_1^{\text{disc}}$ pair of finite elements (corresponding to $r = 2$ in (4.10)) and the $dG(0)$ time discretization (corresponding to $k = 0$ in Eq. (4.10)) on a 64-bit machine, plus some (negligible) additional overhead to store, for instance, the hashes. For a $dG(1)$ time discretization (corresponding to $k = 1$ in Eq. (4.10)), the local Jacobian $\boldsymbol{J}_T$ is a $60 \times 60$ matrix and the needed amount of memory is $170 \cdot 170 \cdot 64\,\text{bit} = 231.2\,\text{kB}$ or $0.231\,\text{MB}$. Since the code is parallelized, every process has to store only the information, data and inverses of the cells that it owns. Therefore, the additional amount of memory, that is needed in each process, can be decreased by increasing the number of involved processors.

### 4.3.4 Efficient data exchange in parallel environments

For pre-computing the inverses of the local Jacobian matrices $\boldsymbol{J}_T^{-1}$, the entries $\boldsymbol{J}_{i,j}$ of $\boldsymbol{J}_T$ in the element $T$ have to be computed. If the code is executed in parallel by multiple processes, the data access problem that is sketched in Fig. 4.4 occurs. When the local matrix $\boldsymbol{J}_{T_1}$ on $T_1$ is assembled by the process 1, all the needed matrix entries of the global Jacobian matrix $\boldsymbol{J}$ are available, and can be copied to the local Jacobian $\boldsymbol{J}_{T_1}$, since process 1 owns all of the involved degrees of freedom. In contrast, process 2 doesn't own the degrees of freedom on the face separating $T_1$ and $T_2$, since in a parallel environment every process has only read access to the entries it owns. For computing $\boldsymbol{J}_{T_2}$, the entries in the global matrix $\boldsymbol{J}$ of process 1, corresponding to the degrees of freedom on the common interface, are required.

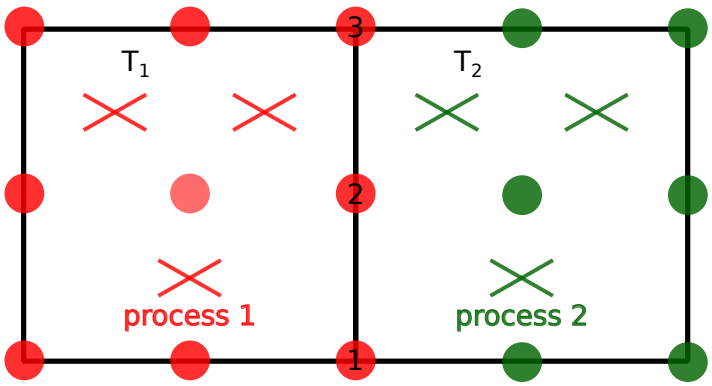



Figure 4.4: Two cells that belong to two different processes with the DoFs for $\mathbb{Q}_2$–$\mathbb{P}_1^{\text{disc}}$ elements.

To provide and exchange the needed data efficiently, the following data structure, called *map_proc_row_column_value*, is generated on each of the involved processes. It involves the following, nested containers (from top to bottom).

- *dealii::MGLevelObject⟨ ⟩*: The top level object, that contains the next elements for each mesh level $l$. The *MGLevelObject⟨ ⟩* is basically a container like *std::vector⟨ ⟩*, but with the option to shift indexing. So if the coarsest mesh starts e.g. on level $l = 2$ one can access the elements inside this object with the [ ] operator and an index, starting from 2 onward.

- *std::map⟨unsigned int, std::unordered_map⟨...⟩ ⟩*: A map, whose keys are the process numbers (an unsigned int) of the neighboring processes, that own certain degrees of freedom. These are exactly those degrees of freedom, that the owning process needs to access during the assembly of the local Jacobian $\boldsymbol{J}_T$. The process numbers are obtained by Algorithm 3. The value of the map is a (hashed) unordered_map, which leads to the next container inside the structure:

- *std::unordered_map⟨ std::pair⟨unsigned int, unsigned int⟩, double⟩*: For each neighboring process a hashed, unordered_map is stored, that contains the global row and column number (both unsigned int) of the needed matrix entries and assigns them to the corresponding value, which is stored as a double. Internally the std::pair of global row and column numbers is stored as hashed value. Since the C++ standard library doesn't include a default hashing algorithm for std::pair, this is done with the operation shown in Listing 1. In line 5 a std::hash object is created and the ( ) operator is called with the first unsigned int variable of the std::pair. The result is then stored in a variable called "hash" and in line 6 this hash is combined with the second unsigned integer of the std::pair using a boost operation. Basically each time an inverse is inserted or accessed into the std::unordered_map, this hashing algorithm is executed and access operations have an average complexity of $\mathcal{O}(1)$.

After generation of the mesh hierarchy, every process executes the Algorithm 3.

```
1 struct hash_pair {
2         template <class T1, class T2>
3         size_t operator()(const std::pair<T1, T2> &p) const
4         {
5                 auto hash = std::hash<T1>{}(p.first);
6                 boost::hash_combine(hash, std::hash<T2>{}(p.second));
7                 return hash;
8         }
9 };
```

Listing 1: Hashing algorithm, suitable for a std:pair⟨unsigned int, unsigned int⟩.

**Algorithm 3:** Determine values of $\boldsymbol{J}$ owned by foreign processes

```
Create map_proc_row_column_value ;
foreach mesh level l do
    foreach locally owned cell K_i do
        foreach DoF j do
            if DoF j is not owned by this process then
                Get the number of the process np that owns it ;
                Add np and all couplings of DoF j with all other locally owned
                 DoFs to map_proc_row_column_value and assign it a value of 0. ;
            end
        end
    end
end
```

**Remark 4.3.** *If the underlying mesh or the distribution of the degrees of freedom to the involved processes is fixed, which is especially the case if no remeshing is necessary between time-steps, then Algorithm 3 needs to be executed only once in the simulation. For instance, this holds in the numerical examples of Sec. 4.4. In the simulation of flow problems on evolving domains by CutFEM approaches, that are currently focused strongly (cf. [26], [27]), this applies similarly and results in negligible computational costs.*

Afterwards the simulation is continued until all contributions of the global system matrix $\boldsymbol{J}$ on all levels $l$ are assembled. For building the Vanka smoother by assembling and storing the local matrices $\boldsymbol{J}_T^{-1}$, the respective sparse matrix entries have to be exchanged such that every process can access the entries of each local Jacobian matrix $\boldsymbol{J}_T$. This is done by Algorithm 4.

**Algorithm 4:** Update *map_proc_row_column_value* on each process

```
foreach mesh level l do
    Use MPI some-to-some communication to transfer information:
     recv_proc_row_column_value ← map_proc_row_column_value ;
    foreach proc i in recv_map_proc_row_column_value do
        foreach global row k and column l in recv_proc_row_column_value do
            value ← J_{k,l}
        end
    end
    Use MPI some-to-some communication to transfer back information:
     map_proc_row_column_value ← recv_proc_row_column_value ;
end
```

**Remark 4.4.** *Algorithm 4 has to be executed whenever the global Jacobian* $\boldsymbol{J}$ *in* (4.13) *is updated. To block as less resources as necessary, just the processes that need to exchange information, communicate with each other. The exchange of data is implemented in the object* map_proc_row_column_value*. The data transfer is done entirely in one single step (see Algorithm 4), instead of querying the data, that is needed for a single cell* $T$ *from foreign processes, in the assembly routine. This exchange of information is then cached in the memory by the object* map_proc_row_column_value*. By this approach, we reduce the communication between processes to an absolute necessary minimum.*

**Remark 4.5.** *To call MPI functions that transfer data between processes, these data have to be serializable. The C++17 version of* std::unordered_map *included in the standard library is by default not serializable. In the code of this work, the boost C++ library is used that provides serialization capabilities for* std::unordered_map *via the headers* serialization.hpp *and* unordered_map.hpp*.*

## 4.4 Numerical examples

In the following we analyze computationally the performance properties of the proposed GMG approach. This is down for the well-known benchmark problems of low around a cylinder; cf. [46]. Our computations were done on a Linux cluster with 96 nodes, each of them with 2 CPUs and 14 cores per CPU. The CPUs are *Intel Xeon E5-2680 v4* with a base frequency of 2.4 GHz, a maximum turbo frequency of 3.3 GHz and a

level 3 cache of 35 MB. Each node has 252 GB of main memory. In this work, scaling experiments on up to the user limit of 32 nodes were performed.

### 4.4.1 Flow around a cylinder in two space dimensions

In the first numerical experiment we consider the well-known 2d DFG benchmark setting of flow around a cylinder, with the same setup and setting as in Sec. 3.5.4.

The final simulation time in the benchmark is $T = 10$, such that $I = (0, 10]$. The space–time discretization is done in the spaces $(\mathbb{P}_1(I_n; H_h^2))^2 \times \mathbb{P}_1(I_n; H_{h,\text{disc}}^1)$ with time step size $\tau = 0.005$. The Newton iteration is stopped when the residual of the nonlinear equation is smaller than $1 \times 10^{-10}$ or a relative reduction of the initial residual by a factor of 10 000 is reached. The computations were done on 4 nodes of the Linux cluster. The computed velocity field of the fully developed flow is presented in Fig. 3.17.

Table 4.2 shows the space–time degrees of freedom in one single time step and summarizes for three mesh levels the computed maximum drag and lift coefficients of the fully developed flow as well as the average number of Newton iterations per time step and the average number of GMRES iterations per Newton step. Table 4.2 shows that on all mesh levels an average of less than two Newton iterations per subinterval is obtained. Moreover, the number of GMRES iterations with GMG preconditioning remains (almost) grid-independent. This demonstrates the efficiency of the proposed approach.

Table 4.2: Computed drag and lift coefficients and average number of Newton steps per time step and of GMRES iterations per Newton step in the two-dimensional benchmark. Reference values: $c_{D_{max}} \in [3.2200, 3.2400]$ and $c_{L_{max}} \in [0.9900, 1.0100]$, see [46].

| Nr. | DoFs | $h_{\max}$ | $c_{D_{max}}$ | $c_{L_{max}}$ | $\bar{n}_{\text{Newton}}$ | $\bar{n}_{\text{GMRES}}$ | $\bar{n}_{\text{GMRES}}$ |
|---|---|---|---|---|---|---|---|
| 1 | 8 305 664 | 0.0022 | 3.2350 | 1.0062 | 1.72 | 10 | 7 |
| 2 | 2 080 256 | 0.0048 | 3.2227 | 1.0060 | 1.67 | 9 | 7 |
| 3 | 521 984 | 0.0090 | 3.1274 | 0.9637 | 1.53 | 9 | 6 |
| | | | Multigrid cycle | | | $V(1,1)$ | $V(4,4)$ |

Table 4.3 summarizes the wall time consumed by the different parts of the algorithms. $t_{\text{GMRES}}$ summarizes the time, spent in the outer flexible GMRES solver (including the GMG preconditioner). $t_{\text{Inv}}$ is the time spent for computing the local inverses of the cell Jacobians $J_T^{-1}$ and $t_{\text{Upd}}$ is the time that is spent in Algorithm 4, exchanging necessary

information between processes. In our simulations, the same number of compute nodes (4 nodes) were used on all grid levels. The usage of 4 nodes for the considered problem dimensions leads to a great difference in the percentage wall time of the GMRES solver. On the finest level, the GMG preconditioned GMRES iterations consumed only 17.96 % of the total wall time. The latter indicates that the benefit of a faster system assembly, by using more nodes, would have paid off and annihilated the costs of an increased parallel communication. In contrast, on the coarsest level the number of nodes was set too high to pay off. We observe just a slight decrease in the wall time despite nearly quartering the number of degrees of freedom.

Table 4.3: Wall time consumption of the two-dimensional benchmark simulation.

(a) Utilizing a $V(1,1)$ multigrid cycle.

| DoFs | $t_{\text{wall}}$ | $t_{\text{GMRES}}$ | % of $t_{wall}$ | $t_{\text{Inv}}$ | % of $t_{wall}$ | $t_{\text{Upd}}$ | % of $t_{wall}$ |
|---|---|---|---|---|---|---|---|
| 8 305 664 | 6.14 h | 1.10 h | 17.96 | 0.15 h | 2.42 | 0.04 h | 0.66 |
| 2 080 256 | 1.10 h | 0.57 h | 51.81 | 0.04 h | 2.16 | 0.02 h | 1.11 |
| 521 984 | 0.83 h | 0.40 h | 48.16 | 0.01 h | 1.51 | 0.01 h | 1.40 |

(b) Utilizing a $V(4,4)$ multigrid cycle.

| DoFs | $t_{\text{wall}}$ | $t_{\text{GMRES}}$ | % of $t_{wall}$ | $t_{\text{Inv}}$ | % of $t_{wall}$ | $t_{\text{Upd}}$ | % of $t_{wall}$ |
|---|---|---|---|---|---|---|---|
| 8 305 664 | 5.76 h | 0.72 h | 12.44 | 0.15 h | 2.61 | 0.04 h | 0.69 |
| 2 080 256 | 0.88 h | 0.35 h | 40.07 | 0.04 h | 4.52 | 0.02 h | 2.26 |
| 521 984 | 0.70 h | 0.27 h | 38.53 | 0.01 h | 1.43 | 0.01 h | 1.42 |

### 4.4.2 Flow around a cylinder in three space dimensions

In this subsection the proposed GMG approach is applied to simulate flow around a cylinder in three space dimension; cf. [46], [50]. We note that this benchmark continues to be a challenging test problem for flow solvers. So far, the benchmark is still an open one since guaranteed numbers for the drag and lift coefficients are not available yet. The geometrical setting of the benchmark is shown in Fig. 4.5.

The goal quantities are again the drag and lift coefficients. In three space dimensions the drag and lift forces are defined as:

$$F_D = \int_S \left( \nu \frac{\partial \boldsymbol{v}_t}{\partial \boldsymbol{n}} n_y - P n_x \right) \mathrm{d}S \,, \qquad F_L = -\int_S \left( \nu \frac{\partial \boldsymbol{v}_t}{\partial \boldsymbol{n}} n_x - P n_y \right) \mathrm{d}S \,, \tag{4.18}$$

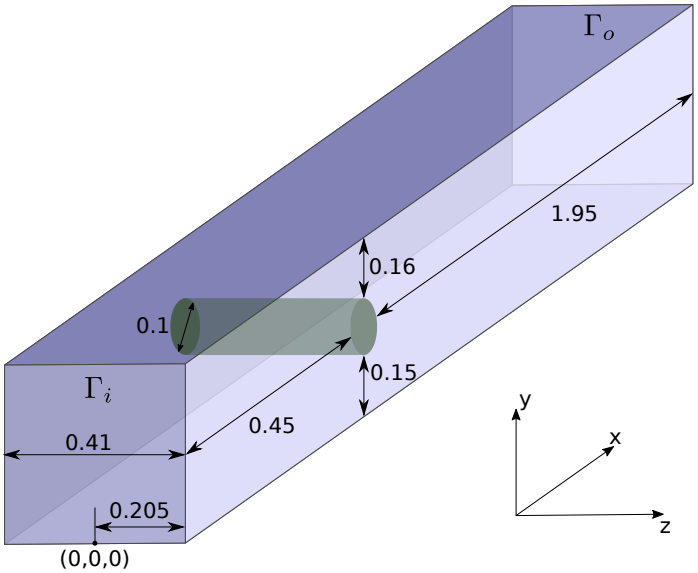



Figure 4.5: Geometrical setting of the 3d benchmark problem.

where $\boldsymbol{n}$ is the normal vector on $S$, pointing outside $S$. The tangential direction is defined as $\boldsymbol{t} = (n_y, -n_x, 0)^\top$ and $\boldsymbol{v}_t$ is the velocity $\boldsymbol{v}$ projected into the direction of $t$. The drag and lift coefficients are then given by

$$c_D = \frac{2}{\bar{U}^2 DH} F_D \,, \qquad\qquad c_L = \frac{2}{\bar{U}^2 DH} F_L \,, \tag{4.19}$$

where the diameter of the cylinder is $D = 0.1$ and the height of the pipe is $H = 0.41$ (cf. Fig. 4.5). On the inflow boundary $\Gamma_i$ the fluid velocity $\boldsymbol{v} = (v_x, v_y, v_y)^\top$ with

$$v_x(\boldsymbol{x}) = \frac{-16 \cdot U_m \cdot y \cdot \left(z - \frac{H}{2}\right) \cdot (H - y) \cdot \left(z + \frac{H}{2}\right)}{H^4} \,, \quad v_y(\boldsymbol{x}) = 0 \,, \quad v_z(\boldsymbol{x}) = 0 \,, \tag{4.20}$$

and $U_m = 2.25$ is prescribed. By the characteristic velocity of the flow of $\overline{U} = 1$ and a viscosity of $\nu = 0.001$ we compute the Reynold's number of the flow to

$$Re = \frac{\overline{U} \cdot D}{\nu} = 100 \,. \tag{4.21}$$

The final simulation time is put to $T = 8$ such that $I = (0, 8]$.

The numerical approximation is done in the space–time finite element spaces $(\mathbb{P}_1(I_n; H^2_h)^3 \times \mathbb{P}_1(I_n; H^1_{h,\text{disc}})$. Thus, the discontinuous Galerkin approximation in time with piecewise linear polynomials is used. On the finest grid level, this results in 96 876 736 spatial degrees of freedom in each time interval $I_n$ , i.e., over all degrees of freedom in time of $I_n$. The time interval $I = (0, T]$ is divided into 1598 slices of

different length, due to the benchmark configuration. The simulations were done on 32 nodes of the Linux cluster. To each CPU core an own process was assigned. Thus, the simulations were run by $32 \cdot 2 \cdot 14 = 896$ processes. On the finest level the mesh consisted of 1 703 936 cells such that each process accessed $1901 \pm 1$ cells of the mesh. The amount of memory of each process to store all the cell inverses $\boldsymbol{J}_K^{-1}$ on the finest level therefore was $231.2\,\text{kB} \cdot 1902 \approx 439.7\,\text{MB}$. For current high performance computing systems this represents a very reasonable or even small amount of memory usage. Fig. 4.6 visualizes the computed velocity field in the longitudinally cut domain at the final simulation time $T = 8$.

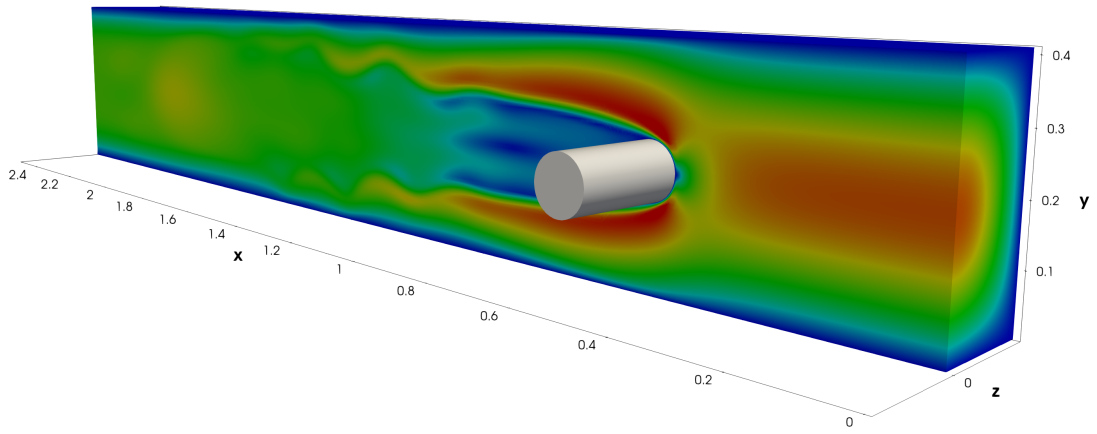
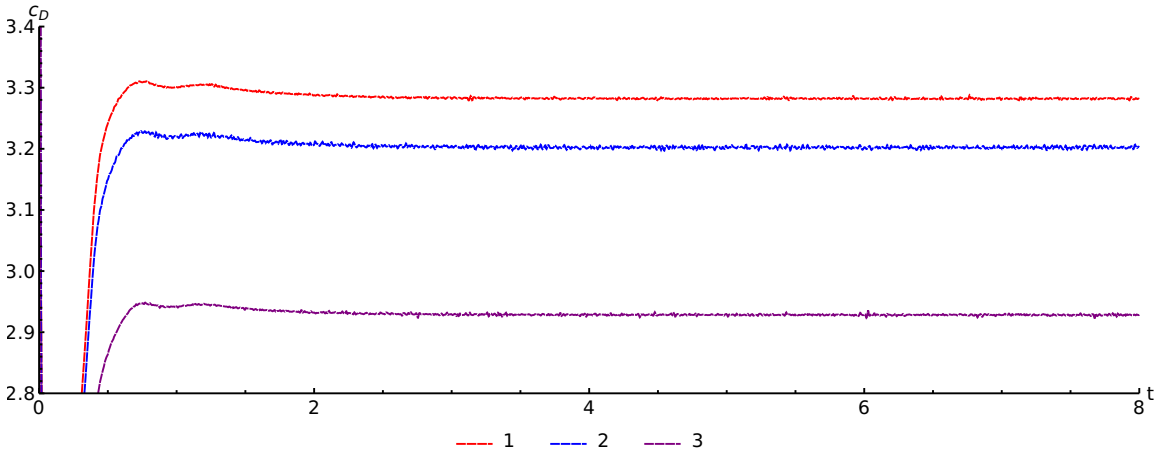



Figure 4.6: Flow profile in the longitudinally cut domain (at $z = 0$) of the benchmark problem with $Re = 100$.

Table 4.4: Computed drag and lift coefficients and average number of Newton steps per time step and of GMRES iterations per Newton step in the three-dimensional benchmark. Reference values: $c_{D_{max}} \in [3.2900, 3.3100]$ and $c_{L_{max}} \in [-0.0110, -0.0080]$, see [46]. $h_{\max}$ is the maximum diameter of the cells on the corresponding mesh level.

| Nr. | DoFs | $h_{\max}$ | $c_{D_{max}}$ | $c_{L_{max}}$ | $\bar{n}_{\text{Newton}}$ | $\bar{n}_{\text{GMRES}}$ | $\bar{n}_{\text{GMRES}}$ |
|---|---|---|---|---|---|---|---|
| 1 | 96 876 736 | 0.0110 | 3.2877 | −0.0072 | 1.47 | 87 | 26 |
| 2 | 12 293 216 | 0.0220 | 3.2070 | −0.0034 | 1.42 | 74 | 22 |
| 3 | 1 583 152 | 0.0440 | 2.9309 | −0.0031 | 1.44 | 72 | 24 |
| | | Multigrid cycle | | | | $V(1,1)$ | $V(4,4)$ |

Table 4.4 and fig. 4.7 present the computed drag and lift coefficients. Moreover, Table 4.4 summarizes the average number of Newton steps per subinterval and GMRES iterations per Newton step. The efficiency of the Newton iteration for solving the nonlinear problem is clearly demonstrated. The average number of Newton iterations is smaller than in the two-dimensional case; cf. Table 4.2. This might be due to fact




(a) Computed drag coefficients $c_D$.


(b) Computed lift coefficients $c_L$.

Figure 4.7: Computed drag and lift coefficients of the 3d benchmark on different mesh levels.

that the non-linear stopping criteria was weakened to a tolerance of $1 \times 10^{-6}$ instead of $1 \times 10^{-8}$ in the two-dimensional case, which is still smaller than the commonly used $1 \times 10^{-5}$, e .g. in [50]. Again, the GMG preconditioned GMRES solver shows an almost grid-independent convergence behavior. The average number of GMRES iterations per Newton steps is only increased very slightly by grid refinements. Thereby, the high efficiency of the proposed GMG preconditioning is demonstrated impressively.

Finally, Table 4.5 shows the wall-time consumption of the code for three mesh levels of successive refinement in space. In contrast to the two-dimensional case, most of the compute time is now spent on solving the Newton-linearized system. The main reason for this shift is probably the increased number of GMRES steps, compared to Table 4.2, that are performed until convergence of the GMRES method is reached.

Table 4.5: Wall time consumption of the three-dimensional benchmark simulation.

(a) Utilizing a $V(1,1)$ multigrid cycle.

| DoFs | $n_{\text{Nodes}}$ | $t_{\text{wall}}$ | $t_{\text{GMRES}}$ | % of $t_{wall}$ | $t_{\text{Inv}}$ | % of $t_{wall}$ | $t_{\text{Upd}}$ | % of $t_{wall}$ |
|---|---|---|---|---|---|---|---|---|
| 96 876 736 | 32 | 153.89 h | 92.22 h | 60.00 | 1.57 h | 1.00 | 0.51 h | 0.44 |
| 12 293 216 | 16 | 45.50 h | 27.90 h | 65.64 | 0.14 h | 0.34 | 0.07 h | 0.16 |
| 1 583 152 | 2 | 7.36 h | 5.22 h | 70.94 | 0.04 h | 0.74 | 0.04 h | 0.50 |

(b) Utilizing a $V(4,4)$ multigrid cycle.

| DoFs | $n_{\text{Nodes}}$ | $t_{\text{wall}}$ | $t_{\text{GMRES}}$ | % of $t_{wall}$ | $t_{\text{Inv}}$ | % of $t_{wall}$ | $t_{\text{Upd}}$ | % of $t_{wall}$ |
|---|---|---|---|---|---|---|---|---|
| 96 876 736 | 32 | 106.99 h | 45.32 h | 42.36 | 1.70 h | 1.59 | 0.47 h | 0.44 |
| 12 293 216 | 16 | 32.00 h | 14.40 h | 45.00 | 0.15 h | 0.46 | 0.08 h | 0.24 |
| 1 583 152 | 2 | 4.96 h | 2.82 h | 56.86 | 0.04 h | 0.77 | 0.04 h | 0.83 |

## 4.5 Parallel scaling

In this section we want to analyze the parallel performance properties of our code with a strong scaling benchmark. We first define the speedup $S$ of a program, accodring to [108]:

**Definition 4.1.** *Parallel speedup* *The speedup $S$ is defined as*

$$S = \frac{1}{r_s + \frac{r_p}{np}},$$

*where $r_s$ is the ratio of the sequential fraction of the program and $r_p$ the portion, that can be scheduled in parallel with np number of processes. This is called Amdahl's Law.*

So, if the problem size is fixed, the parallel speedup is limited by the serial fraction of code. Amdahl's law is under the assumption of an instant communication over a network, infinitely fast. In practice this is not possible and therefore one also has to consider the communication costs, that are introduced when increasing the number of processes np, due to the finite bandwidth and latency of the network. In practice, the speedup $S$ for a simulation, that is run multiple times on a different amount of nodes $n$ is calculated with

$$S = \frac{t_{\text{wall}}(n = n_{\text{min}})}{t_{\text{wall}}(n)}.$$

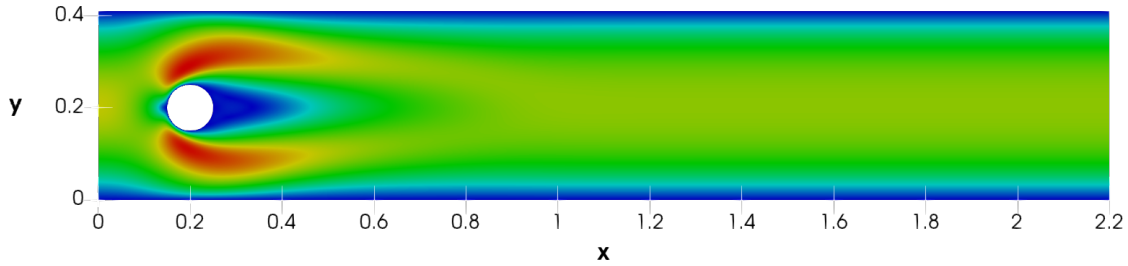
Here $n_{\text{min}}$ is the simulation with the smallest amount of nodes.

To measure the speedup S of our code, we use the spatial setup of the 2d DFG benchmark of Sec. 3.5.4, but set the inflow condition on $\Gamma_i$ as

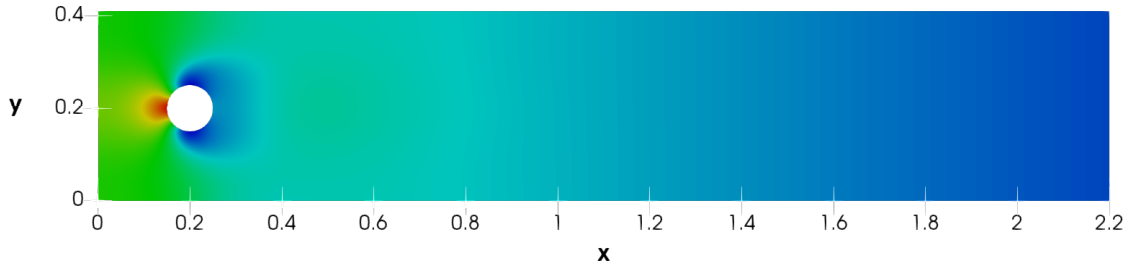
$$\boldsymbol{g}_i(x, y, t) = \left( \frac{4 \cdot 0.3 \cdot y(0.41 - y)}{0.41^2}, 0 \right)^\top .$$

With $\nu = 0.001$ This results in a Reynolds number of $Re = 20$. After about a simulation time of 2.3 s the flow is fully developed, which results in a static flow profile. The final simulation time is put to $T = 3$ such that $I = (0, 3]$ and the time step size is fixed to 0.1.

For the first benchmark, the numerical approximation is done in the space–time finite element spaces $(\mathbb{P}_1(I_n; H_h^2)^2 \times \mathbb{P}_1(I_n; H_{h,\text{disc}}^1)$. The mesh consists of 376 832 cells, which results in 8 305 664 space–time degrees of freedom in each time interval. Fig. 4.8 shows the velocity and the pressure field of the solution at $t = T$.




(a) Pressure field with overlay of rigid body.

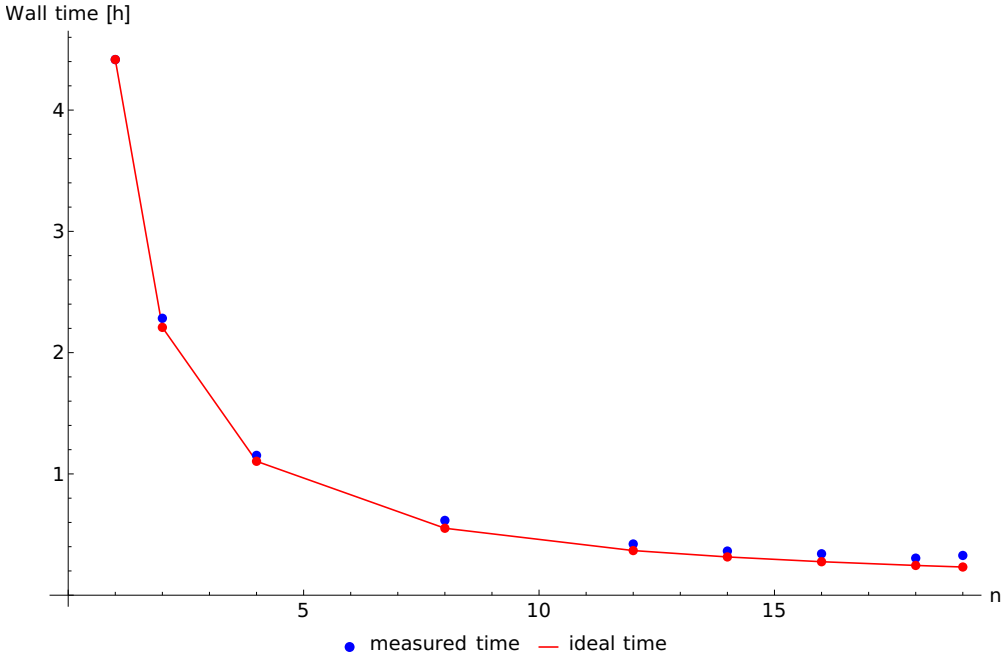



(b) Pressure field with computational background mesh.

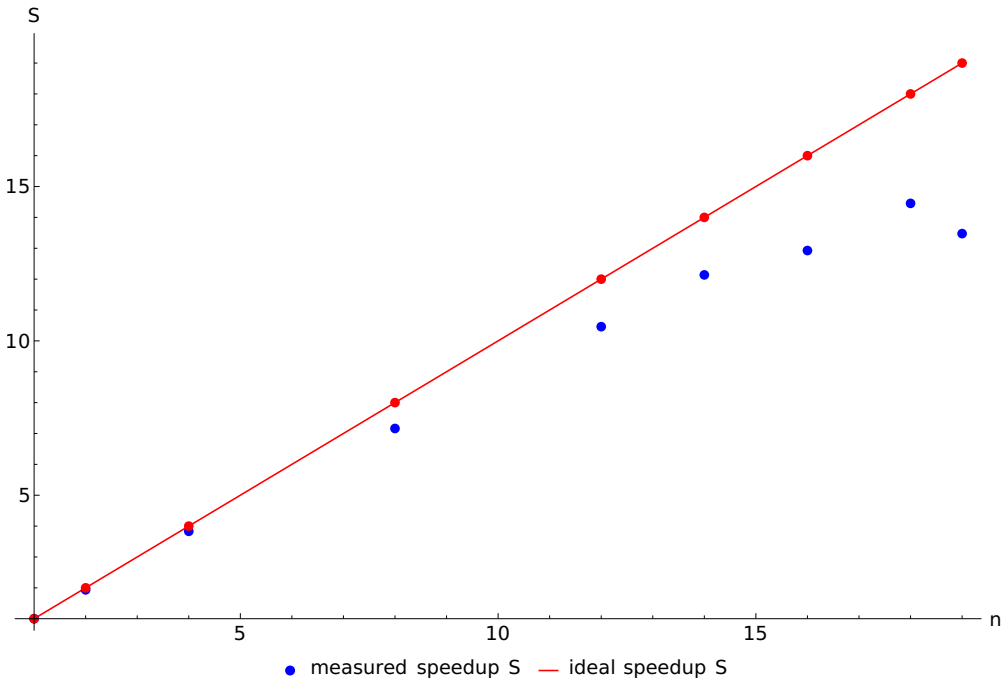

(c) Scaling for the velocity and pressure plots.

Figure 4.8: Solution of the CutFEM simulation at $t = T$.

The benchmark is performed on the same HPC as the example in Sec. 4.4.2, but the number of nodes $n$ is varied. We assign each node 28 processes, so that each physical CPU core owns a single process. Fig. 4.9 shows the results of the benchmarks. The ideal time is computed by setting $r_s$ in Definition 4.1 to zero. We see nearly ideal scaling




(a) Benchmarked and ideal wall time.




(b) Benchmarked and ideal speedup $S$.

Figure 4.9: Strong scaling results for the simulation in $(\mathbb{P}_1(I_n; H^2_h)^2 \times \mathbb{P}_1(I_n; H^1_{h,\text{disc}})$.

properties until the usage of 4 nodes. Afterwards, when we further increase the number of nodes, we still see a decrease of the overall wall time of the simulation until we reach peak performance with 18 nodes. When using even more nodes, the communication costs dominate and lead to an increase of the overall runtime of the simulation. In this

example the wall time could be reduced from 4.42 h to just 19.67 min, when using 18 nodes with 504 processes.

For the second benchmark, the numerical approximation is done in the space–time finite element spaces $(\mathbb{P}_2(I_n; H_h^3)^2 \times \mathbb{P}_2(I_n; H_{h,\text{disc}}^2)$. We use the same spatial mesh as before, with 376 832 cells. That results this time in 27 166 464 space–time degrees of freedom in each time interval. The minimum number of nodes, that were used for this benchmark was 4. With this configuration the runtime was 54.17 h. The maximum number of nodes, that we used, was 64. With this configuration the runtime was reduced to 3.39 h. Fig. 4.10 shows the results of the benchmarks. Until 16 nodes we have nearly optimal scaling results. Then one observes an increasing difference from the ideal speedup to the measured speedup, due to the increase of the communication costs. In Table 4.6 we summarized the characteristic statistics for the two simulations.

Table 4.6: Number of nodes $n$, number of corresponding processes $np$, average cells, that each process owns ($\overline{\text{cells}_\text{p}}$), wall time and speedup $S$ for the two strong scaling benchmarks.

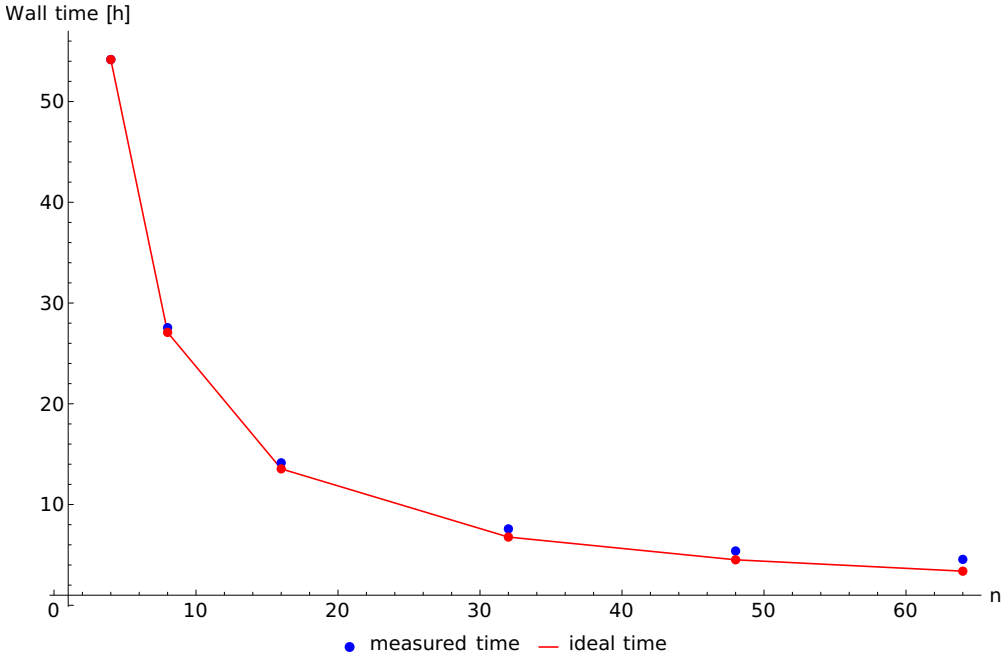
(a) Using $\mathbb{P}_1(I_n; H_h^2)^2 \times \mathbb{P}_1(I_n; H_{h,\text{disc}}^1)$ elements on each time interval.

| $n$ | $np$ | $\overline{\text{cells}_\text{p}}$ | Wall time [h] | Speedup $S$ |
|---|---|---|---|---|
| 1 | 28 | 13 460 | 4.42 | 1.00 |
| 2 | 56 | 6728 | 2.28 | 1.93 |
| 4 | 112 | 3364 | 1.15 | 3.83 |
| 8 | 224 | 1684 | 0.62 | 7.16 |
| 12 | 336 | 1120 | 0.42 | 10.46 |
| 14 | 392 | 960 | 0.36 | 12.14 |
| 16 | 448 | 840 | 0.34 | 16.93 |
| 18 | 504 | 748 | 0.31 | 14.45 |
| 19 | 532 | 708 | 0.32 | 13.47 |

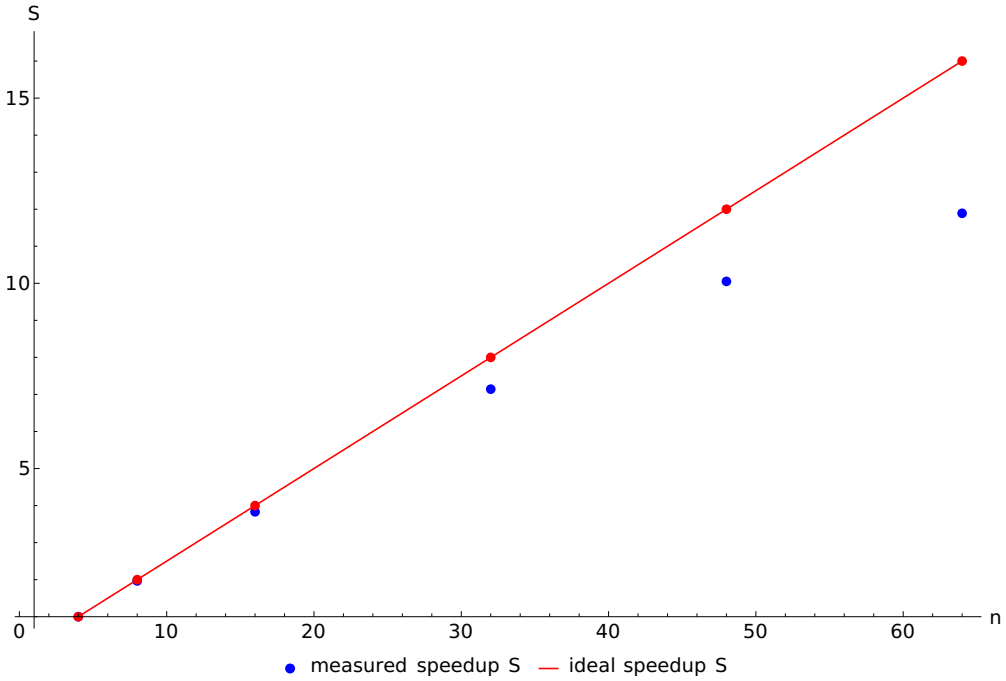
(b) Using $\mathbb{P}_2(I_n; H_h^3)^2 \times \mathbb{P}_2(I_n; H_{h,\text{disc}}^2)$ elements on each time interval.

| $n$ | $np$ | $\overline{\text{cells}_\text{p}}$ | Wall time [h] | Speedup $S$ |
|---|---|---|---|---|
| 4 | 112 | 3364 | 54.17 | 1.00 |
| 8 | 224 | 1684 | 27.08 | 1.97 |
| 16 | 448 | 840 | 13.54 | 3.83 |
| 32 | 896 | 420 | 6.77 | 7.14 |
| 48 | 1344 | 280 | 4.51 | 10.05 |
| 64 | 1792 | 212 | 3.39 | 11.89 |

Both benchmarks show, that the parallelization pays off. Especially higher order space–time elements are applicable in a reasonable amount of time only with parallelization.




(a) Benchmarked and ideal wall time.




(b) Benchmarked and ideal speedup $S$.

Figure 4.10: Strong scaling results for the simulation in $(\mathbb{P}_2(I_n; H_h^3)^2 \times \mathbb{P}_2(I_n; H_{h,\text{disc}}^2)$.

## 4.6 Evolving domains

In this section the idea of the GMG approach is used for solving the linear system of Eq. (3.48), that result from the CutFEM approach of Chapter 3. To achieve this,

the ideas of Sec. 4.3 are slightly modified, by adding the ghost penalty stabilization operator as a contribution to the underlying system, see Eq. (3.17) in Sec. 3.2.3. The resulting, fully discrete system, is then the one of Problem 3.3.

The validity of the assumptions, mentioned in Sec. 3.2.1, is now assumed to be fulfilled on all mesh levels. Especially the coarsest mesh level, which is called minimum level in this section, of the GMG approach has to be able resolve the boundary $\Gamma_r^t$ appropriately.

In the following subsections the performance of GMG combined with CutFEM techniques is computationally analyzed.

### 4.6.1 Laminar flow around a cylinder

In the first numerical example the same spatial setup as in Sec. 3.5.4 is used, but the viscosity is set to $\nu = 1$. The radius of the rigid body is varied. At first $r = r_1 = 0.05$, as in Sec. 3.5.4. Afterwards the radius is decreased to $r = r_2 = 0.025$, to see the effect of a smaller rigid domain. In both cases, the flow is fully developed after about 0.1 s and afterwards time independent. Therefore, the time domain is set to $I = (0, 0.1]$ and the time step size is fixed to 0.01. In the simulation the solution space $\mathbb{P}_0(I_n; H_h^2))^2 \times \mathbb{P}_0(I_n; H_h^1)$ is used. On the finest mesh level (8) this results in 3 610 626 degrees of freedom in each single time interval. Fig. 4.11 shows the mesh on level 3, which results in 3714 degrees of freedom and the corresponding position of the rigid domain. One can see here, that the assumptions of Sec. 3.2.1 are fulfilled on this coarsest mesh level. The stopping criteria in the newton solver is an absolute error, smaller than $5 \times 10^{-7}$. As in Sec. 3.5, the Nitsche penalty parameters of Eq. (3.13) are set to $\gamma_1 = \gamma_2 = 35$. [28], [30]. The ghost penalty parameters of Eq. (3.17) were set to $\gamma_v = \gamma_p = 10^{-2}$. Table 4.7 shows the average number of GMRES steps throughout the whole simulation, as well as the minimum and maximum number of such steps to achieve convergence in the linear solver. Each line of Table 4.7 shows one simulation with a different choice for the coarsest mesh level (coarse lvl) and the corresponding degrees of freedom on this level.

If the coarse mesh level is closer to level 8, on which the outer GMRES solver is applied, the number of iterations decrease. This is probably due to the better resolving of the interface on a finer coarse level or due to an impact of the ghost penalty extension.

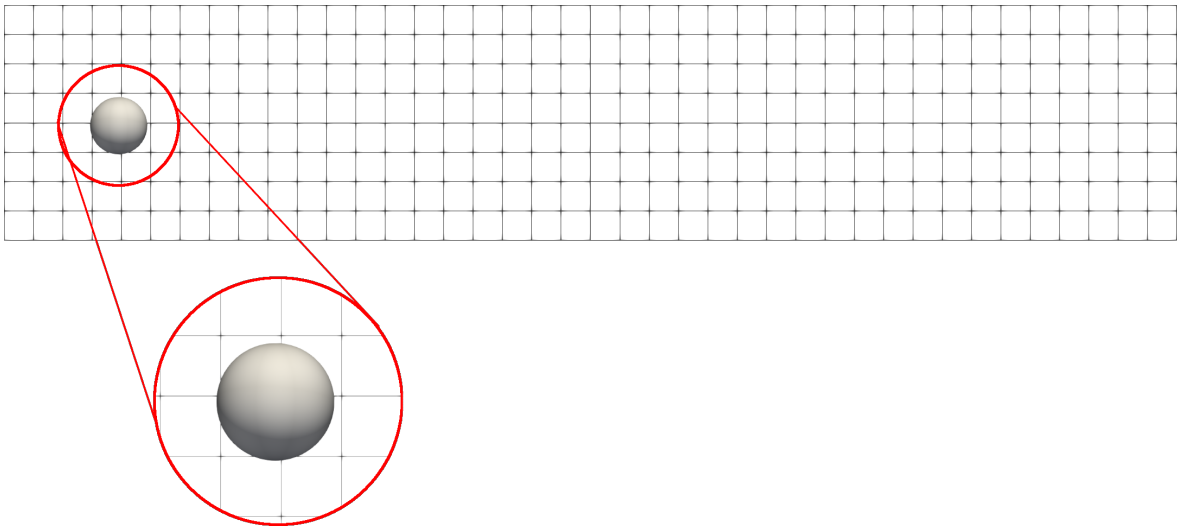


Figure 4.11: Computational mesh and position of the rigid domain $\Omega_r$ on mesh level 3.

Table 4.7: Statistics for the GMRES iterations per time step. The fine mesh corresponds to level 8 with 3 610 626 degrees of freedom.

| coarse lvl | coarse DoFs | $\bar{n}_{\text{GMRES}}$ | $n_{\min}$ | $n_{\max}$ | $\bar{n}_{\text{GMRES}}$ | $n_{\min}$ | $n_{\max}$ |
|---|---|---|---|---|---|---|---|
| 3 | 3714 | 399 | 19 | 820 | 379 | 12 | 794 |
| 4 | 14 466 | 370 | 19 | 747 | 362 | 12 | 743 |
| 5 | 57 090 | 284 | 14 | 533 | 235 | 11 | 484 |
| 6 | 226 818 | 164 | 11 | 256 | 124 | 10 | 193 |
| 7 | 904 194 | 51 | 8 | 91 | 38 | 8 | 83 |
| | | r = 0.05 | | | r = 0.025 | | |

Further, if the size of the rigid body is decreased, the number of iterations also decrease. This indicates, that the non-physical extension on different levels introduces problems to the GMG scheme. This is further studied in the next subsections.

### 4.6.2 Time periodic flow around a cylinder

In the next numerical example we use again the test setting of Sec. 4.6.1, but set $\nu = 0.001$. For $r_1 = 0.05$ this results in the same setting as in Sec. 3.5.4 and a Reynolds number of $Re = 100$.

The time domain is set to $I = (0, 10]$ and the time step size is fixed to 0.005. In the simulation the solution space $\mathbb{P}_0(I_n; H_h^2))^2 \times \mathbb{P}_0(I_n; H_h^1)$ is used. On the finest mesh level (8) this results in 3 610 626 degrees of freedom in each single time interval. The stopping criteria in the newton solver is an absolute error, smaller than $5 \times 10^{-7}$.

For $\gamma_v = \gamma_p = 10^{-2}$, the resulting GMG scheme results in a bad convergence behavior and did not converge within 3000 GMRES steps, except if the coarse level was set to 7, so the resulting scheme was just a two-grid scheme. For $r = r_1 = 0.05$ the average number of GMRES steps was then still 2284. Further decreasing the radius to $r = r_2 = 0.025$ lead to an average number of GMRES steps of 1636.

We explicitly note, that with this smaller radius of $r = r_2 = 0.025$ the Reynolds number of the flow is $Re = 50$, which results in a stationary flow after about 1.4 s of simulation time. Therefore, no new Newton step has to be performed after this start-up phase (and these time steps don't contribute to the amount of GMRES steps in Table 4.7).

These results indicate, that the artificial extension of the physical fluid quantities, $\boldsymbol{v}$ and $p$, from the fluid domain $\Omega_f$ to the rigid domain $\Omega_r$ has an impact on the convergence properties of the GMG scheme, which was also observed in Sec. 4.6.1. Despite the observations of Sec. 3.5, which indicated, that the solution on the fluid domain in such a CutFEM setting converges to the expected, physical solution, the extension has different properties on different mesh levels in the (unphysical) rigid domain $\Omega_r$. The properties depend on the size of the extension and the dynamics of the flow.

This effect can be influenced by adjusting the parameters $\gamma_v$ and $\gamma_p$ (see Sec. 3.2.3). By applying a manual bisection method (starting from 1), a tuning of these parameters was performed to find a "good choice" in the range $[1 \times 10^{-5}, 1]$ for $r_1$. The best parameter, that was found in this study, together with the statistics for the corresponding numbers of GMRES iterations are listed in Table 4.7. These parameters were used then afterwards for the simulation with $r_2$ (Table 4.8) and $r_3 = 0.01$ (Table 4.9). The results are summarized in Table 4.10.

Table 4.8: Applied ghost penalty parameters $\gamma_{v,p}$ and statistics for the GMRES iterations per time step for different radii $r$.

| coarse lvl | coarse DoFs | $\gamma_{v,p}$ | $\bar{n}_{\text{GMRES}}$ | $n_{\min}$ | $n_{\max}$ | $\bar{n}_{\text{GMRES}}$ | $n_{\min}$ | $n_{\max}$ |
|---|---|---|---|---|---|---|---|---|
| 3 | 3714 | 4e−3 | 857 | 476 | 1824 | 623 | 235 | 1067 |
| 4 | 14 466 | 2e−2 | 815 | 476 | 1646 | 578 | 218 | 1012 |
| 5 | 57 090 | 6e−4 | 697 | 359 | 1214 | 497 | 189 | 858 |
| 6 | 226 818 | 8e−4 | 573 | 328 | 915 | 356 | 168 | 781 |
| 7 | 904 194 | 7e−3 | 432 | 285 | 768 | 273 | 132 | 613 |
| | | | r = $r_1$ = 0.05 | | | r = $r_2$ = 0.025 | | |

Table 4.9: Applied ghost penalty parameters $\gamma_{v,p}$ and statistics for the GMRES iterations per time step, setting $r$ to $r_3 = 0.01$.

| coarse lvl | coarse DoFs | $\gamma_{v,p}$ | $\bar{n}_{\text{GMRES}}$ | $n_{\min}$ | $n_{\max}$ |
|---|---|---|---|---|---|
| 3 | 3714 | 4e−3 | 452 | 164 | 874 |
| 4 | 14 466 | 2e−2 | 391 | 146 | 747 |
| 5 | 57 090 | 6e−4 | 302 | 121 | 584 |
| 6 | 226 818 | 8e−4 | 195 | 96 | 395 |
| 7 | 904 194 | 7e−3 | 103 | 54 | 256 |

Table 4.10: Summarized comparison of the average number of GMRES iterations for varying radii of the rigid body. The fine mesh is set to level 8 with 3 610 626 degrees of freedom.

| coarse lvl | coarse DoFs | $\bar{n}_{\text{GMRES}}$ | $\bar{n}_{\text{GMRES}}$ | $\bar{n}_{\text{GMRES}}$ |
|---|---|---|---|---|
| 3 | 3714 | 857 | 623 | 452 |
| 4 | 14 466 | 815 | 578 | 391 |
| 5 | 57 090 | 697 | 497 | 302 |
| 6 | 226 818 | 573 | 356 | 195 |
| 7 | 904 194 | 432 | 273 | 103 |
| | | r = 0.05 | r = 0.025 | r = 0.01 |

### 4.6.3 Dynamic flow around a moving cylinder

In the last numerical example the GMG preconditioner is applied to a CutFEM problem of higher interest in practice. As in Sec. 3.5.6, the dynamic flow around a moving (two-dimensional) ball is simulated with $\nu = 0.001$. The background domain is $\Omega = (0, 3) \times (0, 1)$, and $I = (0, 12.6]$. At the left inflow boundary the following parabolic inflow profile is prescribed

$$\boldsymbol{g}_i(\boldsymbol{x}, t) = \begin{cases} (cty(1-y), 0)^\top , & t \leq 1|, \\ (cy(1-y), 0)^\top , & t > 1\,, \end{cases} \tag{4.22}$$

with a constant $c \in \mathbb{R}$. In contrast to Sec. 3.5.6, the radius of the rigid body is varied. In the first computational simulation it is set to $r = r_1 = 0.1$ and $c$ is set to $c = c_1 = 6$. For a non-moving rigid domain, the flow setting would result in a Reynolds number of $Re = 200$. In the second simulation the radius is further decreased to $r = r_2 = 0.05$. To keep the static Reynold's number at $Re = 200$, the constant $c$ is set to $c = c_2 = 12$.

The initial position of the rigid's body center is set to $\boldsymbol{x}_r(0) = (1.545, 0.6)^\top$. The motion of the ball's center is prescribed by Eq. (3.62), with $A = 0.8$ and $\omega = 0.5$.

On the boundary $\Gamma_r^t$ of the ball the condition $\boldsymbol{v} = \boldsymbol{v}_r = (A\omega\cos(\omega t), 0)^\top$ is applied, such that the fluid and ball velocity coincide on their interface, cf. [97, p. 47]. The time step size is fixed to $\tau = 0.005$ and a uniform structured background mesh with $h = \frac{1}{128\sqrt{2}}$, which results in 2 166 786 degrees of freedom in each time interval, is used. With this setup, the rigid interface doesn't cross more than one spatial mesh cell, i. e. with the maximum distance $s_h$, that the rigid body crosses in one time step, there holds $\frac{s_h}{h} = 0.512$. Discrete solutions are computed in $(\mathbb{P}_0(I_n; H_h^2))^2 \times \mathbb{P}_0(I_n; H_h^1)$. For the GMG scheme we use the ghost penalty parameters $\gamma_{v,p}$ of Sec. 4.6.2.

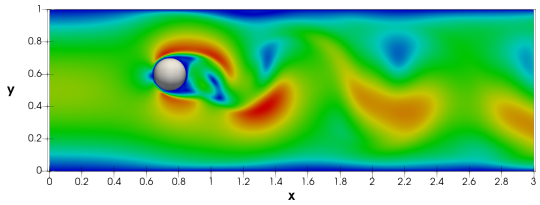
Table 4.11 shows the GMRES statistics for the computational tests. When running a simulation with $r_1$, the scheme didn't converge on coarse level 5 within 3000 GMRES steps. Decreasing the size of the rigid domain lead to a decrease of the needed GMRES steps and also allowed the usage of a coarse level 5, with still less then 1000 GMRES iterations in average. This indicates, that the movement of the rigid domain has a negligible influence on the GMG preconditioner, since the iteration numbers are of the same magnitude in this moving rigid domain scenario as in the static scenario of Sec. 4.6.2.

Table 4.11: Statistics for the GMRES iterations per time step. With $r = 0.1$ there was no convergence within $n_{\max} \leq 3000$ GMRES steps. The fine mesh is set to level 7 with 2 166 786 degrees of freedom.

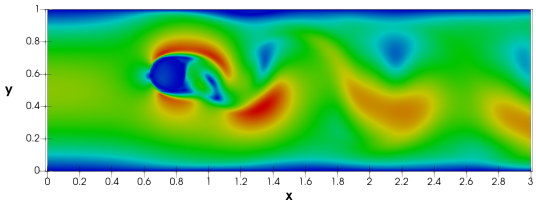
| coarse lvl | coarse DoFs | $\bar{n}_{\text{GMRES}}$ | $n_{\min}$ | $n_{\max}$ | $\bar{n}_{\text{GMRES}}$ | $n_{\min}$ | $n_{\max}$ |
|---|---|---|---|---|---|---|---|
| 5 | 34 306 | - | - | - | 927 | 685 | 1315 |
| 6 | 136 194 | 938 | 631 | 1404 | 615 | 441 | 1091 |
| | | r = $r_1 = 0.1$ | | | r = $r_2 = 0.05$ | | |

Fig. 4.12 illustrate the computed profiles of velocity and the pressure at time $t = 9.40$. The figures show that the application of the ghost penalty stabilization and extension of the discrete solution to the ghost rigid domain leads to stable approximations of the flow problem on a background mesh, as it was also the case in Sec. 3.5.6.

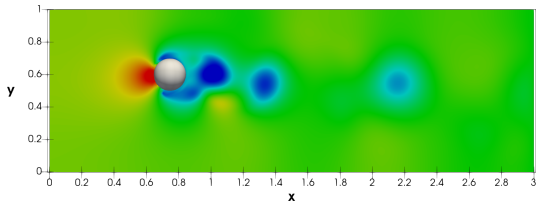
This computational results indicate, that the size of the ghost penalty extension has a big influence on the convergence behavior of the GMG method. For small rigid domains, the approach of a static background mesh works nicely and converges within a few hundred GMRES iterations. When increasing the size of the rigid domain the situation drastically changes and with a reasonable amount of resources no convergence could be obtained in the outer GMRES solver. This indicates, that it is favorable, in such scenarios, to choose the ghost penalty extension zone as small as possible, as it is done



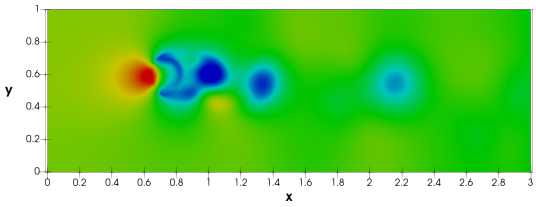
(a) Velocity field with overlay of moving rigid body.



(b) Velocity field with ghost penalty extension.



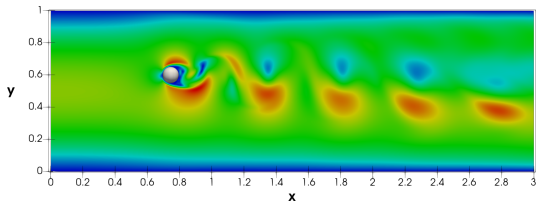
(c) Pressure field with overlay of rigid body.



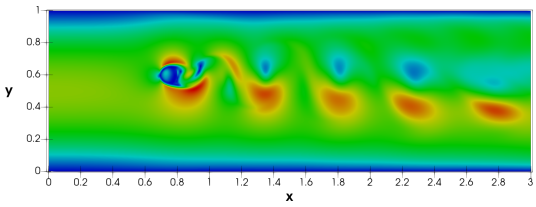
(d) Pressure field with ghost penalty extension.


(e) Scaling for the velocity and pressure plots.

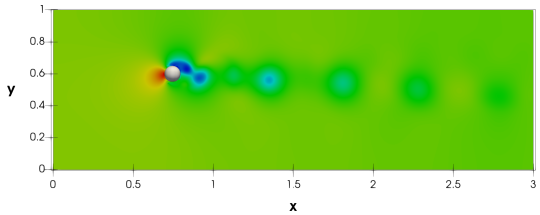
Figure 4.12: Solution of the CutFEM simulation for $r = 0.1$ at $t = 9.40$.



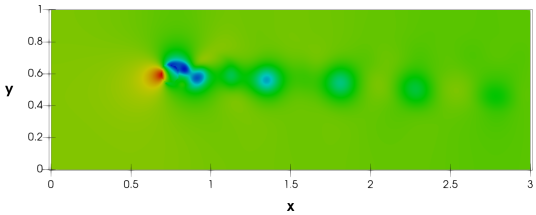
(a) Velocity field with overlay of moving rigid body.



(b) Velocity field with ghost penalty extension.



(c) Pressure field with overlay of rigid body.



(d) Pressure field with ghost penalty extension.


(e) Scaling for the velocity and pressure plots.

Figure 4.13: Solution of the CutFEM simulation for $r = 0.05$ at $t = 9.40$.

e.g. in [26]. The drawback of such an approach is, that a dynamic degree of freedom management is needed, where cells can get activated and deactivated, depending for example on the position and speed of the rigid body. In practice this means updating the structure of the underlying sparse system matrix, which might be computationally expensive, especially in a parallel environment. Such an approach is not in the scope of this work and, especially if the size of the rigid domain is small compared to the fluid domain, the approach, presented in this chapter, performs reasonably well and allows to get solutions of the system efficiently.

### 4.6.4 Evaluation and outlook

In this section we investigate the reason of the drastic increase in the iteration numbers of the GMRES outer solver, in comparison to the benchmark for flow problems on fitted grids in Sec. 4.4. Firstly, we compare the spectral condition number $\kappa_2$ of various settings. To compare fitted and CutFEM simulations, we use the grids of the benchmark of Sec. 4.4.1 and Sec. 4.6.3, with varying refinement levels. Table 4.12 shows the corresponding degrees of freedom of the underlying grid as well as the spectral condition number for the system matrix in the first time step and first Newton iteration.

Table 4.12: Spectral condition number $\kappa_2$ for different scenarios. The index in the grid name stands for the number of (global) grid refinements.

| Grid | DoFs | $\gamma_{v,p}$ | $\kappa_2$ |
|---|---|---|---|
| $\text{Cut}_3$ | 3714 | 0.0001 | 1.356e5 |
| | | 0.001 | 1.356e5 |
| | | 0.01 | 2.297e5 |
| | | 0.1 | 2.259e6 |
| | | 1.0 | 2.256e7 |
| | | 10.0 | 2.257e8 |
| $\text{Fitted}_1$ | 4288 | - | 3.451e5 |
| $\text{Cut}_4$ | 14 466 | 0.01 | 8.557e5 |
| $\text{Fitted}_2$ | 16 672 | - | 1.157e6 |

It can be seen, that for $\gamma_{v,p} = 10^{-2}$, that we used on the finest mesh of our simulations, the condition numbers of the cut and fitted settings are of the same magnitude. The condition number in the cut scenarios is also dependent on the ghost penalty parameters $\gamma_v$ and $\gamma_p$, but as long as it is chosen small enough, the condition number is comparable

to the one of the fitted cases. This indicates, that the procedure of extending the solution to the whole computational domain is not the reason for the high iteration numbers.

To further study the underlying problem of the high iteration numbers of the GMG scheme, we consider the development of the residual $r_l$ on each level after the smoothing operation, cf. Fig. 4.2. An efficient smoother should result in a notable decrease of $r_l$. In our numerical test we use the setup of the fitted (see Sec. 4.4.1) and unfitted (see Sec. 4.6.2) DFG benchmark and just perform one single time step and set $I = (0, 0.01]$, so $\tau = 0.01$. In this time step we compute the residual $r_l$ on each level $l$ before and after the smoothing operation. Table 4.13 shows the average over all smoothing steps in this time step on each level of the residuum after the smoothing operation $r_{\text{after}}$ and the residuum before the smoothing operation $r_{\text{before}}$ in the fitted scenario of Sec. 4.4.1. We performed two simulations, one with just one single smoothing iteration (which we used in all the previous examples) and one with four smoothing iterations.

Table 4.13: Development of the residual after the smoothing operation as an indication for the efficiency of the Vanka smoother in a fitted scenario.

| lvl | DoFs | $\overline{\frac{\|r_{\text{after}}\|}{\|r_{\text{before}}\|}}$ | $\overline{\frac{\|r_{\text{after}}\|}{\|r_{\text{before}}\|}}$ |
|---|---|---|---|
| 5 | 2 080 256 | 0.0344 | 0.0004 |
| 4 | 521 984 | 0.3488 | 0.2420 |
| 3 | 131 456 | 0.6849 | 0.1350 |
| 2 | 33 344 | 0.7680 | 0.1611 |
| 1 | 8576 | 0.5761 | 0.1426 |
| | | 1 smoothing step | 4 smoothing steps |

The results here show, that the Vanka smoother performs well on each level, even with just one single smoothing iteration we have a clear decrease of the residuum.

Table 4.14 shows the same values for the CutFEM setting of Sec. 4.6.2 with $\gamma_{v,p} = 10^{-2}$.

Here one can see, why this scheme did not converge when the coarse level was set coarser than level 7: the smoothing operation leads to an increase of the residuum. The situation even gets worse if multiple smoothing iterations are performed.

If we use the experimentally optimized penalty parameters for $\gamma_{v,p}$ of Sec. 4.6.2, the situation slightly improves. The results for this setup are shown in Table 4.15.

Table 4.14: Development of the residual after the smoothing operation as an indication for the efficiency of the Vanka smoother in a CutFEM scenario with $\gamma_{v,p} = 10^{-2}$.

| lvl | DoFs | $\overline{\frac{\|r_{\text{after}}\|}{\|r_{\text{before}}\|}}$ | $\overline{\frac{\|r_{\text{after}}\|}{\|r_{\text{before}}\|}}$ |
|---|---|---|---|
| 8 | 3 610 626 | 0.7363 | 4.2270 |
| 7 | 904 194 | 1.0989 | 4.6794 |
| 6 | 226 818 | 1.3569 | 9.5322 |
| 5 | 57 090 | 1.8199 | 6.6149 |
| 4 | 14 466 | 0.9732 | 2.3996 |
| | | 1 smoothing step | 4 smoothing steps |

Table 4.15: Development of the residual after the smoothing operation as an indication for the efficiency of the Vanka smoother in a CutFEM scenario.

| lvl | DoFs | $\gamma_{v,p}$ | $\overline{\frac{\|r_{\text{after}}\|}{\|r_{\text{before}}\|}}$ | $\overline{\frac{\|r_{\text{after}}\|}{\|r_{\text{before}}\|}}$ |
|---|---|---|---|---|
| 8 | 3 610 626 | 1e−2 | 0.7068 | 4.0256 |
| 7 | 904 194 | 7e−3 | 0.8573 | 1.6067 |
| 6 | 226 818 | 8e−4 | 1.1957 | 5.1429 |
| 5 | 57 090 | 6e−4 | 1.4365 | 5.1745 |
| 4 | 14 466 | 2e−2 | 0.8686 | 0.5028 |
| | | | 1 smoothing step | 4 smoothing steps |

Despite the slightly better results in comparison to Table 4.14 the decrease in the residuum is far smaller than in the fitted setting of Table 4.13. On some levels it is also increasing, so the cell-based Vanka smoother doesn't work as expected. When applied with four iterations the situation even gets worse on nearly all levels.

The results of this chapter indicate, that not the ghost penalty stabilization with the implicit extension of the physical flow quantities to the whole computational domain seems to be the reason for the slower convergence. This is indicated, by comparable spectral condition numbers in the fitted and in the cut case.

The underlying reason of the problem seems to be, that the cell based Vanka smoother is not able to reduce the high frequency errors, if the physical flow domain is extended in the context of Sec. 3.2.3. A possible resort could be to extend the cell based Vanka smoother to a patch based one, as it was done in [49], [109]. In these works all velocity degrees of freedom that couple directly to a pressure degree of freedom were considered, when assembling the local cell matrices in the smoother. This approach should be better suited for (nearly) incompressible flows [49]. Since we use discontinuous pressure

elements (see Chapter 2) for the GMG, this would result again in just a local cell based coupling. But in our approach, one also has to consider, that the ghost penalty extension of Sec. 3.2.3 leads to an artificial coupling between the pressure and velocity degrees of freedom in the stabilization zone. Therefore, adjusting the cell based Vanka smoother to a patch based Vanka smoother, where each patch consists of a cell and all its neighboring cells in the stabilization area (see Fig. 3.5b) could be a promising extension. An implementation and study has to remain as a work for the future.

## 4.7 Summary

In this chapter a parallel GMG preconditioner with a cell-based Vanka smoother for solving the nonstationary, incompressible Navier–Stokes equations was presented. Its efficient implementation in the deal.II finite element library was discussed. Discontinuous Galerkin methods and inf-sup stable pairs of finite element spaces with discontinuous pressure elements were used for the discretization of the time and space variables, respectively. The GMG preconditioner was applied to a flexible GMRES method for solving the Newton linearized algebraic problem. The performance properties of the GMG method and its parallel implementation were analyzed computationally for the two- and three-dimensional benchmark problem of flow around a cylinder in a fixed grid scenario for time independent domains and body fitted grids.

The approach was then applied to CutFEM simulations with evolving domains. In scenarios, where the rigid domain is small, compared to the fluid domain, the results confirm a good convergence behavior in challenging 2D problems. With an increase of the ghost penalty extension zone, an increase of the iteration numbers was observed.

# 5 Variational time discretization of higher order and regularity on time-independent domains

In the past, space-time finite element methods with continuous and discontinuous discretizations of the time and space variables have been studied strongly for the numerical simulation of incompressible flow, wave propagation, transport phenomena or even problems of multi-physics; cf., e.g., [1], [4], [53], [87], [110]–[114]. Appreciable advantage of variational space-time discretizations is that they offer the potential to naturally construct higher order methods. In practice, these methods provide accurate results by reasonable numerical costs and on computationally feasible grids. Further, variational space-time discretizations allow to utilize fully adaptive finite element techniques to change the magnitude of the space and time elements in order to increase accuracy and decrease numerical costs; cf. [115], [116]. Strong relations between variational time discretization, collocation and Runge–Kutta methods have been observed [117], [118]. Nodal superconvergence properties of variational time discretizations have also been proved [119]. Further, the fully coupled treatment of all time steps versus time-marching approaches is a topic of current research. In particular, the simultaneous computation of all time steps imposes high demands on the linear solver technology (cf., e.g., [2], [3], [120]).

When it comes to flow problems and one is interested in the full pressure trajectory there occurs another challenge. If we expand the discrete solution of the velocity and pressure field $(\boldsymbol{v}_{\tau|I_n}, p_{\tau|I_n}) \in \mathbb{P}_k(I_n; \boldsymbol{V_h}) \times \mathbb{P}_k(I_n; Q_h)$ on each time interval $I_n$ in terms of the basis functions, we get

$$\boldsymbol{v}_{\tau,h|I_n}(\boldsymbol{x},t) = \sum_{l=0}^{k} \boldsymbol{v}_{n,l}(\boldsymbol{x})\chi_l(t)\,,$$

$$p_{\tau,h|I_n}(\boldsymbol{x},t) = \sum_{l=0}^{k} p_{n,l}(\boldsymbol{x})\chi_l(t)\,,$$

where we use a polynomial degree of order $k$ and a nodal Lagrange basis $\chi_l(t)$ in time. If one uses for example a continuous Galerkin–Petrov (cGP) of order 1 in time scheme, a natural choice for the position of the nodal support points are the Gauss-Lobatto points, i.e. the beginning and the end point of each time interval, cf. Fig. 5.1.

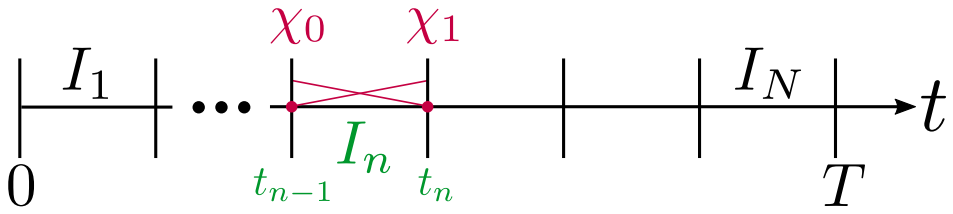



Figure 5.1: Sketch of the time mesh with nodal trial functions, using Gauss-Lobatto support points on each $I_n$.

Therefore, an initial pressure value is needed in the first time interval $I_1$, which is not given by the problem description. The discontinuous Galerkin (dG) time discretization scheme, that is used in Chapters 3 and 4 of this work further circumvents the problem of the need for an initial pressure, when applied to the pressure variable of the Navier–Stokes equations.

Recently, a modification of the standard continuous Galerkin–Petrov method for the time discretization was introduced for ordinary differential equations (cf. [8]). The modification comes through imposing collocation conditions involving the discrete solution's derivatives at the discrete time nodes while on the other hand downsizing the test space of the discrete variational problem compared with the standard cGP approach. We refer to these schemes as Galerkin–collocation (GC) methods. This

Table 5.1: Finite element based time discretizations.

| Acronym | Regularity | DoFs (eff.) in time | Conv. Rate |
|---|---|---|---|
| dG($k$) | $L^2(0,T;B)$ | $k+1$ | $k+1$ |
| cGP($k$) | $C([0,T];B)$ | $k$ | $k+1$ |
| GCC$^1$($k$) | $C^1([0,T];B)$ | $k-1$ | $k+1$ |

combination of variational approximation in time by finite element techniques with the concepts of collocation methods results in a very efficient discretization scheme, which is on a discrete level of higher regularity. For example the GCC$^1$($k$) scheme results in a globally C$^1$-regular solution and offers a superior DoFs to convergence rate ratio, in comparison with traditional discontinuous Galerkin or continuous Galerkin-Petrov schemes. Table 5.1 shows the generally needed amount of degrees of freedom and the

expected convergence rate of different finite element methods applied to the time domain.
The key ingredients and innovations of the approach are:

A. Higher order regularity in time of the fully discrete approximation;

B. Linear systems of reduced complexity;

Ingredient [A] is a direct consequence of the construction of the schemes. Higher order regularity is ensured by imposing collocation conditions at the discrete time nodes and endpoints of the subintervals $[t_{n-1}, t_n]$, for $n = 1, \ldots, N$, of the global time interval $[0, T]$. Higher order regularity in time might offer appreciable advantages for future approximations of coupled multi-physics systems if higher order time derivatives of the discrete solution of one subproblem arise as coefficient functions in other subproblems. Ingredient [B] is ensured by the proper choice of a special for the discrete in time function spaces. Thereby, simple vector identities for the degrees of freedom in time are obtained at the left endpoints of the subintervals without generating computational costs. These vector identities can then be exploited to eliminate conditions from the algebraic systems and reduce its size compared to the standard continuous Galerkin–Petrov approximation in time; cf. [119].

Our motivation for using implicit time discretization schemes comes from the overall goal to apply the proposed Galerkin–collocation techniques to mixed systems like, for instance, fluid-structure interaction for free flow modeled by the Navier–Stokes equations or fully dynamic poroelasticity [121].
In Sec. 5.1 we apply this approach to the wave equation. The numerical example of Sec. 5.1.3.4 mimics typical studies of structural health monitoring (cf. Fig. 5.2) and demonstrates the superiority of the Galerkin–collocation approach over a standard continuous Galerkin–Petrov method admitting continuity and no differentiability in time of the discrete solution. This superiority was the motivation to also apply this scheme to the Navier–Stokes equations, which is done in Sec. 5.2. It is a first step towards the simulation of fully coupled fluid–structure interaction, where the wave equation is coupled with the Navier–Stokes equations.
In contrast to Chapter 3 the following computational studies are performed on fitted meshes without exploiting CutFEM features.

## 5.1 Galerkin–collocation methods for waves phenomena

The accurate and efficient numerical simulation of wave phenomena continues to remain a challenging task and attract researchers' interest. Wave phenomena are studied in various branches of natural sciences and technology. For instance, fluid-structure interaction, acoustics, poroelasticity, seismics, electro-magnetics and non-destructive material inspection represent prominent fields in that wave propagation is studied. One of our key application for wave propagation is structural health monitoring of lightweight material (for instance, carbon-fibre reinforced polymers) by ultrasonic waves in aerospace engineering. The conceptional idea of this new and intelligent approach is sketched in Fig. 5.2. The structure is equipped with an integrated

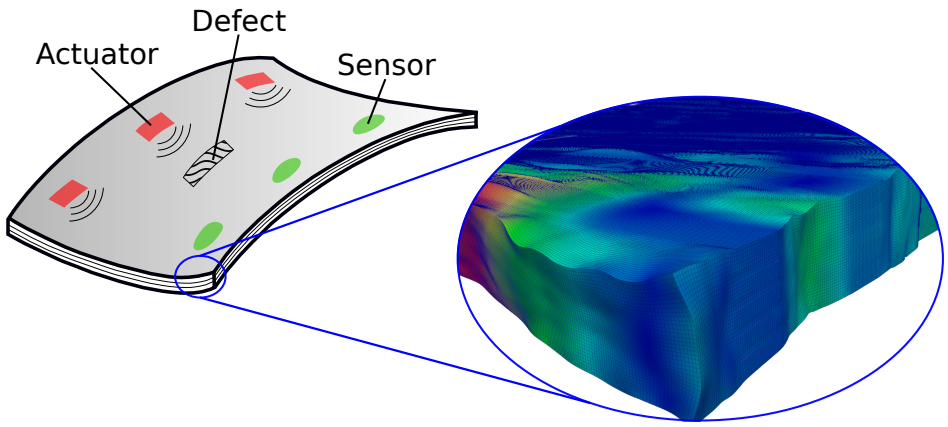



Figure 5.2: Concept of structural health monitoring with finite element simulation (scaled displacement field) illustrating the expansion of elastic waves.

actuator-sensor network. The ultrasonic waves that are emitted by the actuators interact with material defects of the solid structure. By means of an inverse modeling, the signals that are recorded by the sensors monitor material failure (cf. [122]) and, as perspective for the future, may allow prognoses about the structure's residual lifetime. The design of such monitoring systems and the signal interpretation require the elucidation of wave propagation in composite material which demands for highly advanced and efficient numerical simulation techniques.

For the sake of brevity, standard conforming finite element methods are used for the discretization of the spatial variables. This is done since we focus here on the time discretization. In the literature it has been mentioned that discontinuous finite element methods in space offer appreciable advantages over continuous ones for the discretization of wave equations; cf., e.g., [123], [124]. The application of, for instance, the

symmetric interior penalty discontinuous Galerkin method (cf. [88], [122] along with a Galerkin–collocation discretization in time, is straightforward.

### 5.1.1 Mathematical problem

As a prototype model, we study the hyperbolic wave problem

$$\begin{aligned} \partial_t^2 u - c^2 \Delta u &= f\,, \in \Omega \times I\,, \\ u(0) = u_0\,, \quad \partial_t u(0) &= v_0\,, \text{ in } \Omega\,, \\ u = g^u\,, \text{ on } \partial\Omega_D \times I\,, \quad \partial_n u &= 0\,, \text{ on } \partial\Omega_N \times I\,. \end{aligned} \tag{5.1}$$

In our application of structural health monitoring (cf. Fig. 5.2), $u$ denotes the scalar valued displacement field, $c \in R$ with $c > 0$, is a material parameter and $f$ an external force acting on the domain $\Omega \subset \mathbb{R}^d$, with $d = 2, 3$. Further, $g^u$ is a prescribed trace on the Dirichlet part $\partial\Omega_D$ of the boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, with $\partial\Omega_D \cap \partial\Omega_N = \emptyset$. By $\partial_n$ we denote the normal derivative with outer unit normal vector $\boldsymbol{n}$. Homogeneous Neumann boundary conditions on $\partial\Omega_N$ are prescribed for brevity. Problem (5.1) is well-posed and admits a unique solution $(u, \partial_t u) \in L^2(0,T;H^1(\Omega)) \times L^2(0,T;L^2(\Omega))$ under appropriate assumptions about the data; cf. [125]. By embedding, $u \in C([0,T];H^1(\Omega))$ and $v \in C([0,T];L^2(\Omega))$ is ensured; cf. [126]. Throughout, it is assumed that the solution admits all the (improved) regularity being necessary in the arguments.

To derive the Galerkin–collocation approach, problem (5.1) is first rewritten as a first order system in time for the unknowns $(u, v)$, with $v = \partial_t u$,

$$\begin{aligned} \partial_t u - v &= 0\,, \\ \partial_t v - c^2 \Delta u &= f\,. \end{aligned} \tag{5.2}$$

Further, we represent the unknowns $u$ and $v$ in terms of

$$u = u^0 + u^D \quad \text{and} \quad v = v^0 + v^D\,. \tag{5.3}$$

Here, $u^D$, $v^D \in C(\overline{I};H^1(\Omega))$ are supposed to be (extended) functions with traces $u^D = g^u$ and $v^D = g^v := \partial_t g^u$ on the Dirichlet part $\partial\Omega_D$ of $\partial\Omega$.

Using (5.2) and (5.3), we then consider solving the following variational problem:

**Problem 5.1.** *Find* $(u^0, v^0) \in L^2(0,T;H^1_{0,D}(\Omega)) \times L^2(0;T;H^1_{0,D}(\Omega))$ *such that*

$$u^0(0) = u_0 - u^D(0)\,, \qquad v^0(0) = v_0 - v^D(0) \tag{5.4}$$

*and, for all* $(\phi,\psi) \in (L^2(0;T;H^1_{0,D}(\Omega)))^2$,

$$\int_I \big\langle \partial_t u^0, \phi\big\rangle_\Omega - \big\langle v^0, \phi\big\rangle_\Omega \,\mathrm{d}t = 0\,, \tag{5.5}$$

$$\begin{aligned}\int_I \big\langle \partial_t v^0, \psi\big\rangle_\Omega + \langle c^2\nabla u^0, \nabla\psi\rangle_\Omega \,\mathrm{d}t = \int_I \Big( &\big\langle f,\psi\big\rangle_\Omega + \big\langle \partial_n u, \psi\big\rangle_{\partial\Omega_N} \\ &- \big\langle \partial_t v^D, \psi\big\rangle_\Omega - \langle c^2\nabla u^D, \nabla\psi\rangle_\Omega \Big)\,\mathrm{d}t\,.\end{aligned} \tag{5.6}$$

**Remark 5.1.**

- *We note that the correct treatment of inhomogeneous time-dependent boundary conditions is an import issue in the application of variational space-time methods. The space-time discretization that is derived below (cf. Sec. 5.1.2) and based on the variational problem (5.5), (5.6) ensures convergence rates of optimal order in space and time, also for time-dependent boundary conditions. This is confirmed by the second of the numerical experiments given in Sec. 5.1.3.3.*
- *Our Galerkin–collocation approach is based on solving, along with some collocation conditions, the variational equations (5.5), (5.6) in finite dimensional subspaces. In particular, the same approximation space will be used for* $u^0$ *and* $v^0$*. For this reason, the solution space for* $v^0$ *and the test space in Eq. (5.6) are chosen slightly stronger than usually; cf. [5]. Choosing* $L^2(0;T;L^2(\Omega)))$ *instead, would have been sufficient.*

## 5.1.2 Galerkin-collocation schemes

In this section two families of Galerkin–collocation schemes are introduced. These families combine the concept of collation condition methods applied to the spatially discrete counterpart of the equations (5.2) with the finite element discretization of the variational equations (5.5), (5.6). The collocation constrains then allow us to reduce the size of the discrete test spaces for the variational conditions compared to a standard Galerkin–Petrov approach; cf. [88].

We let $\mathcal{A}_h : H^1_{0,D}(\Omega) \mapsto V_h$ be the discrete operator that is defined by

$$\langle \mathcal{A}_h w, v_h \rangle = \langle \nabla w, \nabla v_h \rangle \quad \text{for all } v_h \in V_h \,. \tag{5.7}$$

Moreover, $(u_{0,h}, v_{0,h}) \in V_h^2$ and $(u^D_{\tau,h}, v^D_{\tau,h}) \in (C([0,T]; V_h))^2$ define suitable finite element approximations of the initial values $(u_0, v_0)$ and the extended boundary values $(u^D, v^D)$ in Eq. (5.3). Here, we use interpolation in $V_h$ of the given data.

Now the classes of Galerkin–collocation schemes are defined, where the line of [8] is followed. We restrict ourselves to the schemes studied in the numerical experiments presented in Sec. 5.1.3.3, 5.1.3.4 and 5.1.4.2. The definition of classes of Galerkin–collocation schemes with even higher regularity in time is straightforward, but not done here.

**Definition 5.1** (***C*$^{\boldsymbol{l}}$–regular in time Galerkin-collocation schemes GCC$^{\boldsymbol{l}}$(*k*)).** *Let $l \in \{1,2\}$ be fixed and $k \geq 2l+1$. For $n = 1, \ldots, N$ and given $(u_{\tau,h|I_{n-1}}(t_{n-1}), v_{\tau,h|I_{n-1}}(t_{n-1})) \in V_h^2$ for $n > 1$ and $u_{\tau,h|I_0}(t_0) = u_{0,h}$, $v_{\tau,h|I_0}(t_0) = v_{0,h}$ for $n = 1$, find $(u^0_{\tau,h|I_n}, v^0_{\tau,h|I_n}) \in (\mathbb{P}_k(I_n; V_h))^2$ such that, for $s_0 \in \mathbb{N}_0$, $s_1 \in \mathbb{N}$ with $s_0, s_1 \leq l$,*

$$\partial_t^{s_0} w^0_{\tau,h|I_n}(t_{n-1}) = \partial_t^{s_0} w^0_{\tau,h|I_{n-1}}(t_{n-1}) \,, \quad \text{for } w^0_{\tau,h} \in \left\{ u^0_{\tau,h}, v^0_{\tau,h} \right\} \,, \tag{5.8}$$

$$\partial_t^{s_1} u^0_{\tau,h|I_n}(t_n) - \partial_t^{s_1-1} v^0_{\tau,h|I_n}(t_n) = 0 \,, \tag{5.9}$$

$$\begin{aligned} \partial_t^{s_1} v^0_{\tau,h|I_n}(t_n) + \mathcal{A}_h \partial_t^{s_1-1} u^0_{\tau,h|I_n}(t_n) &= \partial_t^{s-1} f(t_n) \\ &\quad - \partial_t^{s_1} v^D_{\tau,h|I_n}(t_n) - \mathcal{A}_h \partial_t^{s_1-1} u^D_{\tau,h|I_n}(t_n) \,, \end{aligned} \tag{5.10}$$

*and, for all $(\varphi_{\tau,h}, \psi_{\tau,h}) \in (\mathbb{P}_0(I_n; V_h))^2$,*

$$\int_{I_n} \left( \langle \partial_t u^0_{\tau,h}, \varphi_{\tau,h} \rangle_\Omega - \langle v^0_{\tau,h}, \varphi_{\tau,h} \rangle_\Omega \right) \mathrm{d}t = 0 \,, \tag{5.11}$$

$$\begin{aligned} \int_{I_n} \left( \langle \partial_t v^0_{\tau,h}, \psi_{\tau,h} \rangle_\Omega + \langle \mathcal{A} u^0_{\tau,h}, \psi_{\tau,h} \rangle_\Omega \right) \mathrm{d}t &= \int_{I_n} \langle f, \psi_{\tau,h} \rangle_\Omega \, \mathrm{d}t \\ &\quad - \int_{I_n} \left( \langle \partial_t v^D_{\tau,h}, \psi_{\tau,h} \rangle_\Omega + \langle \mathcal{A}_h u^D_{\tau,h}, \psi_{\tau,h} \rangle_\Omega \right) \mathrm{d}t \,. \end{aligned} \tag{5.12}$$

**Remark 5.2.**

- *In Eq. (5.8), the discrete initial values $(\partial_t u_{\tau,h}(0), \partial_t v_{\tau,h}(0))$ arise for $s_0 = 1$. For $\partial_t u_{\tau,h}(0)$ we use a suitable finite element approximation $v_{0,h} \in V_h$ (here, an interpolation) of $v_0 \in V$. For $\partial_t v_{\tau,h}(0)$ we evaluate the wave equation in the initial time point and use a suitable finite element approximation (here, an interpolation)*

*of* $\partial_t^2 u(0) = c^2 \Delta u(0) + f(0)$. *For* $s_0 = 2$ *in Eq.* (5.8), *the initial value* $\partial_t^2 v_{\tau,h}(0)$ *is computed as a suitable finite element approximation (here, an interpolation) of* $\partial_t^3 u(0) = c^2 \Delta \partial_t u(0) + \partial_t f(0)$. *Mathematically, this approach requires that the partial equation and its time derivative are satisfied up to the initial time point and, thereby, sufficient regularity of the continuous solution. Without such regularity assumptions, the application of higher order discretization schemes cannot be justified rigorously. Nevertheless, in practice such methods often show a superiority over lower-order ones, even for solutions without the expected high regularity (cf. Sec. 5.1.3.4).*

- *From Eq.* (5.8), $(u_{\tau,h}, v_{\tau,h}) \in (C^l(\overline{I}; V_h))^2$, *for fixed* $l \in \{1, 2\}$, *is easily concluded.*

An optimal order error analysis for the GCC$^1(k)$ family of schemes of Definition 5.1 is provided in [9]. The following theorem is proved.

**Theorem 5.1** (Error estimates for $(\boldsymbol{u_{\tau,h}}, \boldsymbol{v_{\tau,h}})$ of GCC$^1(\boldsymbol{k})$)**.** *Let* $l = 1$ *and* $k \geq 3$. *For the error* $(e^{\,u}, e^{\,v}) = (u - u_{\tau,h}, v - v_{\tau,h})$ *of the fully discrete scheme GCC*$^l(k)$ *of Definition 5.1 there holds that*

$$\begin{aligned} \|e^{\,u}(t)\| + \|e^{\,v}(t)\| &\lesssim \tau^{k+1} + h^{p+1}\,, \quad t \in \overline{I}\,, \\ \|\nabla e^{\,u}(t)\| &\lesssim \tau^{k+1} + h^{p}\,, \quad t \in \overline{I}\,, \end{aligned} \tag{5.13}$$

*as well as*

$$\begin{aligned} \|e^{\,u}(t)\|_{L^2(I;H)} + \|e^{\,v}(t)\|_{L^2(I;H)} &\lesssim \tau^{k+1} + h^{p+1}\,, \\ \|\nabla e^{\,u}(t)\|_{L^2(I;H)} &\lesssim \tau^{k+1} + h^{p}\,. \end{aligned} \tag{5.14}$$

Error estimates for the GCC$^2(k)$ family remain as a work for the future. In Sec. 5.1.4.2, the convergence of GCC$^2(5)$ is demonstrated numerically. Further, we note that a computationally cheap post-processing of improved regularity and accuracy for continuous Galerkin–Petrov methods is presented and studied in [119].

In the next sections we study the schemes GCC$^1(3)$ and GCC$^2(5)$ of Definition 5.1 in detail. Their algebraic forms are derived and the algebraic linear solver are presented. Finally, the results of our numerical experiments with the proposed methods are presented. Here, we restrict ourselves to the lowest-order cases with $k = 3$ for $l = 1$ and $k = 5$ for $l = 2$ of Definition 5.1. This is sufficient to demonstrate the potential of the Galerkin–collocation approach and its superiority over the standard continuous Galerkin approach in space and time [112], [119]. An implementation of GCC$^l(k)$ for higher values of $k$ along with efficient algebraic solvers is currently still missing.

### 5.1.3 Galerkin–collocation GCC$^1$(3)

Here, we derive the algebraic system of the GCC$^1$(3) approach and discuss our algebraic solver for the arising block system. For brevity, the derivation is done for $k = 3$ only. The generalization to larger values of $k$ is straightforward; cf. [9].

#### 5.1.3.1 Fully discrete system

To derive the discrete counterparts of the variational conditions (5.11), (5.12) and the collocation constrains (5.8) to (5.10), we let $\{\phi_j\}_{j=1}^J \subset V_h$, denote a (global) nodal Lagrangian basis of $V_h$. The mass matrix $\boldsymbol{M}$ and the stiffness matrix $\boldsymbol{A}$ are defined by

$$\boldsymbol{M} \coloneqq \big(\langle \phi_i, \phi_j \rangle_\Omega\big)_{i,j=1}^J \,, \qquad \boldsymbol{A} \coloneqq \big(\langle \nabla\phi_i, \nabla\phi_j \rangle_\Omega\big)_{i,j=1}^J \,, \tag{5.15}$$

On the reference time interval $\hat{I} = [0,1]$ we define a Hermite-type basis $\{\hat{\xi}_l\}_{l=0}^3 \subset \mathbb{P}_3(\hat{I};\mathbb{R})$ of $\mathbb{P}_3(\hat{I};\mathbb{R})$ by the conditions

$$\begin{aligned}
&\hat{\xi}_0(0) = 1\,, \quad \hat{\xi}_0(1) = 0\,, \quad \partial_t\hat{\xi}_0(0) = 0\,, \quad \partial_t\hat{\xi}_0(1) = 0\,,\\
&\hat{\xi}_1(0) = 0\,, \quad \hat{\xi}_1(1) = 0\,, \quad \partial_t\hat{\xi}_1(0) = 1\,, \quad \partial_t\hat{\xi}_1(1) = 0\,,\\
&\hat{\xi}_2(0) = 0\,, \quad \hat{\xi}_2(1) = 1\,, \quad \partial_t\hat{\xi}_2(0) = 0\,, \quad \partial_t\hat{\xi}_2(1) = 0\,,\\
&\hat{\xi}_3(0) = 0\,, \quad \hat{\xi}_3(1) = 0\,, \quad \partial_t\hat{\xi}_3(0) = 0\,, \quad \partial_t\hat{\xi}_3(1) = 1\,.
\end{aligned} \tag{5.16}$$

These conditions then define the basis of $\mathbb{P}_3(\hat{I};\mathbb{R})$ by

$$\hat{\xi}_0 = 1 - 3t^2 + 2t^3, \quad \hat{\xi}_1 = t - 2t^2 + t^3, \quad \hat{\xi}_2 = 3t^2 - 2t^3, \quad \hat{\xi}_3 = -t^2 + t^3. \tag{5.17}$$

By means of the affine transformation $T_n(\hat{t}) \coloneqq t_{n-1} + \tau_n \cdot \hat{t}$, with $\hat{t} \in \hat{I}$, from the reference interval $\hat{I}$ to $I_n$ such that $t_{n-1} = T_n(0)$ and $t_n = T_n(1)$, the basis $\{\xi_l\}_{l=0}^3 \subset \mathbb{P}_3(I_n;\mathbb{R})$ is given by $\xi_l = \hat{\xi}_l \circ T_n^{-1}$ for $l = 0,\dots,3$. In terms of basis functions, $w_{\tau,h} \in \mathbb{P}_3(I_n;V_h)$ is thus represented by

$$w_{\tau,h}(\boldsymbol{x},t) = \sum_{l=0}^{3}\sum_{j=1}^{J} w_{n,l,j}\phi_j(\boldsymbol{x})\xi_l(t)\,, \quad (\boldsymbol{x},t) \in \Omega \times \overline{I_n}\,. \tag{5.18}$$

For $\zeta_0 \equiv 1$ on $\overline{I_n}$, a test basis of $\mathbb{P}_0(I_n;V_h)$ is then given by

$$\mathcal{B} = \{\phi_1\zeta_0, \dots, \phi_J\zeta_0\}\,. \tag{5.19}$$

To evaluate the time integrals on the right-hand side of Eq. (5.12) we still use the Hermite-type interpolation operator $I_{\tau|I_n}$, on $I_n$, defined by

$$I_{\tau|I_n} g(t) \coloneqq \sum_{s=0}^{l} \tau_n^s \hat{\xi}_s(0) \underbrace{\partial_t^s g_{|I_n}(t_{n-1})}_{=:g_s} + \sum_{s=0}^{l} \tau_n^s \hat{\xi}_{s+l+1}(1) \underbrace{\partial_t^s g|_{I_n}(t_n)}_{=:g_{s+l+1}} . \tag{5.20}$$

Here, the values $\partial_t^s g_{|I_n}(t_{n-1})$ and $\partial_t^s g_{|I_n}(t_n)$ in (5.20) denote the corresponding one-sided limits of values $\partial_t^s g(t)$ from the interior of $I_n$.

Now, we can put the equations of the proposed GCC$^1$(3) approach in their algebraic forms. In the variational equations (5.11) and (5.12), we use the representation (5.18) for each component of $(u_{\tau,h}, v_{\tau,h}) \in (\mathbb{P}_3(I_n; V_h))^2$, choose the test functions (5.19) and interpolate the right-hand side of (5.12) by applying (5.20). All of the arising time integrals are evaluated analytically. Then, we can recover the variational conditions (5.11) and (5.12) on the subinterval $I_n$ in their algebraic forms

$$\boldsymbol{M}\left(-\boldsymbol{u}_{n,0}^0 + \boldsymbol{u}_{n,2}^0\right) - \tau_n \boldsymbol{M}\left(\frac{1}{2}\boldsymbol{v}_{n,0}^0 + \frac{1}{12}\boldsymbol{v}_{n,1}^0 + \frac{1}{2}\boldsymbol{v}_{n,2}^0 - \frac{1}{12}\boldsymbol{v}_{n,3}^0\right) = \boldsymbol{0}\,, \tag{5.21}$$

$$\begin{aligned}\boldsymbol{M}\left(-\boldsymbol{v}_{n,0}^0 + \boldsymbol{v}_{n,2}^0\right) + \tau_n \boldsymbol{A}\left(\frac{1}{2}\boldsymbol{u}_{n,0}^0 + \frac{1}{12}\boldsymbol{u}_{n,1}^0 + \frac{1}{2}\boldsymbol{u}_{n,2}^0 - \frac{1}{12}\boldsymbol{u}_{n,3}^0\right) = \\ \tau_n \boldsymbol{M}\left(\frac{1}{2}\boldsymbol{f}_{n,0} + \frac{1}{12}\boldsymbol{f}_{n,1} + \frac{1}{2}\boldsymbol{f}_{n,2} - \frac{1}{12}\boldsymbol{f}_{n,3}\right) - \boldsymbol{M}\left(-\boldsymbol{v}_{n,0}^D + \boldsymbol{v}_{n,2}^D\right) \\ - \tau_n \boldsymbol{A}\left(\frac{1}{2}\boldsymbol{u}_{n,0}^D + \frac{1}{12}\boldsymbol{u}_{n,1}^D + \frac{1}{2}\boldsymbol{u}_2^D - \frac{1}{12}\boldsymbol{u}_{n,3}^D\right).\end{aligned} \tag{5.22}$$

This gives us the first two equations for the set of eight unknown solution vectors $\mathcal{L} = \{\boldsymbol{u}_{n,0}^0, \ldots, \boldsymbol{u}_{n,3}^0, \boldsymbol{v}_{n,0}^0, \ldots, \boldsymbol{v}_{n,3}^0\}$ on each subinterval $I_n$, where each of these vectors is defined by means of (5.18) through $\boldsymbol{w} = (w_1, \ldots, w_J)^\top$ for $\boldsymbol{w} \in \mathcal{L}$.

Next, we study the algebraic forms of the collocations conditions (5.8) to (5.10). By means of the definition (5.16) of the basis of $\mathbb{P}(I_n; \mathbb{R})$, the constraints (5.8) read as

$$\boldsymbol{u}_{n,0}^0 = \boldsymbol{u}_{n-1,2}^0\,, \quad \boldsymbol{u}_{n,1}^0 = \boldsymbol{u}_{n-1,3}^0\,, \quad \boldsymbol{v}_{n,0}^0 = \boldsymbol{v}_{n-1,2}^0\,, \quad \boldsymbol{v}_{n,1}^0 = \boldsymbol{v}_{n-1,3}^0\,. \tag{5.23}$$

By means of (5.16) along with (5.15), the conditions (5.9) and (5.10) can be recovered as

$$\boldsymbol{M}\frac{1}{\tau_n}\boldsymbol{u}_{n,3}^0 - \boldsymbol{M}\boldsymbol{v}_{n,2}^0 = \boldsymbol{0}\,, \tag{5.24}$$

$$\boldsymbol{M}\frac{1}{\tau_n}\boldsymbol{v}_{n,3}^0 + \boldsymbol{A}\boldsymbol{u}_{n,2}^0 = \boldsymbol{M}\boldsymbol{f}_{n,2} - \boldsymbol{M}\frac{1}{\tau_n}\boldsymbol{v}_{n,3}^D - \boldsymbol{A}\boldsymbol{u}_{n,2}^D\,. \tag{5.25}$$

Putting relations (5.23) into the identities (5.21) and (5.22) and combining the resulting equations with (5.24) and (5.25) yields for the subinterval $I_n$ the linear block system

$$\boldsymbol{S}\boldsymbol{x} = \boldsymbol{b}\,, \tag{5.26}$$

for the vector of unknowns

$$\boldsymbol{x} = \left( \left(\boldsymbol{v}^0_{n,2}\right)^\top, \left(\boldsymbol{v}^0_{n,3}\right)^\top, \left(\boldsymbol{u}^0_{n,2}\right)^\top, \left(\boldsymbol{u}^0_{n,3}\right)^\top \right)^\top \tag{5.27}$$

and the system $\boldsymbol{S}$ and right-hand side $\boldsymbol{b}$ given by

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{M} & \boldsymbol{0} & \boldsymbol{0} & \frac{1}{\tau_n}\boldsymbol{M} \\ \boldsymbol{0} & \frac{1}{\tau_n}\boldsymbol{M} & \boldsymbol{A} & \boldsymbol{0} \\ -\frac{\tau_n}{2}\boldsymbol{M} & \frac{\tau_n}{12}\boldsymbol{M} & \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{M} & \boldsymbol{0} & \frac{\tau_n}{2}\boldsymbol{A} & -\frac{\tau_n}{12}\boldsymbol{A} \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{M}\left(\boldsymbol{f}_{n,2} - \frac{1}{\tau_n}\boldsymbol{v}^D_{n,3}\right) - \boldsymbol{A}\boldsymbol{u}^D_{n,2} \\ \boldsymbol{M}\left(\boldsymbol{u}^0_{n,0} + \frac{\tau_n}{2}\boldsymbol{v}^0_{n,0} + \frac{\tau_n}{12}\boldsymbol{v}^0_{n,1}\right) \\ \boldsymbol{b}_{n,4} \end{pmatrix}, \tag{5.28}$$

with $\boldsymbol{b}_{n,4} = \boldsymbol{M}\big(\boldsymbol{v}^0_{n,0} + \boldsymbol{v}^D_{n,0} - \boldsymbol{v}^D_{n,2} + \frac{\tau_n}{2}(\boldsymbol{f}_{n,0} + \boldsymbol{f}_{n,2}) + \frac{\tau_n}{12}(\boldsymbol{f}_{n,1} - \boldsymbol{f}_{n,3})\big) - \boldsymbol{A}\big(\frac{\tau_n}{2}(\boldsymbol{u}^0_{n,0} + \boldsymbol{u}^D_{n,0} + \boldsymbol{u}^D_{n,2}) + \frac{\tau_n}{12}(\boldsymbol{u}^0_{n,1} + \boldsymbol{u}^D_{n,1} - \boldsymbol{u}^D_{n,3})\big)$. By means of the collocation constraints (5.23), the number of unknown coefficient vectors for the discrete solution $(u_{\tau,h|I_n}, v_{\tau,h|I_n}) \in (\mathbb{P}_3(I_n; V_h))^2$ is thus effectively reduced from eight to four vectors, assembled now in $\boldsymbol{x}$ by (5.27).

We note that the first two rows of Eq. (5.28) represent the collocation conditions (5.24) and (5.25). They have a sparser structure then the last two rows representing the variational conditions which can be advantageous or exploited for the construction of efficient iterative solvers for (5.26). Compared with a pure variational approach (cf. [87], [112], [127]), more degrees or freedom are obtained directly by computationally cheap vector identities (cf. (5.21)) in GCC$^l(k)$ such that they can be eliminated from the overall linear system and, thereby, used to reduce the systems size.

#### 5.1.3.2 Solver technology

In the sequel, we present two different iterative approaches for solving the linear system (5.26) with the non-symmetric matrix $\boldsymbol{S}$. In Sec. 5.1.3.4, a runtime comparison between the two concepts is provided. As basic toolbox we use the deal.II finite element library [57] along with the Trilinos library [94] for parallel computations.

**1. Approach: Condensing the linear system** The first method for solving (5.26) is based on the concepts developed in [122]. The key idea is to use Gaussian block elimination within the system matrix $\boldsymbol{S}$ and end up with a linear system with matrix $\boldsymbol{S}_r$ of reduced size for one of the subvectors in $\boldsymbol{x}$ in (5.27) only, and to compute the remaining subvectors of (5.27) by computationally cheap post-processing steps afterwards. The reduced system matrix $\boldsymbol{S}_r$ should have sufficient potential that an efficient preconditioner for the iterative solution of the reduced system can be constructed. Of course, the Gauss elimination on the block level can be done in different ways. The goal of our approach is to avoid the inversion of the stiffness matrix $\boldsymbol{A}$ in (5.28) in the computation of the condensed system matrix $\boldsymbol{S}_r$ such that a matrix-vector multiplication with $\boldsymbol{S}_r$ just involves calculating $\boldsymbol{M}^{-1}$. At least for discontinuous Galerkin methods in space, where $\boldsymbol{M}$ is block diagonal, this is computationally cheap; cf. [88], [122]. We note that a continuous Galerkin approach in space is used here only in order to simplify the notation and since the discretization in time by the combined Galerkin–collocation approach is in the scope of interest.

Here, we choose the subvector $\boldsymbol{u}_{n,2}^0$ of $\boldsymbol{x}$ in (5.27) as the essential unknown, i.e. as the unknown solution vector of the condensed system with matrix $\boldsymbol{S}_r$. By block Gaussian elimination we then end up with solving the linear system,

$$\left(\boldsymbol{M} + \frac{\tau_n^2}{12}\boldsymbol{A} + \frac{\tau_n^4}{144}\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}\right)\boldsymbol{u}_{n,2}^0 = \boldsymbol{b}_{n,r}\,, \tag{5.29}$$

with right-hand side vector

$$\begin{aligned}\boldsymbol{b}_{n,r} = {}& \boldsymbol{M}\left(\frac{1}{2}\boldsymbol{f}_{n,0} + \frac{1}{12}\boldsymbol{f}_{n,1} + \frac{1}{3}\boldsymbol{f}_{n,2} - \frac{1}{12}\boldsymbol{f}_{n,3}\right) + \boldsymbol{M}\left(2\boldsymbol{v}_{n,0}^0 + \frac{1}{6}\boldsymbol{v}_{n,1}^0 + \frac{2}{\tau_n}\boldsymbol{u}_{n,0}^0\right)\\
& - \boldsymbol{A}\left(\frac{2}{3}\tau_n\boldsymbol{u}_{n,0}^0 + \frac{1}{12}\tau_n\boldsymbol{u}_{n,1}^0 + \frac{1}{12}\tau_n^2\boldsymbol{v}_{n,2}^0 + \frac{1}{72}\tau_n^2\boldsymbol{v}_{n,1}^0\right) + \boldsymbol{M}\left(2\boldsymbol{v}_{n,0}^D + \frac{1}{6}\boldsymbol{v}_{n,1}^D + \frac{2}{\tau_n}\boldsymbol{u}_{n,0}^D - \frac{2}{\tau_n}\boldsymbol{u}_{n,2}^D\right)\\
& + \boldsymbol{A}\left(\frac{1}{72}\tau_n^3\boldsymbol{f}_{n,2} - \boldsymbol{M}^{-1}\frac{1}{72}\tau_n^3\boldsymbol{u}_{n,2}^D\right) + \boldsymbol{A}\left(-\frac{2}{3}\tau_n\boldsymbol{u}_{n,0}^D - \frac{1}{12}\tau_n\boldsymbol{u}_{n,1}^D - \frac{\tau_n}{6}\boldsymbol{u}_{n,2}^D - \frac{\tau_n^2}{12}\boldsymbol{v}_{n,0}^D - \frac{\tau_n^2}{72}\boldsymbol{v}_{n,1}^D\right).\end{aligned} \tag{5.30}$$

The product of $\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}$ in (5.29) mimics the discretization of a fourth order operator due to the appearance of the product of $\boldsymbol{A}$ with its "weighted" form $\boldsymbol{M}^{-1}\boldsymbol{A}$. Thereby, the conditioning number of the condensed system is strongly increased (cf. [122]) which is the main drawback in this concept of condensing the overall system (5.26) to (5.29) for the essential unknown $\boldsymbol{u}_{n,2}^0$. On the other hand, since $\boldsymbol{M}$ and $\boldsymbol{A}$ are symmetric

and, thus, $\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A} = (\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A})^\top$, the condensed matrix $\boldsymbol{S}_r$ is symmetric such that the preconditioned conjugate gradient method can be applied. Solving systems of type (5.29) is carefully studied in [88], [122] and the references given therein.

We solve (5.29) by the conjugate gradient method. The left preconditioning operator

$$\boldsymbol{P} = \boldsymbol{K}_\mu \boldsymbol{M}^{-1} \boldsymbol{K}_\mu = \left(\mu \boldsymbol{M} + \frac{\tau_n^2}{4}\boldsymbol{A}\right) \boldsymbol{M}^{-1} \left(\mu \boldsymbol{M} + \frac{\tau_n^2}{4}\boldsymbol{A}\right), \tag{5.31}$$

with positive $\mu \in \mathbb{R}$ chosen such that the spectral norm of $\boldsymbol{P}^{-1}\boldsymbol{S}_r$ is minimized, is applied. For details of the choice of the parameter $\mu$, we refer to [88], [122]. Here, we use $\mu = \sqrt{11/2}$. In order to apply the preconditioning operator $\boldsymbol{P}$ in the conjugate gradient iterations, without assembling $\boldsymbol{P}$ explicitly, i.e. to solve the auxiliary system with matrix $\boldsymbol{P}$, we have to solve linear systems for the mass matrix $\boldsymbol{M}$ and the stiffness matrix $\boldsymbol{A}$. For this, we use embedded conjugate gradient iterations combined with an algebraic multigrid preconditioner of the Trilinos library [94]. The overall algorithm for solving (5.29) is sketched in Fig. 5.3. The advantage of this approach is that we just have to store $\boldsymbol{M}$ and $\boldsymbol{A}$ as sparse matrices in the computer memory. We never have to assemble the full matrix $\boldsymbol{S}$ from (5.28), nor do we have to store the reduced matrix $\boldsymbol{S}_r$ from (5.29). Finally, the remaining unknown subvectors $\boldsymbol{v}_{n,2}^0, \boldsymbol{v}_{n,3}^0$ and $\boldsymbol{u}_{n,3}^0$ in (5.26) are successively computed in post-processing steps.

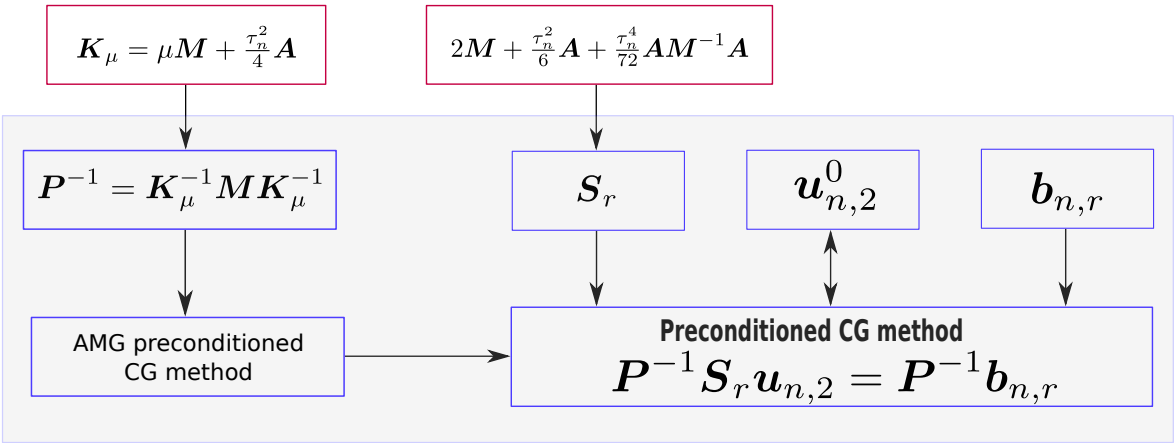



Figure 5.3: Preconditioning and solver for the condensed system (5.29) of GCC$^1$(3).

**2. Approach: Solving the non-symmetric system** The second approach used to solve (5.26) relies on assembling the system matrix $\boldsymbol{S}$ of (5.28) as a sparse matrix and solving the resulting non-symmetric system. For smaller dimensions of $\boldsymbol{S}$ a parallel direct solver [89] is used. For constant time step sizes $\tau_n$ the matrix $\boldsymbol{S}$ needs to be factorized once only, which results in excellent performance properties for large sequences

of time steps. For high-dimensional problems with interest in practice, we use the Generalized Minimal Residual (GMRES) method, an iterative Krylov subspace method, to solve (5.26). The drawback of this approach then comes through the necessity to provide an efficient preconditioner, i.e. an approximation to the inverse of $\boldsymbol{S}$, for the complex block matrix $\boldsymbol{S}$ of (5.28). Here, we use the algebraic multigrid method as preconditioning technique. We use the MueLue preconditioner [128], which is part of the Trilinos project, with non-symmetric smoothed aggregation. We use an usual V-cycle algorithm along with a symmetric successive over-relaxation (SSOR) smoother with a damping factor of 1.33. The design of efficient algebraic solvers for block systems like (5.26), and for higher order variational time discretizations in general, is still an active field of research. We expect further improvement in the future.

#### 5.1.3.3 Numerical convergence tests

In this section we present a numerical convergence test for the proposed GCC$^1$(3) approach of Definition 5.1 and Sec. 5.1.3.1, respectively. For the solution $\{u, v\}$ of Eq. (5.2) and the fully discrete approximation GCC$^1$(3) of Definition 5.1 we let

$$e^u \coloneqq u(\boldsymbol{x}, t) - u_{\tau,h}(\boldsymbol{x}, t), \qquad e^v \coloneqq v(\boldsymbol{x}, t) - v_{\tau,h}(\boldsymbol{x}, t). \tag{5.32}$$

We study the error $(e^u, e^v)$ with respect to the norms

$$\|e^w\|_{L^\infty(L^2)} \coloneqq \max_{t\in I} \left( \int_\Omega \|e^w\|^2 \mathrm{d}x \right)^{\frac{1}{2}}, \quad \|e^w\|_{L^2(L^2)} \coloneqq \left( \int_I \int_\Omega \|e^w\|^2 \mathrm{d}x \, \mathrm{d}t \right)^{\frac{1}{2}}, \tag{5.33}$$

where $w \in (u, v)$, and in the energy quantities

$$|||E|||_{L^\infty} \coloneqq \max_{t\in I} \left( \|\nabla e^u\|^2 + \|e^v\|^2 \right)^{\frac{1}{2}}, \quad |||E|||_{L^2} \coloneqq \left( \int_I \int_\Omega \|\nabla e^u\|^2 + \|e^v\|^2 \mathrm{d}\boldsymbol{x} \, \mathrm{d}t \right)^{\frac{1}{2}}. \tag{5.34}$$

All $L^\infty$-norms in time are computed on the discrete time grid

$$I = \left\{ t_n^d | t_n^d = t_{n-1} + d \cdot k_n \cdot \tau_n, \quad k_n = 0.001, d = 0, \ldots, 999, n = 1, \ldots, N \right\}. \tag{5.35}$$

For our first convergence test we prescribe the solution

$$u_1(\boldsymbol{x}, t) = \sin(4\pi t) \cdot x_1 \cdot (x_1 - 1) \cdot x_2 \cdot (x_2 - 1). \tag{5.36}$$

on $\Omega \times I = (0,1)^2 \times [0,1]$. We let $c = 1$, use a constant mesh size $h_0 = 0.25$ and start with the time step size $\tau_0 = 0.1$. We compute the errors on a sequence of successively refined time meshes by halving the step sizes in each refinement step. We choose a bicubic discretization of the space variables in $V_h^3$ (cf. Eq. (2.7)) such that the spatial part of the solution is resolved exactly by its numerical approximation. Table 5.2 summarizes the computed errors and experimental orders of convergence. The expected convergence rates of Theorem 5.1 are nicely confirmed.

Table 5.2: Calculated errors for GCC$^1$(3) with solution (5.36).

| $\tau$ | $h$ | $\|e^u\|_{L^\infty(L^2)}$ | EOC | $\|e^v\|_{L^\infty(L^2)}$ | EOC | $\|\|\|E\|\|\|_{L^\infty}$ | EOC |
|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0$ | 2.318e-04 | – | 1.543e-03 | – | 1.574e-03 | – |
| $\tau_0/2^1$ | $h_0$ | 1.541e-05 | 3.91 | 9.694e-05 | 3.99 | 1.004e-04 | 3.97 |
| $\tau_0/2^2$ | $h_0$ | 9.825e-07 | 3.97 | 6.260e-06 | 3.95 | 6.478e-06 | 3.95 |
| $\tau_0/2^3$ | $h_0$ | 6.185e-08 | 3.99 | 3.946e-07 | 3.99 | 4.082e-07 | 3.99 |
| $\tau_0/2^4$ | $h_0$ | 3.873e-09 | 4.00 | 2.472e-08 | 4.00 | 2.557e-08 | 4.00 |
| $\tau_0/2^5$ | $h_0$ | 2.422e-10 | 4.00 | 1.548e-09 | 4.00 | 1.609e-09 | 3.99 |
| $\tau$ | $h$ | $\|e^u\|_{L^2(L^2)}$ | EOC | $\|e^v\|_{L^2(L^2)}$ | EOC | $\|\|\|E\|\|\|_{L^2}$ | EOC |
| $\tau_0/2^0$ | $h_0$ | 1.634e-04 | – | 1.232e-03 | – | 1.441e-03 | – |
| $\tau_0/2^1$ | $h_0$ | 1.070e-05 | 3.93 | 7.864e-05 | 3.97 | 9.269e-05 | 3.96 |
| $\tau_0/2^2$ | $h_0$ | 6.765e-07 | 3.98 | 4.943e-06 | 3.99 | 5.836e-06 | 3.99 |
| $\tau_0/2^3$ | $h_0$ | 4.240e-08 | 4.00 | 3.094e-07 | 4.00 | 3.654e-07 | 4.00 |
| $\tau_0/2^4$ | $h_0$ | 2.652e-09 | 4.00 | 1.934e-08 | 4.00 | 2.285e-08 | 4.00 |
| $\tau_0/2^5$ | $h_0$ | 1.659e-10 | 4.00 | 1.212e-09 | 4.00 | 1.433e-09 | 3.99 |

In our second numerical experiment we study the space-time convergence behavior of a solution satisfying non-homogeneous Dirichlet boundary conditions,

$$u_2(\boldsymbol{x}, t) = \sin(2 \cdot \pi \cdot t + x_1) \cdot \sin(2 \cdot \pi \cdot t \cdot x_2)\,, \tag{5.37}$$

on $\Omega \times I = (0,1)^2 \times [0,1]$. We choose a bicubic discretization in $V_h^3$ (cf. (2.7)) of the space variable. We refine the space-time mesh by halving both step sizes in each refinement step. Table 5.3 shows the computed errors and experimental orders of convergence for this example. In all measured norms, optimal rates in space and time (cf. Theorem 5.1) are confirmed. This underlines the correct treatment of the prescribed non-homogeneous Dirichlet boundary conditions.

Table 5.3: Calculated errors for $\mathrm{GCC}^1(3)$ with solution (5.37).

| $\tau$ | $h$ | $\|e^u\|_{L^\infty(L^2)}$ | EOC | $\|e^v\|_{L^\infty(L^2)}$ | EOC | $\|\|\|E\|\|\|_{L^\infty}$ | EOC |
|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 3.486e-03 | – | 3.602e-02 | – | 5.013e-02 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 2.329e-04 | 3.90 | 2.392e-03 | 3.90 | 3.338e-03 | 3.92 |
| $\tau_0/2^2$ | $h_0/2^2$ | 1.483e-05 | 3.97 | 1.527e-04 | 3.97 | 2.128e-04 | 3.98 |
| $\tau_0/2^3$ | $h_0/2^3$ | 9.320e-07 | 3.99 | 9.609e-06 | 3.99 | 1.338e-05 | 3.99 |
| $\tau_0/2^4$ | $h_0/2^4$ | 5.837e-08 | 4.00 | 6.022e-07 | 4.00 | 8.383e-08 | 4.00 |
| $\tau_0/2^5$ | $h_0/2^5$ | 3.649e-09 | 4.00 | 3.767e-08 | 4.00 | 5.243e-08 | 4.00 |

| $\tau$ | $h$ | $\|e^u\|_{L^2(L^2)}$ | EOC | $\|e^v\|_{L^2(L^2)}$ | EOC | $\|\|\|E\|\|\|_{L^2}$ | EOC |
|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 2.700e-03 | – | 2.568e-02 | – | 3.458e-02 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 1.771e-04 | 3.93 | 1.689e-03 | 3.93 | 2.278e-03 | 3.92 |
| $\tau_0/2^2$ | $h_0/2^2$ | 1.120e-05 | 3.98 | 1.070e-04 | 3.98 | 1.444e-04 | 3.98 |
| $\tau_0/2^3$ | $h_0/2^3$ | 7.020e-07 | 4.00 | 6.713e-06 | 3.99 | 9.061e-06 | 3.99 |
| $\tau_0/2^4$ | $h_0/2^4$ | 4.391e-08 | 4.00 | 4.199e-07 | 4.00 | 5.669e-07 | 4.00 |
| $\tau_0/2^5$ | $h_0/2^5$ | 2.744e-09 | 4.00 | 2.624e-08 | 4.00 | 3.543e-08 | 4.00 |

#### 5.1.3.4 Test case of structural health monitoring

Next, we consider a test problem that is based on [5] and related to typical problems of structural health monitoring by ultrasonic waves (cf. Fig. 5.2). We aim to compare the $\mathrm{GCC}^1(3)$ approach with a standard continuous in time Galerkin–Petrov approach cGP(2) of piecewise quadratic polynomials in time; cf. [87], [119] for details. The cGP(2) scheme has superconvergence properties in the discrete time nodes $t_n$ for $n = 1, \dots, N$ as shown in [119]. Thus,the errors $\max_{n=1,\dots,N} \|e^u(t_n)\|$ and $\max_{n=1,\dots,N} \|e^v(t_n)\|$ for the $\mathrm{GCC}^1(3)$ and the cGP(2) scheme admit the same fourth order rate of convergence in time and, thus, are comparable with respect to accuracy.

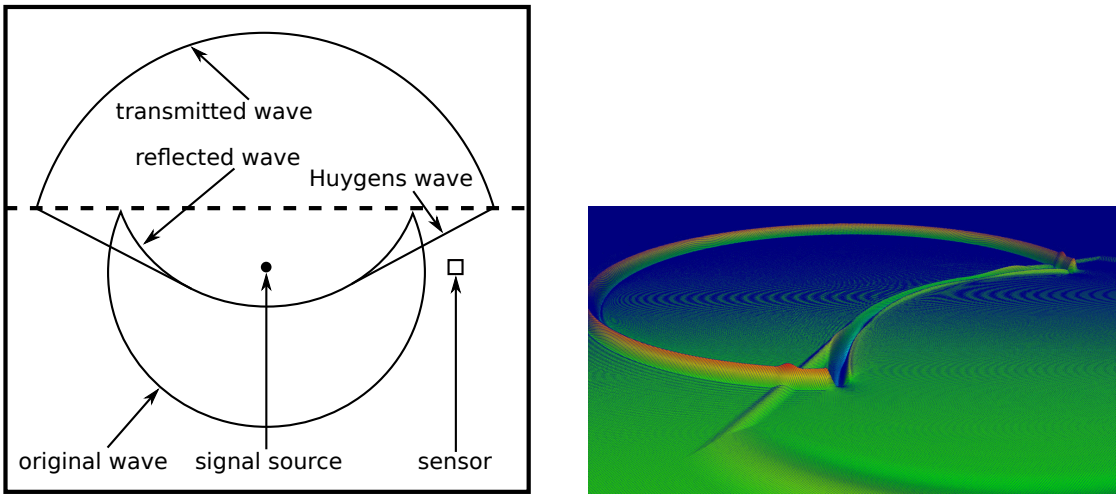
The test setting is sketched in Fig. 5.4a. We consider $\Omega \times I = (-1,1)^2 \times (0,1)$, let $f = 0$ and, for simplicity, prescribe homogeneous Dirichlet boundary conditions such that $u^D = 0$. For the initial value we prescribe a regularized Dirac impulse by

$$u_0(\boldsymbol{x}) = e^{-|\boldsymbol{x}_s|^2}\big(1 - |\boldsymbol{x}_s|^2\big)\Theta\big(1 - |\boldsymbol{x}_s|\big), \quad \boldsymbol{x}_s = 100\boldsymbol{x}\,, \tag{5.38}$$

where $\Theta$ is the Heaviside function. The coefficient function $c(\boldsymbol{x})$, mimicking a material parameter, has a jump discontinuity and is given by $c(\boldsymbol{x}) = 1$ for $x_2 < 0.2$ and $c(\boldsymbol{x}) = 9$ for $x_2 \geq 0.2$. Further we put $v_0 = 0$ for the second initial value. Finally, we define

the control region $\Omega_c = (0.75 - h_c, 0.75 + h_c) \times (-h_c, h_c)$ where we calculate the signal arrival, at a sensor position for instance, in terms of

$$u_c(t) = \int_{\Omega_c} u_{\tau,h}(\boldsymbol{x}, t) \mathrm{d}\boldsymbol{x}. \tag{5.39}$$



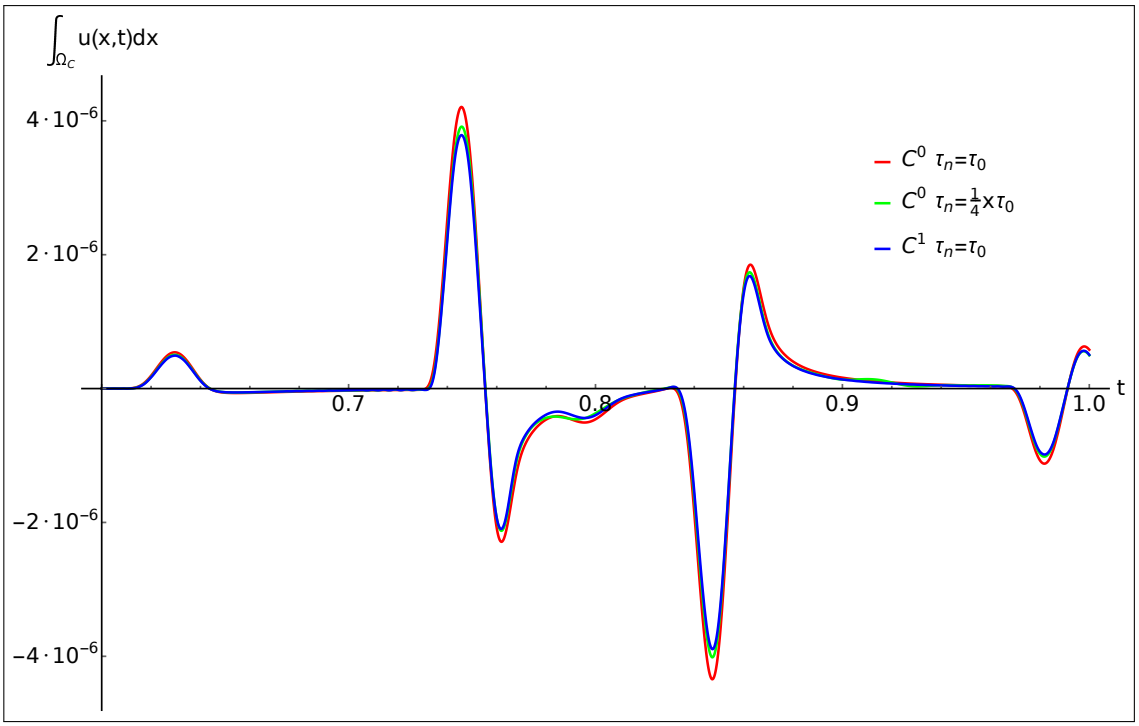

(a) Test setting (according to [5]). (b) Solution at $t = 0.5$ with spatial mesh.

Figure 5.4: Test case of structural health monitoring.

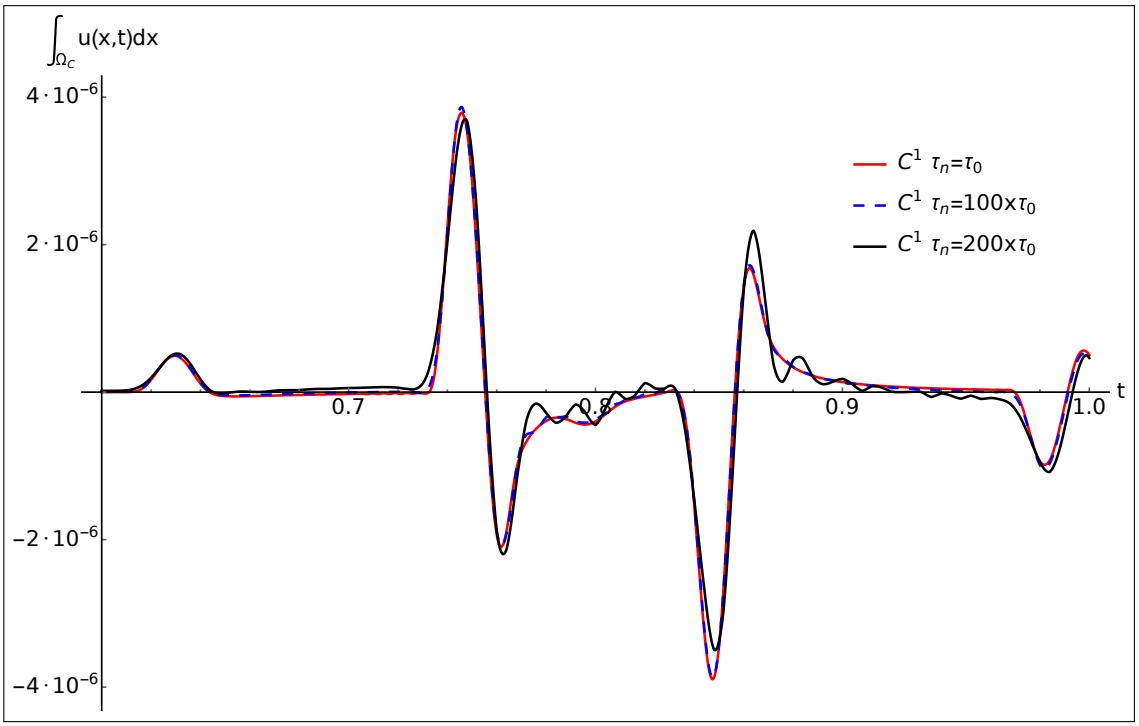
We choose a spatial mesh of 65 536 cells and $\mathbb{Q}_7$ elements; cf. Fig. 5.4b. This leads to more than $3.2 \times 10^6$ degrees of freedom in space in each time step for each of the solution vectors. For each computation of the control quantity (5.39), with $t \in (0, 1]$, we use a constant time step size $\tau_n$ for all time steps and compare the computation with the initially chosen reference time step size of $\tau_0 = 2 \times 10^{-5}$.

Fig. 5.5 shows the signal arrival and control quantity (5.39) over $t \in (0.6, 1)$ with different choices of the time step sizes for the Galerkin–collocation scheme GCC$^1$(3) and the standard Galerkin–Petrov approach cGP(2) (cf. [87], [119]) of a continuous in time approximation. For the cGP(2) approach, very small time step sizes are required to avoid over- and undershoots in the control quantity $u_c(t)$. For the GCC$^1$(3) approach with $C^1$ regularity in time, much larger time steps, approximately 100 times larger, can be applied without loss of accuracy compared to the fully converged reference solution given by GCC$^1$()3 with step size $\tau_0$. This clearly shows the superiority of the Galerkin–collocation scheme GCC$^1$(3).

In Table 5.4 the computational costs are summarized, where $r_1$ is the runtime for solving the condensed system by the approach of Sec. 5.1.3.2 and $r_2$ is the runtime for solving the block system by the approach of Sec. 5.1.3.2. For the cGP(2) approach,




(a) Control quantity (5.39) for cGP(2) (method $C^0$) with different time step sizes and reference solution GCC$^1$(3) (method $C^1$).




(b) Control quantity (5.39) for GCC$^1$(3) (method $C^1$) with different time step sizes.

Figure 5.5: Control quantity (5.39) for GCC$^1$(3) (method $C^1$) and cGP(2) (method $C^0$) for different time step sizes.

only the first of the either iterative solver techniques was implemented. Recalling from Fig. 5.5 that GCC$^1$(3) with $\tau_n = 100 \times \tau_0$ leads to the fully converged solution whereas cGP(2) with $\tau_n = \tau_0$ already shows over- and undershoots, a strong superiority of GCC$^1$(3) over cGP(2) is observed in Table 5.4. For both solver, a reduction in the wall clock time by a factor of about 25 is shown.

Table 5.4: Runtime (wall clock time) for GCC$^1$(3) (method $C^1$) and cGP(2) (method $C^0$) for different time step sizes and solvers of Sec. 5.1.3.2 ($r_1$) and 5.1.3.2 ($r_2$) .

| DoF (space) | cores | method | $\tau_n$ | $r_1$[h] | $r_2$[h] |
|---|---|---|---|---|---|
| $3.2 \times 10^6$ | 224 | $C^0$ | $0.25 \times \tau_0$ | 219.3 | - |
| | | $C^0$ | $\tau_0$ | 40.0 | - |
| | | $C^1$ | $\tau_0$ | 46.6 | 25.3 |
| | | $C^1$ | $2 \times \tau_0$ | 33.1 | 19.4 |
| | | $C^1$ | $25 \times \tau_0$ | 4.5 | 2.3 |
| | | $C^1$ | $35 \times \tau_0$ | 3.7 | 2.2 |
| | | $C^1$ | $50 \times \tau_0$ | 2.9 | 1.6 |
| | | $C^1$ | $100 \times \tau_0$ | 1.7 | 0.9 |
| | | $C^1$ | $200 \times \tau_0$ | 1.1 | 0.7 |
| $4.2 \times 10^6$ | 336 | $C^1$ | $50 \times \tau_0$ | 3.3 | 1.7 |

## 5.1.4 Galerkin–collocation GCC$^2$(5)

Here, we briefly derive the algebraic form of the Galerkin–collocation scheme GCC$^2(k)$ of Definition 5.1 with fully discrete solutions $(u_{\tau,h|I_n}, v_{\tau,h|I_n}) \in (X_\tau^k(V_h))^2$ such that $(u_{\tau,h}, v_{\tau,h}) \in (C^2(\overline{I}; V_h))^2$. For brevity, we restrict ourselves to the lowest polynomial degree in time $k = 5$ that is possible to get $C^2$-regularity. The convergence properties are then demonstrated numerically.

### 5.1.4.1 Fully discrete system

We follow the lines of Sec. 5.1.3.1 and use the notation introduced there. The six basis function of $\mathbb{P}_5(\hat{I}; \mathbb{R})$ on the reference interval $\hat{I}$ are defined by the conditions

$$\hat{\xi}_i^{(l)}(j) = \delta_{i-2*j-l,j} \qquad \forall\, i \in \{0, \cdots, 5\} \quad \wedge \quad j \in \{0, 1\}\,, \quad l \in \{0, 1, 2\}\,, \tag{5.40}$$

where $\delta_{i,j}$ denotes the usual Kronecker symbol. This gives us

$$\hat{\xi}_0 = -6t^5 + 15t^4 - 10t^3 + 1\,, \quad \hat{\xi}_1 = -3t^5 + 8t^4 - 6t^3 + t\,, \quad \hat{\xi}_2 = -\tfrac{1}{2}t^5 + \tfrac{3}{2}t^4 - \tfrac{3}{2}t^3 + \tfrac{1}{2}t^2\,,$$
$$\hat{\xi}_3 = 6t^5 - 15t^4 + 10t^3\,, \qquad \hat{\xi}_4 = -3t^5 + 7t^4 - 4t^3\,, \qquad \hat{\xi}_5 = \tfrac{1}{2}t^5 - t^4 + \tfrac{1}{2}t^3\,. \tag{5.41}$$

For this basis of $\mathbb{P}_5(\hat{I};\mathbb{R})$, the discrete variational conditions (5.11), (5.12) then read as

$$\boldsymbol{M}\left(-\boldsymbol{u}^0_{n,0} + \boldsymbol{u}^0_{n,3}\right) - \tau_n \boldsymbol{M}\left(\frac{1}{2}\boldsymbol{v}^0_{n,0} + \frac{1}{10}\boldsymbol{v}^0_{n,1} + \frac{1}{120}\boldsymbol{v}^0_{n,2} + \frac{1}{2}\boldsymbol{v}^0_{n,3} - \frac{1}{10}\boldsymbol{v}^0_{n,4} + \frac{1}{120}\boldsymbol{v}^0_{n,5}\right) = \boldsymbol{0}\,, \tag{5.42}$$

$$\begin{aligned}\boldsymbol{M}\left(-\boldsymbol{v}^0_{n,0} + \boldsymbol{v}^0_{n,3}\right) + \tau_n \boldsymbol{A}\left(\frac{1}{2}\boldsymbol{u}^{n,0}_{n,0} + \frac{1}{10}\boldsymbol{u}^0_{n,1} + \frac{1}{120}\boldsymbol{u}^0_{n,2} + \frac{1}{2}\boldsymbol{u}^0_{n,3} - \frac{1}{10}\boldsymbol{u}^0_{n,4} + \frac{1}{120}\boldsymbol{u}^0_{n,5}\right) = \\ \tau_n \boldsymbol{M}\left(\frac{1}{2}\boldsymbol{f}_{n,0} + \frac{1}{10}\boldsymbol{f}_{n,1} + \frac{1}{120}\boldsymbol{f}_{n,2} + \frac{1}{2}\boldsymbol{f}_{n,3} - \frac{1}{10}\boldsymbol{f}_{n,4} + \frac{1}{120}\boldsymbol{f}_{n,5}\right) - \boldsymbol{M}\left(-\boldsymbol{v}^D_{n,0} + \boldsymbol{v}^D_{n,3}\right) \\ - \tau_n \boldsymbol{A}\left(\frac{1}{2}\boldsymbol{u}^D_{n,0} + \frac{1}{10}\boldsymbol{u}^D_{n,1} + \frac{1}{120}\boldsymbol{u}^D_{n,2} + \frac{1}{2}\boldsymbol{u}^D_{n,3} - \frac{1}{10}\boldsymbol{u}^D_{n,4} + \frac{1}{120}\boldsymbol{u}^D_{n,5}\right).\end{aligned} \tag{5.43}$$

In the basis, the first collocation conditions (5.8) yield for $\boldsymbol{w}^0_{n,i} \in \{\boldsymbol{u}^0_{n,i}, \boldsymbol{v}^0_{n,i}\}$ that

$$\boldsymbol{w}^0_{n,0} = \boldsymbol{w}^0_{n-1,3}\,, \qquad \boldsymbol{w}^0_{n,1} = \boldsymbol{w}^0_{n-1,4}\,, \qquad \boldsymbol{w}^0_{n,2} = \boldsymbol{w}^0_{n-1,5}\,, \tag{5.44}$$

which reduces the number of unknown solution vectors by 6 on each subinterval $I_n$. For the collocation conditions (5.9), (5.10) at $t_n$ and $s = 1$ we deduce that

$$\boldsymbol{M}\frac{1}{\tau_n}\boldsymbol{u}^0_{n,4} - \boldsymbol{M}\boldsymbol{v}^0_{n,3} = \boldsymbol{0}\,, \qquad \boldsymbol{M}\frac{1}{\tau_n}\boldsymbol{v}^0_{n,4} + \boldsymbol{A}\boldsymbol{u}^0_{n,3} = \boldsymbol{M}\boldsymbol{f}_{n,3} - \boldsymbol{M}\frac{1}{\tau_n}\boldsymbol{v}^D_{n,4} - \boldsymbol{A}\boldsymbol{u}^D_{n,3}\,. \tag{5.45}$$

Similarly, for $s = 2$ the collocation conditions (5.9), (5.10) at $t_n$ read as

$$\boldsymbol{M}\frac{1}{\tau_n}\boldsymbol{u}^0_{n,5} - \boldsymbol{M}\boldsymbol{v}^0_{n,4} = \boldsymbol{0}\,, \qquad \boldsymbol{M}\frac{1}{\tau_n}\boldsymbol{v}^0_{n,5} + \boldsymbol{A}\boldsymbol{u}^0_{n,4} = \boldsymbol{M}\boldsymbol{f}_{n,4} - \boldsymbol{M}\frac{1}{\tau_n}\boldsymbol{v}^D_{n,5} - \boldsymbol{A}\boldsymbol{u}^D_{n,4}. \tag{5.46}$$

Finally, we recover the previous conditions as the linear system $\boldsymbol{S}\boldsymbol{x} = \boldsymbol{b}$ for the vector of unknowns $\boldsymbol{x} = \big((\boldsymbol{u}^0_{n,3})^\top, (\boldsymbol{u}^0_{n,4})^\top, (\boldsymbol{v}^0_{n,5})^\top, \boldsymbol{u}^0_{n,5})^\top\big)^\top$ and with the system matrix $\boldsymbol{S}$ and right-hand side vector $\boldsymbol{b}$ given by

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} & \frac{1}{\tau_n^2}\boldsymbol{M} \\ \boldsymbol{0} & \boldsymbol{A} & \frac{1}{\tau_n}\boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{M} & -\frac{1}{2}\boldsymbol{M} & -\frac{\tau_n}{120}\boldsymbol{M} & \frac{1}{10}\boldsymbol{M} \\ \frac{\tau_n}{2}\boldsymbol{A} & \frac{1}{\tau_n}\boldsymbol{M} - \frac{\tau_n}{10}\boldsymbol{A} & \boldsymbol{0} & \frac{\tau_n}{120}\boldsymbol{A} \end{pmatrix}. \quad \boldsymbol{b} = \begin{pmatrix} \boldsymbol{f}_{n,3} - \boldsymbol{A}\boldsymbol{u}^D_{n,3} - \frac{1}{\tau_n}\boldsymbol{M}\boldsymbol{v}^D_{n,4} \\ \boldsymbol{f}_{n,4} - \boldsymbol{A}\boldsymbol{u}^D_{n,4} - \frac{1}{\tau_n}\boldsymbol{M}\boldsymbol{v}^D_{n,5} \\ \boldsymbol{b}_{n,3} \\ \boldsymbol{b}_{n,4} \end{pmatrix}, \tag{5.47}$$

with $\boldsymbol{b}_{n,3} = \boldsymbol{M}\big(\boldsymbol{u}^0_{n,0} + \boldsymbol{u}^D_{n,0} - \boldsymbol{u}^D_{n,3} + \frac{\tau_n}{2}(\boldsymbol{v}^0_{n,0} + \boldsymbol{v}^D_{n,0}) + \frac{\tau_n}{10}(\boldsymbol{v}^0_{n,1} + \boldsymbol{v}^D_{n,1}) + \frac{\tau_n}{120}(\boldsymbol{v}^0_{n,2} + \boldsymbol{v}^D_{n,2}) + \tau_n(\frac{1}{2}\boldsymbol{v}^D_{n,3} - \frac{1}{10}\boldsymbol{v}^D_{n,4} + \frac{1}{120}\boldsymbol{v}^D_{n,5})$ and $\boldsymbol{b}_{n,4} = \boldsymbol{M}\big(\boldsymbol{v}^0_{n,0} + \boldsymbol{v}^D_{n,0} - \boldsymbol{v}^D_{n,3}\big) + \tau_n\big(\frac{1}{2}\boldsymbol{f}_{n,3} + \frac{1}{10}\boldsymbol{f}_{n,1} + \frac{1}{120}\boldsymbol{f}_{n,2} + \frac{1}{2}\boldsymbol{f}_{n,3} - \frac{1}{10}\boldsymbol{f}_{n,4} + \frac{1}{120}\boldsymbol{f}_{n,5}\big) - \tau_n\boldsymbol{A}\big(\frac{1}{2}(\boldsymbol{u}^0_{n,0} + \boldsymbol{u}^D_{n,0})2 + \frac{1}{10}(\boldsymbol{u}^0_{n,1} + \boldsymbol{u}^D_{n,1}) + \frac{1}{120}(\boldsymbol{u}^0_{n,2} + \boldsymbol{u}^D_{n,2}) + \frac{1}{2}\boldsymbol{u}^D_{n,3} - \frac{1}{10}\boldsymbol{u}^D_{n,4} + \frac{1}{120}\boldsymbol{u}^D_{n,5}\big)$.

#### 5.1.4.2 Iterative solver and convergence study

To solve the linear system $\boldsymbol{S}\boldsymbol{x} = \boldsymbol{b}$ with $\boldsymbol{S}$ from (5.47), we use block Gaussian elimination, as sketched in Sec. 5.1.3.2, to find a reduced system $\boldsymbol{S}_r\boldsymbol{u}^0_{n,4} = \boldsymbol{b}_r$ for the essential unknown $\boldsymbol{u}^0_{n,4}$. All remaining unknown subvectors of $\boldsymbol{x}$ can be computed in post-processing steps. In explicit form, the condensed system reads as

$$\Big(14400\boldsymbol{M} + 720\tau_n^2\boldsymbol{A} + 24\tau_n^4\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A} + \tau_n^6\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}\Big)\,\boldsymbol{u}^0_{n,4} = \boldsymbol{b}_r\,. \tag{5.48}$$

For brevity, we omit the exact definition of $\boldsymbol{b}_r$ that can be deduced easily from (5.47).

The matrix $\boldsymbol{S}_r$ is symmetric such that preconditioned conjugate gradient iterations are used for its solution. The preconditioner is constructed along the lines of Sec. 5.1.3.2. The remainder part $\tau_n^6\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}$ is still ignored in the construction of the preconditioner. Even though the remainder is weighted by the small factor $\tau_n^6$, numerical experiments indicate that this scaling is not sufficient to balance its impact on the iteration process. For the construction of an efficient preconditioning technique for $\boldsymbol{S}_r$ of GCC$^2$(5) further improvements are still necessary.

To illustrate the convergence behavior and performance of the GCC$^2$(5) Galerkin–collocation approach, we present in Table 5.5 our numerical results for the test problem

(5.36). The expected convergence of sixth order in time is nicely observed in all norms.

Table 5.5: Calculated errors for $\mathrm{GCC}^2(5)$ with solution (5.36).

| $\tau$ | $h$ | $\|e^u\|_{L^\infty(L^2)}$ | EOC | $\|e^v\|_{L^\infty(L^2)}$ | EOC | $\|\|\|E\|\|\|_{L^\infty}$ | EOC |
|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0$ | 8.748e-06 | – | 4.355e-05 | – | 4.985e-05 | – |
| $\tau_0/2^1$ | $h_0$ | 1.370e-07 | 6.00 | 7.404e-07 | 5.88 | 8.043e-07 | 5.95 |
| $\tau_0/2^2$ | $h_0$ | 2.165e-09 | 5.98 | 1.202e-08 | 5.95 | 1.266e-08 | 5.99 |
| $\tau_0/2^3$ | $h_0$ | 3.388e-11 | 6.00 | 1.883e-10 | 6.00 | 1.980e-10 | 6.00 |
| $\tau_0/2^4$ | $h_0$ | 5.301e-13 | 6.00 | 2.940e-12 | 6.00 | 3.093e-12 | 6.00 |

| $\tau$ | $h$ | $\|e^u\|_{L^2(L^2)}$ | EOC | $\|e^v\|_{L^2(L^2)}$ | EOC | $\|\|\|E\|\|\|_{L^2}$ | EOC |
|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0$ | 4.022e-06 | – | 2.996e-05 | – | 3.502e-05 | – |
| $\tau_0/2^1$ | $h_0$ | 6.353e-08 | 5.98 | 4.808e-07 | 5.96 | 5.599e-07 | 5.97 |
| $\tau_0/2^2$ | $h_0$ | 9.957e-10 | 6.00 | 7.565e-09 | 5.99 | 8.800e-09 | 5.99 |
| $\tau_0/2^3$ | $h_0$ | 1.557e-11 | 6.00 | 1.184e-10 | 6.00 | 1.377e-10 | 6.00 |
| $\tau_0/2^4$ | $h_0$ | 2.431e-13 | 6.00 | 1.849e-12 | 6.00 | 2.151e-12 | 6.00 |

## 5.2 Galerkin–collocation methods for flow problems

The results of applying the Galerkin–collocation time discretization schemes to the wave equation in Sec. 5.1 resulted in very efficient discretization schemes. For wave problems, the GCC$^1$ approach has demonstrated its superiority over pure continuous Galerkin–Petrov approximations in time (cGP), even for numerical challenging examples, see Sec. 5.1.3.4. Therefore, it seems to be natural to study the GCC$^1$ scheme also for the approximation of the Navier–Stokes equations.

In this section, the Galerkin–collocation approximation of the Navier–Stokes equations is developed along with Nitsche's method for imposing Dirichlet boundary conditions. Imposing the boundary conditions in a weak sense is a prerequisite for the application of immersed boundary methods, like in Chapter 3. For the Galerkin–collocation scheme GCC$^1$(3) with piecewise cubic polynomials the algebraic system along with its Newton linearization is derived explicitly which is done here since the Galerkin–collocation approximation of the Navier–Stokes equations is presented for the first time and to facilitate the traceability of its implementation. The expected convergence behavior of optimal order in time (and space) is demonstrated for the velocity and pressure variable. The effect of imposing boundary conditions is also computationally studied for flow around a cylinder. It is illustrated, that the accuracy of the approximation does not suffer from enforcing the boundary conditions by the application of Nitsche's method. Moreover, to show the superiority of the proposed approach over more standard time discretization schemes, a careful comparison of the GCC$^1$(3) approach with the continuous Galerkin–Petrov method using piecewise linear polynomials in time is performed. In algebraic form, the latter one can be recovered as the well-known Crank–Nicholson scheme. The errors of both approaches and, with regard to the future construction of efficient iterative solvers, the condition numbers of their Jacobian matrices are evaluated. Further, the performance properties for computing laminar flow around a cylinder are investigated. For this, the well-known DFG flow benchmark with $Re = 100$ (cf. [46]) as a challenging flow problem is used. The superiority of the GCC$^1$(3) approach is clearly observed.

### 5.2.1 Space–time finite element discretization with Galerkin–collocation time discretization and Nitsche's method

For the time discretization a continuous Galerkin–Petrov approach (cf., e.g., [1], [53], [127]) with discrete solutions $\boldsymbol{v}_{\tau,h} \in (C([0,T];V_h))^2$ and $p_{\tau,h} \in C([0,T];Q_h)$ is modified to a Galerkin–collocation approximation by combining the Galerkin techniques with the concepts of collocation.

#### 5.2.1.1 Continuous Galerkin–Petrov time discretization

For completeness and comparison, the standard continuous Galerkin–Petrov approximation in time of Problem 3.1 on time-independent domains is presented here, referred to as cGP($k$), along with the space discretization space; cf., e.g., [1], [53], [127]. This reads as follows.

**Problem 5.2** (Global problem of cGP($k$))**.** *Let an approximation $\boldsymbol{v}_{0,h} \in \boldsymbol{V}_h^{\mathrm{div}}$ of the initial value $\boldsymbol{v}_0$ be given. Let $\widehat{\boldsymbol{g}}_{\tau,h} \in \boldsymbol{X}_{\tau,h}^{k}$ denote a prolongation in the finite element spaces of the Dirichlet conditions on $\Gamma_i$ and $\Gamma_w$ . Put $\widehat{\boldsymbol{u}}_{\tau,h} = (\widehat{\boldsymbol{g}}_{\boldsymbol{\tau},\boldsymbol{h}}, 0)$. Let $\boldsymbol{f} \in L^2(0,T;\boldsymbol{V}')$ be given. Find $\boldsymbol{u}_{\tau,h} \in \widehat{\boldsymbol{u}}_{\tau,h} + \boldsymbol{X}_{\tau,h}^{k,0}$ such that $\boldsymbol{v}_{\tau,h}(0) = \boldsymbol{v}_{0,h}$ and*

$$\int_0^T \langle \partial_t \boldsymbol{v}_{\tau,h}, \boldsymbol{\psi}_{\tau,h} \rangle_{\Omega_f} + A((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))\mathrm{d}t = \int_0^T L(\psi_{\tau,h}; \boldsymbol{f}, \boldsymbol{g})\mathrm{d}t\,,$$

*for all $\boldsymbol{\phi}_{\tau,h} \in \boldsymbol{X}_{\tau,h}^{k-1,0}$.*

By choosing test functions in $\boldsymbol{Y}_{\tau,h}^{k-1,0}$ supported on a single subinterval $I_n$ of the time mesh $\mathcal{M}_\tau$ we recast Problem 5.2 as a time-marching scheme that is given by the following sequence of local problems on the subintervals $I_n$.

**Problem 5.3** (Local problem of cGP($k$))**.** *Let an approximation $\boldsymbol{v}_{0,h} \in \boldsymbol{V}_h^{\mathrm{div}}$ of the initial value $\boldsymbol{v}_0$ be given. Let $\widehat{\boldsymbol{g}}_{\tau,h} \in \boldsymbol{X}_{\tau,h}^{k}$ denote a prolongation into the finite element space of the Dirichlet conditions on $\Gamma_i$ and $\Gamma_w$ . Put $\widehat{\boldsymbol{u}}_{\tau,h} = (\widehat{\boldsymbol{g}}_{\boldsymbol{\tau},\boldsymbol{h}}, 0)$. Let $\boldsymbol{f} \in L^2(0,T;\boldsymbol{V}')$ be given. For $n = 1,\dots,N$ and given $\boldsymbol{u}_{\tau,h|I_{n-1}} \in \widehat{\boldsymbol{u}}_{\tau,h} + (\mathbb{P}_k(I_{n-1};V_h^0))^2 \times \mathbb{P}_k(I_{n-1};Q_h)$ find $\boldsymbol{u}_{\tau,h|I_n} \in \widehat{\boldsymbol{u}}_{\tau,h} + (\mathbb{P}_k(I_n;V_h^0))^2 \times \mathbb{P}_k(I_n;Q_h)$ such that*

$$\begin{aligned} \boldsymbol{v}_{\tau,h|I_n}(t_{n-1}) &= \boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1})\,, \\ p_{\tau,h|I_n}(t_{n-1}) &= p_{\tau,h|I_{n-1}}(t_{n-1}) \end{aligned} \tag{5.49a}$$

*and*

$$\int_{I_n} \langle \partial_t \boldsymbol{v}_{\tau,h}, \boldsymbol{\psi}_{\tau,h} \rangle_{\Omega_f} + A((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))\mathrm{d}t = \int_{I_n} L(\psi_{\tau,h}; \boldsymbol{f}, \boldsymbol{g})\mathrm{d}t\,, \tag{5.50}$$

*for all* $\boldsymbol{\phi}_{\tau,h} \in (\mathbb{P}_{k-1}(I_n; V_h^0))^2 \times \mathbb{P}_{k-1}(I_n; Q_h)$.

In the variational equation (5.50), Dirichlet boundary conditions for the velocity field are enforced by the definition of the function space $(\mathbb{P}_k(I_n; V_h^0))^2$ where by the discrete space $(V_h^0)^2$ homogeneous Dirichlet boundary conditions are prescribed for the velocity approximation $\boldsymbol{v}_{\tau,h} - \widehat{\boldsymbol{g}}_{\tau,h}$ and the test function $\boldsymbol{\psi}_{\tau,h}$. Using the Nitsche method, we solve instead of Problem 5.3 the following one to that we refer to as cGP($k$)–N.

**Problem 5.4** (Local Nitsche problem of cGP($k$): cGP($k$)–N)**.** *Let an approximation* $\boldsymbol{v}_{0,h} \in \boldsymbol{V}_h^{\mathrm{div}}$ *of the initial value* $\boldsymbol{v}_0$ *be given. For* $n = 1, \ldots, N$ *and given* $\boldsymbol{u}_{\tau,h|I_{n-1}} \in (\mathbb{P}_k(I_{n-1}; V_h))^2 \times \mathbb{P}_k(I_{n-1}; Q_h)$ *find* $\boldsymbol{u}_{\tau,h|I_n} \in (\mathbb{P}_k(I_n; V_h))^2 \times \mathbb{P}_k(I_n; Q_h)$ *such that*

$$\begin{aligned} \boldsymbol{v}_{\tau,h|I_n}(t_{n-1}) &= \boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1})\,, \\ p_{\tau,h|I_n}(t_{n-1}) &= p_{\tau,h|I_{n-1}}(t_{n-1}) \end{aligned} \tag{5.51}$$

*and*

$$\int_{I_n} \langle \partial_t \boldsymbol{v}_{\tau,h}, \boldsymbol{\psi}_{\tau,h} \rangle_{\Omega_f} + A_h((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))\mathrm{d}t = \int_{I_n} L_h(\psi_{\tau,h}; \boldsymbol{f}, \boldsymbol{g})\mathrm{d}t\,, \tag{5.52}$$

for all $\boldsymbol{\phi}_{\tau,h} \in (\mathbb{P}_{k-1}(I_n; V_h))^2 \times \mathbb{P}_{k-1}(I_n; Q_h)$.

In practice, the integrals on the right-hand side of Eqs. (5.50) and (5.52) is evaluated by means of an appropriate quadrature formula; cf. [1], [53], [119], [127].

**Remark 5.3** (Definition of initial pressure)**.**

- *In Problem Problems 5.3 and 5.4, the quantities* $\boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1})$ *and* $p_{\tau,h|I_{n-1}}(t_{n-1})$ *still need to be defined for* $n = 1$*. For the velocity field we put* $\boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1}) := \boldsymbol{v}_{0,h}$ *for* $n = 1$ *and with the approximation* $\boldsymbol{v}_{0,h}$ *of the initial value* $\boldsymbol{v}_0$*. Thus, it remains to define an approximation* $p_{0,h}$ *of the initial pressure* $p_0 := p(0)$*. This problem is more involved since the Navier–Stokes system does not provide an initial pressure. It is also impacted by the choice of the quadrature formula and the nodal interpolation properties of the temporal basis functions. A remedy based on Gauss quadrature in time and a post-processing for higher order pressure values*

*in the discrete time nodes is proposed in [1], [53]. A further remedy consists in the application of a discontinuous Galerkin approximation (cf. [1]) for the initial time step. In [83], a modification of the Crank–Nicholson scheme that is (up to quadrature) algebraically equivalent to the cGP(1) scheme is proposed by replacing the first two time steps with two implicit Euler steps. Regularity results for the Stokes equations, ensuring the optimal second order of convergence for the Crank–Nicholson scheme, are also studied in [129]. In Sec. 5.2.4 this topic, including computationally cheap resorts, are further studied.*

- *If the Navier–Stokes problem (4.1) is considered with (homogeneous) Dirichlet boundary conditions only, the unknown initial pressure $p_0 := p(0) \in L_0^2(\Omega)$ satisfies the boundary value problem (cf. [77, p. 376], [81])*

$$-\Delta p_0 = -\nabla \cdot \boldsymbol{f}(0) + \nabla \cdot ((\boldsymbol{v}_0 \cdot \nabla)\boldsymbol{v}_0) \qquad \text{in } \Omega\,, \tag{5.53a}$$

$$\nabla p_0 \cdot \boldsymbol{n} = \big(\boldsymbol{f}(0) + \nu \Delta \boldsymbol{v}_0\big) \cdot \boldsymbol{n} \qquad \text{on } \partial\Omega\,. \tag{5.53b}$$

*In this case, we put $p_{\tau,h|I_{n-1}}(t_{n-1}) := p_{0,h}$ for $n = 1$ where $p_{0,h} \in Q_h \cap L_0^2(\Omega)$ denotes a finite element approximation of the solution $p(0)$ to (5.53).*

#### 5.2.1.2 Galerkin–collocation time discretization

In this section the standard continuous Galerkin–Petrov method (cGP) is modified in the manner of Sec. 5.1.2: A collocation condition that involves the discrete solution's first derivative at the endpoint of the subinterval $I_n$ along with $C^1$-continuity constraints at the initial point of the subinterval $I_n$ is imposed, while on the other hand downsizing the test space of the variational equation (5.52). This principle is applied to the velocity as well as the pressure variable, with the perspective of achieving the accuracy of Galerkin schemes with reduced computational costs.

The presentation is restricted to the $\mathrm{GCC}^1(k)$ schemes, so a polynomial degree of $k \geq 3$ is assumed from now on. To introduce the Galerkin–collocation approximation, a Hermite quadrature formula and the corresponding interpolation operator is defined at first. Let $\hat{t}_1^{\mathrm{H}} = -1$, $\hat{t}_{k-1}^{\mathrm{H}} = 1$, and $\hat{t}_s^{\mathrm{H}}$, $s = 2, \ldots, k-2$, be the roots of the Jacobi polynomial on $\widehat{I} := [-1, 1]$ with degree $k-3$ associated to the weighting function $(1-\hat{t})^2(1+\hat{t})^2$. Let $\widehat{I}^{\mathrm{H}} : C^1\big(\widehat{I}; B\big) \to \mathbb{P}_k\big(\widehat{I}; B\big)$ denote the Hermite interpolation

operator with respect to point value and first derivative at both $-1$ and 1 as well as the point values at $\hat{t}_s^{\mathrm{H}}$, $s = 2, \dots, k-2$. By

$$\widehat{Q}^{\mathrm{H}}(\hat{g}) := \int_{-1}^{1} \widehat{I}^{\mathrm{H}}(\hat{g})(\hat{t}) \mathrm{d}\hat{t} \tag{5.54}$$

we define an Hermite-type quadrature on $[-1, 1]$ which can be written as

$$\widehat{Q}^{\mathrm{H}}(\hat{g}) = \widehat{\omega}_L \hat{g}'(-1) + \sum_{s=1}^{k-1} \widehat{\omega}_s \hat{g}(\hat{t}_s^{\mathrm{H}}) + \widehat{\omega}_R \hat{g}'(1)\,, \tag{5.55}$$

where all weights are non-zero. Using the affine mapping $T_n : \widehat{I} \to \overline{I}_n$ with $T_n(-1) = t_{n-1}$ and $T_n(1) = t_n$, we obtain

$$Q_n^{\mathrm{H}}(g) = \left(\frac{\tau_n}{2}\right)^2 \widehat{\omega}_L \mathrm{d}_t g(t_{n-1}^+) + \frac{\tau_n}{2} \sum_{s=1}^{k-1} \widehat{\omega}_s g(t_{n,s}^{\mathrm{H}}) + \left(\frac{\tau_n}{2}\right)^2 \widehat{\omega}_R \mathrm{d}_t g(t_n^-) \tag{5.56}$$

as Hermite-type quadrature formula on $I_n$, where $t_{n,s}^{\mathrm{H}} := T_n(\hat{t}_s^{\mathrm{H}})$, $s = 1, \dots, k-1$. We note that $Q_n^{\mathrm{H}}$ as defined by (5.56) integrates all polynomials up to degree $2k-3$ exactly, cf. [130]. Using $\widehat{I}^{\mathrm{H}}$ and $T_n$, the local Hermite interpolation on $I_n$ is given by

$$I_n^{\mathrm{H}} : C^1(\overline{I}_n; B) \to \mathbb{P}_k(\overline{I}_n; B)\,, \qquad v \mapsto \left(\widehat{I}^{\mathrm{H}}(v \circ T_n)\right) \circ T_n^{-1}\,. \tag{5.57}$$

Moreover, for all $n = 1, \dots, N$ we define the global Hermite interpolation $I_\tau^{\mathrm{H}} : C^1(\overline{I}; B) \to X_\tau^k(B)$ by means of

$$I_\tau^{\mathrm{H}} w|_{I_n} := I_n^{\mathrm{H}}(w|_{I_n})\,. \tag{5.58}$$

This operator is applied component wise to vector-valued functions.

The local problem of the Galerkin–collocation approach along with Nitsche's method for enforcing Dirichlet boundary conditions then reads as follows.

**Problem 5.5** (Local $I_n$ problem of $\mathrm{GCC}^1(k)$)**.** *Let $k \geq 3$ and an approximation $\boldsymbol{v}_{0,h} \in \boldsymbol{V}_h^{\mathrm{div}}$ of the initial value $\boldsymbol{v}_0$ be given. For $n = 1, \dots, N$ and given $\boldsymbol{u}_{\tau,h|I_{n-1}} \in (\mathbb{P}_k(I_{n-1}; V_h))^2 \times \mathbb{P}_k(I_{n-1}; Q_h)$ find $\boldsymbol{u}_{\tau,h|I_n} \in (\mathbb{P}_k(I_n; V_h))^2 \times \mathbb{P}_k(I_n; Q_h)$ such that*

$$\boldsymbol{v}_{\tau,h|I_n}(t_{n-1}) = \boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1})\,, \qquad \partial_t \boldsymbol{v}_{\tau,h|I_n}(t_{n-1}) = \partial_t \boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1})\,, \tag{5.59}$$

$$p_{\tau,h|I_n}(t_{n-1}) = p_{\tau,h|I_{n-1}}(t_{n-1})\,, \qquad \partial_t p_{\tau,h|I_n}(t_{n-1}) = \partial_t p_{\tau,h|I_{n-1}}(t_{n-1}) \tag{5.60}$$

*and*

$$\langle \partial_t \boldsymbol{v}_{\tau,h}(t_n), \boldsymbol{\psi}_h \rangle_{\Omega_f} + A_h((\boldsymbol{v}_{\tau,h}(t_n), p_{\tau,h}(t_n)), (\boldsymbol{\psi}_h, \xi_h)) = L_h(\psi_h; I_\tau^{\mathrm{H}} \boldsymbol{f}(t_n), I_\tau^{\mathrm{H}} \boldsymbol{g}(t_n)) \,, \tag{5.61}$$

*for all* $\boldsymbol{\phi}_h \in \boldsymbol{X}_h$ *as well as*

$$\int_{I_n} \langle \partial_t \boldsymbol{v}_{\tau,h}, \boldsymbol{\psi}_{\tau,h} \rangle_{\Omega_f} + A_h((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h})) \mathrm{d}t = \int_{I_n} L_h(\psi_{\tau,h}; I_\tau^{\mathrm{H}} \boldsymbol{f}, I_\tau^{\mathrm{H}} \boldsymbol{g}) \mathrm{d}t \,, \tag{5.62}$$

*for all* $\boldsymbol{\phi}_{\tau,h} \in (\mathbb{P}_{k-3}(I_n; V_h))^2 \times \mathbb{P}_{k-3}(I_n; Q_h)$.

**Remark 5.4.**

- *In Problem Problem 5.5 the variational equation (5.62) is combined with the collocation condition (5.61) at the endpoint* $t_n$ *of* $I_n$ *and the continuity constraints (5.59), (5.60).*
- *By definition (5.58) of the Hermite-type interpolation operator* $I_\tau^{\mathrm{H}}$*, we have that* $\partial_t^s I_\tau^{\mathrm{H}} \boldsymbol{f}(t_n) = \partial_t^s \boldsymbol{f}(t_n)$ *and* $\partial_t I_\tau^{\mathrm{H}} \boldsymbol{g}(t_n) = \partial_t \boldsymbol{g}(t_n)$ *for* $s \in \{0, 1\}$ *on the right-hand side of (5.61).*
- *The choice of the temporal basis (cf. Eqs. (5.16), that is induced by the definition of the Hermite-type quadrature formula (5.59) and the interpolation operator definition (5.58), allows a computationally cost-efficient implementation of the continuity constraints (5.59), (5.60). By these constraints the condition* $(\boldsymbol{v}_{\tau,h}, p_{\tau,h}) \in ((C(\overline{I}; V_h))^2 \times C(\overline{I}; Q_h)) \cap ((C^1(\overline{I}; V_h))^2 \times C^1(\overline{I}; Q_h))$ *and, thus, the* $C^1$ *regularity in time of* $\boldsymbol{v}_{\tau,h}$ *and* $p_{\tau,h}$ *is ensured.*
- *For the initial time interval* $I_1$*, i.e.* $n = 1$*, the continuity constraints (5.59), (5.60) are a source of trouble since we do not have an initial pressure* $p(0)$ *in the Navier–Stokes system (4.1). This holds similarly to the case of the cGP(k) approximation in time; cf. Remark 5.3. By the construction of the GCC¶(k) approach and its temporal basis (cf. Eqs. (5.69), (5.16)), even a spatial approximation of the time derivative of the initial pressure* $\partial_t p(0)$ *is needed now. An initial value for* $\partial_t \boldsymbol{v}(0)$ *and its' approximation can still be computed from the momentum equation (4.1b). Remedies for the initial time interval* $I_1$ *are sketched in Remark 5.3. However, this topic still deserves further research in the future. In our numerical convergence study presented in Sec. 5.2.3.1 the prescribed solution is used for providing the needed initial values. In the numerical study of flow around a cylinder presented*

*in Sec. 5.2.3.2 and 5.2.3.3 zero initial values are used. This is done to due to the specific problem setting.*

## 5.2.2 Algebraic system of Galerkin–collocation GCC$^1$(3) discretization in time and inf-sup stable finite approximation in space and its Newton linearization

In this section we derive the algebraic formulation of Problem 5.5. The Newton method is applied for solving the resulting nonlinear system of equations. The approach in this chapter differs from that in Chapter 3, since we assume a time-independent fluid domain $\Omega_f$ here. This allows an analytic computation of the coefficients that appear from the time integration.

To simplify the notation we restrict ourselves to the polynomial degree $k = 3$ for the discrete spaces $(\mathbb{P}_k(I_n; V_h))^2 \times \mathbb{P}_k(I_n; Q_h)$. The choice $k = 3$ is also used for the numerical experiments presented in Sec. 5.2.3. To derive the algebraic form of Problem 5.5, a Rothe type approach is applied to the system (4.1) by studying firstly in Sec. 5.2.2.1 the GCC$^1$(3) discretization in time of the system (4.1) along with its Newton linearization and then, doing the discretization in space by the Taylor-Hood family in Sec. 5.2.2.2. The discretization of the Nitsche terms hidden in the forms $a_\gamma$ and $b_\gamma$ of Problem 5.5 is derived separately in Sec. 5.2.2.3.

### 5.2.2.1 Semi-discretization in time by GCC$^1$(3) and Newton linearization

To simplify the presentation and enhance their confirmability, this is only done formally in the Banach space and without providing functions spaces. To keep the derivation as short as possible the usage of Nitsche's method is avoided in the first step. Instead homogeneous Dirichlet boundary conditions $\boldsymbol{g} = \mathbf{0}$ are assumed in this subsection. The extension that are necessary for Nitsche's method will introduce additional terms, that are sketched in Sec. 5.2.2.3.

The GCC$^1$(3) discretization in time of (4.1) reads as follows.

**Problem 5.6** ($\mathrm{GCC}^1(3)$ semidiscretization in time of (4.1))**.** *Let $k \geq 3$. For $n = 1, \ldots, N$ and given $(\boldsymbol{v}_{\tau|I_{n-1}}, p_{\tau|I_{n-1}}) \in (\mathbb{P}_k(I_{n-1}; V_0))^2 \times \mathbb{P}_k(I_{n-1}; W)$ find $(\boldsymbol{v}_{\tau|I_n}, p_{\tau|I_n}) \in (\mathbb{P}_k(I_n; V_0))^2 \times \mathbb{P}_k(I_n; Q)$ such that*

$$\boldsymbol{v}_{\tau|I_n}(t_{n-1}) = \boldsymbol{v}_{\tau|I_{n-1}}(t_{n-1})\,, \qquad \partial_t \boldsymbol{v}_{\tau|I_n}(t_{n-1}) = \partial_t \boldsymbol{v}_{\tau,h|I_{n-1}}(t_{n-1})\,, \tag{5.63}$$

$$p_{\tau|I_n}(t_{n-1}) = p_{\tau|I_{n-1}}(t_{n-1})\,, \qquad \partial_t p_{\tau|I_n}(t_{n-1}) = \partial_t p_{\tau|I_{n-1}}(t_{n-1})\,, \tag{5.64}$$

*and*

$$\partial_t \boldsymbol{v}_\tau(t_n) + (\boldsymbol{v}_\tau(t_n) \cdot \nabla)\boldsymbol{v}_\tau(t_n) - \nu \Delta \boldsymbol{v}_{\tau,h}(t_n) + \nabla p_\tau(t_n) = \boldsymbol{f}(t_n)\,, \tag{5.65}$$

$$\nabla \cdot \boldsymbol{v}_{\tau,h}(t_n) = 0 \tag{5.66}$$

*and, for all $\zeta_\tau \in \mathbb{P}_0(I_n; \mathbb{R})$,*

$$\int_{I_n} (\partial_t \boldsymbol{v}_\tau + (\boldsymbol{v}_\tau \cdot \nabla)\boldsymbol{v}_\tau - \nu \Delta \boldsymbol{v}_\tau + \nabla p_\tau) \cdot \zeta_\tau \mathrm{d}t = \int_{I_n} I_\tau^{\mathrm{H}} \boldsymbol{f} \cdot \zeta_\tau \mathrm{d}t\,, \tag{5.67}$$

$$\int_{I_n} \nabla \cdot \boldsymbol{v}_\tau \cdot \zeta_\tau \mathrm{d}t = 0\,. \tag{5.68}$$

In contrast to Problem 3.3 of Chapter 3 the spatial integration boundaries are not time-dependent and therefore, the time integrals in (5.67) and (5.68) can be computed exactly by the quadrature rule (5.56) with $k = 3$. For the derivation of an algebraic formulation, we firstly rewrite Problem 5.6 in terms of conditions about the coefficient functions of an expansion of the unknown variables $(\boldsymbol{v}_{\tau|I_n}, p_{\tau|I_n}) \in (\mathbb{P}_k(I_n; V))^2 \times \mathbb{P}_k(I_n; W_h)$ in temporal basis functions $\{\hat{\xi}_l\}_{l=0}^3$ of $\mathbb{P}_3(\hat{I}; \mathbb{R})$. With the notation $\boldsymbol{v}_\tau = (v_{\tau,1}, v_{\tau,2})$, such an expansion reads as

$$v_{\tau,i|I_n}(\boldsymbol{x}, t) = \sum_{l=0}^{3} v_{n,l,i}(\boldsymbol{x})\xi_l(t)\,, \quad \text{for } i \in \{1, 2\}\,, \quad p_{\tau|I_n}(\boldsymbol{x}, t) = \sum_{l=0}^{3} p_{n,l}(\boldsymbol{x})\xi_l(t)\,, \tag{5.69}$$

with coefficient functions $\boldsymbol{v}_{n,l} = (v_{n,l,1}, v_{n,l,2}) \in \boldsymbol{V}_0$ and $p_{n,l} \in Q$ and $t \in I_n$. We make use of the same time basis as in Eq. (5.17). We note that by (5.16) the expansions in (5.69) thus comprise the function values and time derivatives of $\boldsymbol{v}_{\tau|I_n}$ and $p_{\tau|I_n}$ at $t_{n-1}$

and $t_n$. By (5.16), the Hermite-type interpolation operator $I_\tau^{\mathrm{H}}$ defined by (5.57) and (5.58) then admits the explicit representation

$$\boldsymbol{g}_\tau := I_{\tau|I_n}\boldsymbol{g}(t) = \sum_{s=0}^{1} \tau_n^s \hat{\xi}_s(0) \underbrace{\partial_t^s \boldsymbol{g}_{|I_n}(t_{n-1})}_{=:\boldsymbol{g}_{n,s}} + \sum_{s=0}^{1} \tau_n^s \hat{\xi}_{s+2}(1) \underbrace{\partial_t^s \boldsymbol{g}|_{I_n}(t_n)}_{=:\boldsymbol{g}_{n,s+2}} . \tag{5.70}$$

In terms of the expansions (5.69) along with (5.16) we recast the conditions (5.63) and (5.64) as

$$\boldsymbol{v}_{n,0} = \boldsymbol{v}_{n-1,2}\,, \quad \boldsymbol{v}_{n,1} = \boldsymbol{v}_{n-1,3}\,, \quad p_{n,0} = p_{n-1,2}\,, \quad p_{n,1} = p_{n-1,3}\,. \tag{5.71}$$

Therefore, on each time interval $I_n$ we obtain four unknown coefficient functions from the previous time interval $I_{n-1}$ which amounts to computationally cheap copies of vectors on the fully discrete level. Next, integrating analytically the variational conditions (5.67) and (5.68) with the representations (5.69) of the unknowns implies that

$$\begin{aligned}
&\boldsymbol{v}_{n,0} + \boldsymbol{v}_{n,2} + \tau_n \bigg( \frac{13}{35}\nabla\boldsymbol{v}_{n,0}\boldsymbol{v}_{n,0} + \frac{11}{210}\nabla\boldsymbol{v}_{n,0}\boldsymbol{v}_{n,1} + \frac{9}{70}\nabla\boldsymbol{v}_{n,0}\boldsymbol{v}_{n,2} - \frac{13}{420}\nabla\boldsymbol{v}_{n,0}\boldsymbol{v}_{n,3} \\
&\quad + \frac{11}{210}\nabla\boldsymbol{v}_{n,1}\boldsymbol{v}_{n,0} + \frac{1}{105}\nabla\boldsymbol{v}_{n,1}\boldsymbol{v}_{n,1} + \frac{13}{420}\nabla\boldsymbol{v}_{n,1}\boldsymbol{v}_{n,2} - \frac{1}{140}\nabla\boldsymbol{v}_{n,1}\boldsymbol{v}_{n,3} + \frac{9}{70}\nabla\boldsymbol{v}_{n,2}\boldsymbol{v}_{n,0} \\
&\quad + \frac{13}{420}\nabla\boldsymbol{v}_{n,2}\boldsymbol{v}_{n,1} + \frac{13}{35}\nabla\boldsymbol{v}_{n,2}\boldsymbol{v}_{n,2} - \frac{11}{210}\nabla\boldsymbol{v}_{n,2}\boldsymbol{v}_{n,3} - \frac{13}{420}\nabla\boldsymbol{v}_{n,3}\boldsymbol{v}_{n,0} - \frac{1}{140}\nabla\boldsymbol{v}_{n,3}\boldsymbol{v}_{n,1} \\
&\quad - \frac{11}{210}\nabla\boldsymbol{v}_{n,3}\boldsymbol{v}_{n,2} + \frac{1}{105}\nabla\boldsymbol{v}_{n,3}\boldsymbol{v}_{n,0} \bigg) - \tau_n\nu \bigg( \frac{1}{2}\Delta\boldsymbol{v}_{n,0} + \frac{1}{12}\Delta\boldsymbol{v}_{n,1} + \frac{1}{2}\Delta\boldsymbol{v}_{n,2} \\
&\quad - \frac{1}{12}\Delta\boldsymbol{v}_{n,3} \bigg) + \tau_n \bigg( \frac{1}{2}\nabla p_{n,0} + \frac{1}{12}\nabla p_{n,1} + \frac{1}{2}\nabla p_{n,2} - \frac{1}{12}\nabla p_{n,3} \bigg) \\
&\qquad = \tau_n \bigg( \frac{1}{2}\nabla f_{n,0} + \frac{1}{12}\nabla f_{n,1} + \frac{1}{2}\nabla f_{n,2} - \frac{1}{12}\nabla f_{n,3} \bigg)
\end{aligned} \tag{5.72}$$

and

$$\tau_n \bigg( \frac{1}{2}\nabla\cdot\boldsymbol{v}_{n,0} + \frac{1}{12}\nabla\cdot\boldsymbol{v}_{n,1} + \frac{1}{2}\nabla\cdot\boldsymbol{v}_{n,2} - \frac{1}{12}\nabla\cdot\boldsymbol{v}_{n,3} \bigg) = 0\,. \tag{5.73}$$

In order to keep the notation as short as possible, the brackets in $(\nabla\boldsymbol{v}_{n,i})\,\boldsymbol{v}_{n,j}$ are omitted. Further it is assumed, that the differential operator just acts on the vector next to it only such that $(\nabla\boldsymbol{v}_{n,i})\,\boldsymbol{v}_{n,j} = \nabla\boldsymbol{v}_{n,i}\boldsymbol{v}_{n,j}$. Finally, by (5.69) along with (5.16) the collocation conditions(5.65) and (5.66) read as

$$\frac{1}{\tau_n}\boldsymbol{v}_{n,3} \; + \nabla\boldsymbol{v}_{n,2}\boldsymbol{v}_{n,2} \; - \nu\Delta\boldsymbol{v}_{n,2} + \nabla p_{n,2} = \boldsymbol{f}_{n,2}\,, \tag{5.74}$$

$$\nabla\cdot\boldsymbol{v}_{n,2} = 0\,. \tag{5.75}$$

On each subinterval $I_n$, Eqs. (5.71) to (5.75) form a nonlinear system in the Banach space. To solve this system of nonlinear equations (after an additional discretization in space; cf. Sec. 5.2.2.2) we use an inexact Newton method as in Sec. 3.3.

For the sake of completeness, the (non-damped) Newton iteration for the system (5.71) to (5.75) in the Banach space is briefly sketched here. For this, we let

$$\boldsymbol{x} \coloneqq (\boldsymbol{v}_{n,2}, p_{n,2}, \boldsymbol{v}_{n,3}, p_{n,3})^\top \tag{5.76}$$

denote the vector of remaining unknown coefficient functions of the expansions (5.69) under the identities (5.71) . Further, we subtract the right-hand sides of Eqs. (5.72) to (5.75) from its left-hand sides respectively and denote the resulting left-hand side functions by $\boldsymbol{q}(\boldsymbol{x})$ with its components $\{\boldsymbol{q}_1(\boldsymbol{x}), q_2(\boldsymbol{x}), \boldsymbol{q}_3(\boldsymbol{x}), q_4(\boldsymbol{x})\}$. Then, the Newton iteration reads as

$$\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{u}^k_{\tau,h}} \boldsymbol{d}^{k+1}_{\tau,h} = -\boldsymbol{q}(\boldsymbol{u}^k_{\tau,h}, \boldsymbol{\phi}_{\tau,h})\,. \tag{5.77}$$

for the correction $\boldsymbol{d}^{m+1}_{\tau,h} = (\boldsymbol{d}^{m+1}_{\boldsymbol{v}_{\tau,h}}, d^{m+1}_{p_{\tau,h}})^\top \coloneqq \boldsymbol{u}^{m+1}_{\tau,h} - \boldsymbol{u}^m_{\tau,h}$. Introducing the abbreviations

$$\boldsymbol{a}(\boldsymbol{x}) = \left(\frac{9}{70}\boldsymbol{v}_{n,0} + \frac{13}{420}\boldsymbol{v}_{n,1} + \frac{13}{35}\boldsymbol{v}_{n,2} - \frac{11}{210}\boldsymbol{v}_{n,3}\right)\,, \tag{5.78}$$

$$\boldsymbol{b}(\boldsymbol{x}) = \left(\frac{-13}{420}\boldsymbol{v}_{n,0} - \frac{1}{140}\boldsymbol{v}_{n,1} - \frac{11}{210}\boldsymbol{v}_{n,2} + \frac{1}{105}\boldsymbol{v}_{n,3}\right) \tag{5.79}$$

and defining

$$\begin{aligned}
&\boldsymbol{f}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}) \coloneqq \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}} - \frac{\tau_n\nu}{2}\Delta\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}} + \tau_n\nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\boldsymbol{a}(\boldsymbol{x}^k) + \tau_n\nabla\boldsymbol{a}(\boldsymbol{x}^k)\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\,,\\
&\boldsymbol{f}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) \coloneqq \frac{\tau_n\nu}{12}\Delta\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}} + \tau_n\nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}\boldsymbol{b}(\boldsymbol{x}^k) + \tau_n\nabla\boldsymbol{b}(\boldsymbol{x}^k)\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}\,,\\
&\boldsymbol{f}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}) \coloneqq \nu\Delta\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}} + \nabla\boldsymbol{v}^k_{n,2}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}} + \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\boldsymbol{v}^k_{n,2}\,, \quad \boldsymbol{f}_4(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) \coloneqq \frac{1}{\tau_n}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}\,,\\
&\boldsymbol{b}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}) \coloneqq -\frac{\tau_n}{2}\nabla\cdot\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\,, \quad \boldsymbol{b}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) \coloneqq -\frac{\tau_n}{12}\nabla\cdot\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}\,, \quad \boldsymbol{b}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) \coloneqq -\nabla\cdot\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}\,,\\
&\boldsymbol{b}_1^\top(d^{k+1}_{p_{n,2}}) \coloneqq -\frac{\tau_n}{2}\nabla d^{k+1}_{p_{n,2}}\,, \quad \boldsymbol{b}_2^\top(d^{k+1}_{p_{n,3}}) \coloneqq -\frac{\tau_n}{12}\nabla d^{k+1}_{p_{n,3}}\,, \quad \boldsymbol{b}_3^\top(d^{k+1}_{p_{n,3}}) \coloneqq -\nabla d^{k+1}_{p_{n,3}}\,,
\end{aligned} \tag{5.80}$$

we recast the system (5.77) as

$$\boldsymbol{f}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}) + \boldsymbol{f}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) + \boldsymbol{b}_1^\top(d^{k+1}_{p_{n,2}}) + \boldsymbol{b}_2^\top(d^{k+1}_{p_{n,3}}) = -\boldsymbol{q}_1(\boldsymbol{x}^k)\,, \tag{5.81a}$$

$$-\boldsymbol{b}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}) + \boldsymbol{b}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) = -q_2(\boldsymbol{x}^k)\,, \tag{5.81b}$$

$$\boldsymbol{f}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}) + \boldsymbol{f}_4(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) + \boldsymbol{b}_3^\top(d^{k+1}_{p_{n,3}}) = -\boldsymbol{q}_3(\boldsymbol{x}^k)\,, \tag{5.81c}$$

$$-\boldsymbol{b}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) = -q_4(\boldsymbol{x}^k)\,. \tag{5.81d}$$

In weak form, system (5.81) leads to the following problem to be solved in each Newton step.

**Problem 5.7** (Newton iteration of GCC[1](3) time discretization)**.** *Find corrections* $(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) \in \boldsymbol{V}_0^2$ *and* $(d^{k+1}_{p_{n,2}}, d^{k+1}_{p_{n,3}}) \in Q^2$ *such that for all* $\boldsymbol{\psi} \in \boldsymbol{V}_0^2$ *and* $\xi \in Q^2$ *there holds that*

$$\begin{aligned}\mathcal{F}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{\psi}) + \mathcal{F}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \boldsymbol{\psi}) + \mathcal{B}_1^\top(d^{k+1}_{p_{n,2}}, \boldsymbol{\psi}) + \mathcal{B}_2^\top(d^{k+1}_{p_{n,3}}, \boldsymbol{\psi})\\ = -\Big\langle \boldsymbol{q}_1(\boldsymbol{x}^k)\,, \boldsymbol{\psi}\Big\rangle + \frac{\tau_n\nu}{2}\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}(\boldsymbol{x}^k)\,, \boldsymbol{\psi}\Big\rangle_{\Gamma_o} - \frac{\tau_n\nu}{12}\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}(\boldsymbol{x}^k)\,, \boldsymbol{\psi}\Big\rangle_{\Gamma_o}\,,\end{aligned} \tag{5.82a}$$

$$-\mathcal{B}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \xi) - \mathcal{B}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \xi) = -\Big\langle q_2(\boldsymbol{x}^k)\,, \xi\Big\rangle\,, \tag{5.82b}$$

$$\mathcal{F}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{\psi}) + \mathcal{F}_4(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \boldsymbol{\psi}) + \mathcal{B}_3^\top(d^{k+1}_{p_{n,3}}, \boldsymbol{\psi}) = -\Big\langle \boldsymbol{q}_3(\boldsymbol{x}^k)\,, \boldsymbol{\psi}\Big\rangle - \nu\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}(\boldsymbol{x}^k)\,, \boldsymbol{\psi}\Big\rangle_{\Gamma_o}\,, \tag{5.82c}$$

$$-\mathcal{B}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \xi) = -\Big\langle q_4(\boldsymbol{x}^k)\,, \xi\Big\rangle\,, \tag{5.82d}$$

*with* $\partial_{\boldsymbol{n}}\boldsymbol{w} = \nabla\boldsymbol{w}\cdot\boldsymbol{n}$ *and*

$$\begin{aligned}\mathcal{F}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{\psi}) &\coloneqq \Big\langle \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{\psi}\Big\rangle + \frac{\tau_n\nu}{2}\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \nabla\boldsymbol{\psi}\Big\rangle\\ &\qquad + \tau_n\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\boldsymbol{a}(\boldsymbol{x}^k), \boldsymbol{\psi}\Big\rangle + \tau_n\Big\langle \nabla\boldsymbol{a}(\boldsymbol{x}^k)\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{\psi}\Big\rangle\,,\\ \mathcal{F}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \boldsymbol{\psi}) &\coloneqq -\frac{\tau_n\nu}{12}\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \nabla\boldsymbol{\psi}\Big\rangle + \tau_n\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}\boldsymbol{b}(\boldsymbol{x}^k), \boldsymbol{\psi}\Big\rangle + \tau_n\Big\langle \nabla\boldsymbol{b}(\boldsymbol{x}^k)\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \boldsymbol{\psi}\Big\rangle\,,\\ \mathcal{F}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{\psi}) &\coloneqq \nu\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \nabla\boldsymbol{\psi}\Big\rangle + \Big\langle \nabla\boldsymbol{v}^k_{n,2}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{\psi}\Big\rangle + \Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\boldsymbol{v}^k_{n,2}, \boldsymbol{\psi}\Big\rangle\,,\\ \mathcal{F}_4(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \boldsymbol{\psi}) &\coloneqq \frac{1}{\tau_n}\Big\langle \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \boldsymbol{\psi}\Big\rangle\end{aligned} \tag{5.83}$$

*and*

$$\begin{aligned}
\mathcal{B}_1(\boldsymbol{d}_{\boldsymbol{v}_{n,2}}^{k+1}, \xi) &:= -\frac{\tau_n}{2}\left\langle \nabla \cdot \boldsymbol{d}_{\boldsymbol{v}_{n,2}}^{k+1}, \xi \right\rangle, \quad \mathcal{B}_1^\top(d_{p_{n,2}}^{k+1}, \boldsymbol{\psi}) := \frac{\tau_n}{2}\left\langle d_{p_{n,2}}^{k+1}, \nabla \cdot \boldsymbol{\psi} \right\rangle, \\
\mathcal{B}_2(\boldsymbol{d}_{\boldsymbol{v}_{n,3}}^{k+1}, \xi) &:= \frac{\tau_n}{12}\left\langle \nabla \cdot \boldsymbol{d}_{\boldsymbol{v}_{n,3}}^{k+1}, \xi \right\rangle, \quad \mathcal{B}_2^\top(d_{p_{n,3}}^{k+1}, \boldsymbol{\psi}) := -\frac{\tau_n}{12}\left\langle d_{p_{n,3}}^{k+1}, \nabla \cdot \boldsymbol{\psi} \right\rangle, \\
\mathcal{B}_3(\boldsymbol{d}_{\boldsymbol{v}_{n,3}}^{k+1}, \xi) &:= -\left\langle \nabla \cdot \boldsymbol{d}_{\boldsymbol{v}_{n,3}}^{k+1}, \xi \right\rangle, \quad \mathcal{B}_3^\top(d_{p_{n,3}}^{k+1}, \boldsymbol{\psi}) := \left\langle d_{p_{n,3}}^{k+1}, \nabla \cdot \boldsymbol{\psi} \right\rangle.
\end{aligned} \tag{5.84}$$

#### 5.2.2.2 Fully discrete system with inf-sup stable elements

In this subsection we briefly present the discretization in space of the system (5.82) of Problem 5.7 in the pair $V_h^0 \times Q_h$ of inf-sup stable finite element spaces. For our computations presented in Sec. 5.2.3 we used the $\mathcal{Q}_p$–$\mathcal{Q}_{p-1}, p \geq 2$ pair of the well-known Taylor–Hood family. Due to their inf-sup stability a stabilization of the discretization is not required, as long as the Reynolds number of the fluid flow is assumed to be small such that no convection-dominance occurs. In the case of higher Reynolds numbers, an additional stabilization of the discretization becomes indispensible; cf. [77, Sec. 5.3 and 5.4] and the references therein. However, this is beyond the scope of interest in this work and left as a work for the future.

For the space discretization, let $\{\boldsymbol{\psi}_j\}_{j=1}^J \subset \boldsymbol{V}_h^0$ and $\{\phi_m\}_{m=1}^M \subset Q_h$ denote a nodal Lagrangian basis of $\boldsymbol{V}_h^0$ and $Q_h$, respectively. Then, the fully discrete unknowns admit the representations

$$\boldsymbol{v}_{\tau,h|I_n}(\boldsymbol{x}, t) = \sum_{l=0}^{3}\sum_{j=1}^{J} v_{n,l,j}\boldsymbol{\psi}_j(\boldsymbol{x})\xi_l(t)\,, \quad p_{\tau,h|I_n}(\boldsymbol{x}, t) = \sum_{l=0}^{3}\sum_{m=1}^{M} p_{n,l,m}\phi_m(\boldsymbol{x})\xi_l(t)\,, \tag{5.85}$$

for $(\boldsymbol{x}, t) \in \Omega \times \overline{I_n}$ with the unknown coefficient vector $\boldsymbol{v}_{n,l} := (v_{n,l,j})_{j=1}^{J} \in \mathbb{R}^J$ and $\boldsymbol{p}_{n,l} := (p_{n,l,m})_{m=1}^{M} \in \mathbb{R}^M$, for $l = 0, \ldots, 3$, as the degrees of freedom. Next, we define

$$\boldsymbol{F}_1 := \big(\mathcal{F}_1(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\big)_{i,j=1}^{J}, \quad \boldsymbol{F}_2 := \big(\mathcal{F}_2(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\big)_{i,j=1}^{J}, \quad \boldsymbol{F}_3 := \big(\mathcal{F}_3(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\big)_{i,j=1}^{J}, \tag{5.86a}$$

$$\boldsymbol{F}_4 := \big(\mathcal{F}_4(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\big)_{i,j=1}^{J}, \quad \boldsymbol{B}_1 := (\mathcal{B}_1(\boldsymbol{\psi}_i, \phi_l))_{i,m=1}^{J,M}, \quad \boldsymbol{B}_1^\top := \big(\mathcal{B}_1^\top(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\big)_{i,j=1}^{J}, \tag{5.86b}$$

$$\boldsymbol{B}_2 := (\mathcal{B}_2(\boldsymbol{\psi}_i, \phi_m))_{i,m=1}^{J,M}, \quad \boldsymbol{B}_2^\top := \big(\mathcal{B}_2^\top(\boldsymbol{\psi}_i, \boldsymbol{\phi}_j)\big)_{i,j=1}^{J}, \quad \boldsymbol{B}_3 := (\mathcal{B}_3(\boldsymbol{\psi}_i, \phi_m))_{i,m=1}^{J,M}, \tag{5.86c}$$

$$\boldsymbol{B}_3^\top := \big(\mathcal{B}_3^\top(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\big)_{i,j=1}^{J}. \tag{5.86d}$$

Solving Problem 5.7 in the finite dimensional subspaces $\boldsymbol{V}_h^0$ and $Q_h$ of $\boldsymbol{V}_0$ and $Q$, respectively, leads to the following problem to be solved within each Newton iteration of a time step.

**Problem 5.8** (Newton iteration of GCC[1](3) time discretization and inf-sup stable elements for space discretization)**.** *Find corrections* $(\boldsymbol{d}_{\boldsymbol{v}_{n,2}}^{k+1}, \boldsymbol{d}_{\boldsymbol{v}_{n,3}}^{k+1}) \in \mathbb{R}^{2J}$ *and* $(d_{p_{n,2}}^{k+1}, d_{p_{n,3}}^{k+1}) \in \mathbb{R}^{2M}$ *such that*

$$\boldsymbol{S}\boldsymbol{d}_{\boldsymbol{x}}^{k+1} = \boldsymbol{d}(\boldsymbol{x}^k), \tag{5.87}$$

*where* $\boldsymbol{d}_{\boldsymbol{x}}^{k+1} := (\boldsymbol{d}_{\boldsymbol{v}_{n,2}}^{k+1}, d_{p_{n,2}}^{k+1}, \boldsymbol{d}_{\boldsymbol{v}_{n,3}}^{k+1}, d_{p_{n,3}}^{k+1})^\top \in \mathbb{R}^{2(J+M)}$ *denotes the vector of Newton corrections for the degrees of freedom and* $\boldsymbol{d}(\boldsymbol{x}^k)$ *is the fully discrete counterpart of the terms on the right-hand side of* (5.82)*. The block system matrix* $\boldsymbol{S}$ *in* (5.87) *is given by*

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{F}_1 & \boldsymbol{B}_1^\top & \boldsymbol{F}_2 & \boldsymbol{B}_2^\top \\ -\boldsymbol{B}_1 & \boldsymbol{0} & -\boldsymbol{B}_2 & \boldsymbol{0} \\ \boldsymbol{F}_3 & \boldsymbol{B}_3^\top & \boldsymbol{F}_4 & \boldsymbol{0} \\ -\boldsymbol{B}_3 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}. \tag{5.88}$$

Since we use the family of inf-sup stable Taylor-Hood element here, the resulting system matrix (5.88) comprises non-quadratic sub-matrices $\boldsymbol{B}$. The sparsity pattern of $\boldsymbol{S}$ is illustrated in Fig. 5.6. The system matrix $\boldsymbol{S}$ consists of three submatrices $\boldsymbol{S}_i$ of the common structure

$$\boldsymbol{S}_i = \begin{pmatrix} \boldsymbol{F}_i & \boldsymbol{B}_i^\top \\ -\boldsymbol{B}_i & \boldsymbol{0} \end{pmatrix} \tag{5.89}$$

and an additional block of $\boldsymbol{F}_4$ together with blocks of zero entries. Due to the collocation conditions at the final time points $t_n$ of the subinterval $I_n$ the matrices $\boldsymbol{B}_4^\top$ and a $-\boldsymbol{B}_4$ do not arise in the right lower block of $\boldsymbol{S}$ such that in (5.88) a sparser matrix structure is obtained compared to a pure variational approach. In order to solve Eq. (5.87) we use a (parallel) GMRES solver with a (preliminary) block preconditioner that is motivated by an approach presented in [38]. In (5.88), we consider each of the three submatrices $S_i$ as an uncoupled block of the structure (5.89). For each of these blocks we then use a Schur complement preconditioner with an approximation of the mass matrix of the pressure variable. This results in reasonable numbers of iterations for the GMRES solver for two-dimensional problems of medium size but is far from being acceptable for three-dimensional or large scale problems. The design of a more efficient and robust preconditioner that is tailored to the specific structure of the matrix $\boldsymbol{S}$ in (5.88) or a multigrid approach remains as a work for the future.

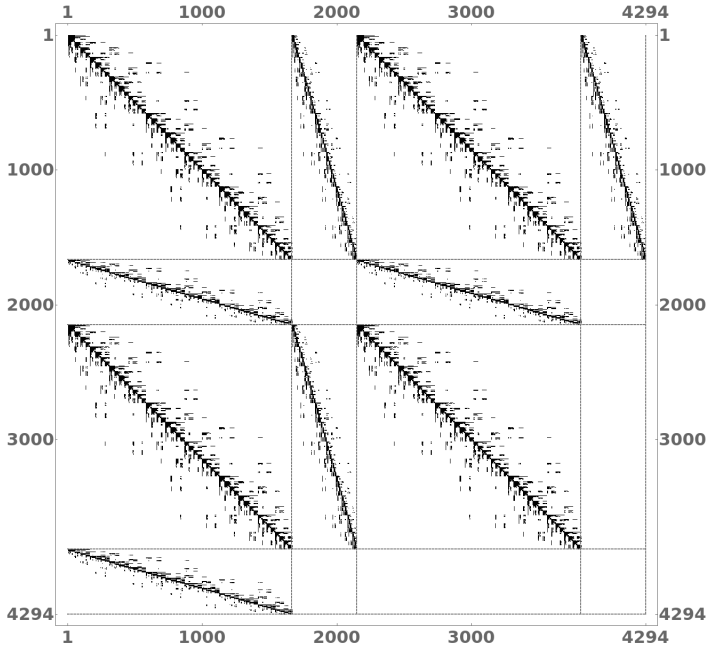



Figure 5.6: Sparsity pattern of $\boldsymbol{S}$ in (5.88) for the discrete problem of Sec. 5.2.3.1, using $\mathcal{Q}_4$–$\mathcal{Q}_3$ Taylor-Hood elements.

### 5.2.2.3 Nitsche's method for boundary conditions of Dirichlet type

In this subsection modifications to be made to Problem 5.7 and 5.8, respectively, for the application of Nitsche's method to enforce Dirichlet boundary conditions; cf. Problem 5.5. In contrast to Problems 5.7 and 5.8, the Dirichlet boundary conditions are

now ensured by augmenting the weak formulation with additional line integrals (surface integrals in three space dimensions); cf. Problem 5.5. The structure of the underlying system matrix will stay the same as in Eq. (5.88), but the additional terms result in a more complex notation than the one in Sec. 5.2.2.2.

One of the appreciable advantages of using Nitsche's method over the standard implementation of Dirichlet boundary conditions is, that this method can be used in conjunction with immersed boundary methods.

In contrast to Sec. 5.2.2.1, the continuous solution and test space is now $\boldsymbol{X} = \boldsymbol{V} \times Q$ instead of $\boldsymbol{X}_0 = \boldsymbol{V}_0 \times Q$. For the weak form of the time discrete Newton linearization (5.81) integration by parts then yields that

$$-\nu\big\langle \Delta \boldsymbol{d}_{\boldsymbol{v}_\tau}^{k+1}, \boldsymbol{\psi}\big\rangle = \nu\big\langle \nabla \boldsymbol{d}_{\boldsymbol{v}_\tau}^{k+1}, \nabla\boldsymbol{\psi}\big\rangle - \nu\big\langle \partial_{\boldsymbol{n}} \boldsymbol{d}_{\boldsymbol{v}_\tau}^{k+1}, \boldsymbol{\psi}\big\rangle_\Gamma\,, \tag{5.90a}$$

$$\big\langle \nabla d_{p_\tau}^{k+1}, \boldsymbol{\psi}\big\rangle = -\big\langle d_{p_\tau}^{k+1}, \nabla\cdot\boldsymbol{\psi}\big\rangle + \big\langle d_{p_\tau}^{k+1}\boldsymbol{n}, \boldsymbol{\psi}\big\rangle_\Gamma\,, \tag{5.90b}$$

where $\langle\cdot,\cdot\rangle_\Gamma$ denotes the inner product of $L^2(\partial\Omega)$ and $\boldsymbol{g}_\tau$ is defined by means of the Hermite-type interpolation (5.70). To preserve the symmetry properties of the continuous system, the forms (cf. Problem 5.5)

$$S_v(\boldsymbol{\psi}) := \nu\big\langle \partial_{\boldsymbol{n}}\boldsymbol{\psi}, \boldsymbol{v}_\tau^k + \boldsymbol{d}_{\boldsymbol{v}_\tau}^{k+1} - \boldsymbol{g}_\tau\big\rangle_{\Gamma_D}\,, \tag{5.91a}$$

$$S_p(\xi) := \big\langle \xi\boldsymbol{n}, \boldsymbol{v}_\tau^k + \boldsymbol{d}_{\boldsymbol{v}_\tau}^{k+1} - \boldsymbol{g}_\tau\big\rangle_{\Gamma_D}\,, \tag{5.91b}$$

are added on the right-hand side of (5.90a) and (5.90b), respectively, to the viscous and pressure part. Finally, we add the penalty terms (cf. Problem 5.5)

$$P_{\boldsymbol{g}_\tau}(\boldsymbol{\psi}) := \underbrace{\frac{\eta_1}{h}\nu\big\langle \boldsymbol{v}_\tau^k + \boldsymbol{d}_{\boldsymbol{v}_\tau}^{k+1} - \boldsymbol{g}_\tau, \boldsymbol{\psi}\big\rangle_{\Gamma_D}}_{\text{viscous penalty}} + \underbrace{\frac{\eta_2}{h}\big\langle (\boldsymbol{v}_\tau^k + \boldsymbol{d}_{\boldsymbol{v}_\tau}^{k+1} - \boldsymbol{g}_\tau)\cdot\boldsymbol{n}, \boldsymbol{\psi}\cdot\boldsymbol{n}\big\rangle_{\Gamma_D}}_{\text{penalty along } \boldsymbol{n}}\,. \tag{5.92}$$

For the test space $\boldsymbol{X} = \boldsymbol{V}\times Q$, such that $(\boldsymbol{\psi}, \xi) \in \boldsymbol{V}\times Q$, the integrals over the Dirichlet part $\Gamma_D$ of the boundary $\Gamma = \Gamma_D \cup \Gamma_o$ no longer vanish. For viscous-dominated flow, the additional terms in (5.92) enforce in weak form the boundary condition $\boldsymbol{v} - \boldsymbol{g} = \boldsymbol{0}$ on the Dirichlet part of the boundary. For convection-dominated flow with a small viscosity parameter $\nu$ this enforcement is weakened in the first term on the right-hand side of (5.92). However, in the inviscid limit $\nu \to 0$, the condition $(\boldsymbol{v} - \boldsymbol{g})\cdot\boldsymbol{n} = 0$ is still imposed weakly by the second of the terms on the right-hand side of (5.92) such that the normal component of the Dirichlet boundary condition is preserved in the limit case. For our computations shown in Sec. 5.2.3 we put $\eta_1 = \eta_2 = 35$. For a more

refined analysis of these parameters we refer to [24], [131]. Changing the sign of the symmetric term (5.91a) generates a non-symmetric formulation. Current results (cf. [24], [62], [132]) show that, on the one hand, this reduces the sensitivity with respect of the choice of the penalty parameters and might even allow for a parameter free penalty variant, but, on the other hand, it results in a non-symmetric structure of the underlying elliptic sub-problems, which complicates the design efficient linear solver and preconditioning techniques.

Instead of Problem 5.7 we then get following Newton iteration for the GGC$^1$(3) semidiscretization in time of the Navier–Stokes system (4.1) along with Nitsche's method for enforcing Dirichlet-type boundary conditions. Due to the cumbersome derivation of the equations and the innovation of the GGC$^1$(3) approach all formulas are explicitly given here in order to facilitate its application and implementation and enhance the confirmability of this work.

**Problem 5.9** (Newton iteration of GCC$^1$(3) time discretization with Nitsche's method)**.** *Find corrections* $(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}) \in \boldsymbol{V}^2$ *and* $(d^{k+1}_{p_{n,2}}, d^{k+1}_{p_{n,3}}) \in Q^2$ *such that for all* $\boldsymbol{\phi} = (\boldsymbol{\psi}, \xi) \in \boldsymbol{V}^2 \times Q^2$ *there holds that*

$$
\begin{aligned}
&\widetilde{\mathcal{F}}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \boldsymbol{\phi}) + \widetilde{\mathcal{F}}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \boldsymbol{\phi}) + \widetilde{\mathcal{B}}_1^{\top}(d^{k+1}_{p_{n,2}}, \boldsymbol{\phi}) + \widetilde{\mathcal{B}}_2^{\top}(d^{k+1}_{p_{n,3}}, \boldsymbol{\phi}) \\
&\quad = -\Big\langle \boldsymbol{q}_1(\boldsymbol{x}^k), \boldsymbol{\psi} \Big\rangle + \frac{\tau_n \nu}{2} \Big\langle \partial_{\boldsymbol{n}}(\boldsymbol{v}^k_{n,2}(\boldsymbol{x}^k), \boldsymbol{\psi} \Big\rangle_{\Gamma_o} - \frac{\tau_n \nu}{12} \Big\langle \partial_{\boldsymbol{n}}(\boldsymbol{v}^k_{n,3}(\boldsymbol{x}^k), \boldsymbol{\psi} \Big\rangle_{\Gamma_0} \\
&\qquad + \frac{\tau_n \nu}{2} \Big\langle \partial_{\boldsymbol{n}} \boldsymbol{\psi}, \boldsymbol{v}^k_{n,2} - \boldsymbol{g}_{n,2} \Big\rangle_{\Gamma_D} - \frac{\eta_1 \tau_n}{2h} \nu \Big\langle \boldsymbol{v}^k_{n,2} - \boldsymbol{g}_{n,2}, \boldsymbol{\psi} \Big\rangle_{\Gamma_D} \\
&\quad - \frac{\eta_2 \tau_n}{2h} \Big\langle (\boldsymbol{v}^k_{n,2} - \boldsymbol{g}_{n,2}) \cdot \boldsymbol{n}, \boldsymbol{\psi} \cdot \boldsymbol{n} \Big\rangle_{\Gamma_D} - \frac{\tau_n \nu}{12} \Big\langle \partial_{\boldsymbol{n}} \boldsymbol{\psi}, \boldsymbol{v}^k_{n,3} - \boldsymbol{g}_{n,3} \Big\rangle_{\partial\Gamma} \\
&\qquad + \frac{\eta_1 \tau_n}{12h} \nu \Big\langle \boldsymbol{v}^k_{n,3} - \boldsymbol{g}_{n,3}, \boldsymbol{\psi} \Big\rangle_{\partial\Gamma_D} + \frac{\eta_2 \tau_n}{12h} \Big\langle (\boldsymbol{v}^k_{n,3} - \boldsymbol{g}_{n,3}) \cdot \boldsymbol{n}, \boldsymbol{\psi} \cdot \boldsymbol{n} \Big\rangle_{\Gamma_D},
\end{aligned}
$$

$$
\begin{aligned}
&- \widetilde{\mathcal{B}}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \xi) - \widetilde{\mathcal{B}}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \xi) \\
&\qquad = -\Big\langle q_2(\boldsymbol{x}^k), \xi \Big\rangle - \frac{\tau_n}{2} \Big\langle \xi \boldsymbol{n}, \boldsymbol{v}^k_{n,2} - \boldsymbol{g}_{n,2} \Big\rangle_{\Gamma_D} + \frac{\tau_n}{12} \Big\langle \xi \boldsymbol{n}, \boldsymbol{v}^k_{n,3} - \boldsymbol{g}_{n,3} \Big\rangle_{\Gamma_D},
\end{aligned}
$$

$$\widetilde{\mathcal{F}}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\boldsymbol{\phi})+\mathcal{F}_4(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}},\boldsymbol{\phi})+\widetilde{\mathcal{B}}_3^{\top}(d^{k+1}_{p_{n,3}},\boldsymbol{\phi})$$
$$=-\Big\langle \boldsymbol{q}_3(\boldsymbol{x}^k)\,,\boldsymbol{\psi}\Big\rangle-\nu\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{v}_{n,2}(\boldsymbol{x}^k)\,,\boldsymbol{\psi}\Big\rangle_{\Gamma_o}$$
$$+\nu\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{\psi},\boldsymbol{v}^k_{n,2}-\boldsymbol{g}_{n,2}\Big\rangle_{\Gamma_D}-\frac{\eta_1}{h}\nu\Big\langle \boldsymbol{v}^k_{n,2}-\boldsymbol{g}_{n,2},\boldsymbol{\psi}\Big\rangle_{\Gamma_D}-\frac{\eta_2}{h}\Big\langle (\boldsymbol{v}^k_{n,2}-\boldsymbol{g}_{n,2})\cdot\boldsymbol{n},\boldsymbol{\psi}\cdot\boldsymbol{n}\Big\rangle_{\Gamma_D},$$

$$-\widetilde{\mathcal{B}}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}},\xi)=-\Big\langle q_4(\boldsymbol{x}^k)\,,\xi\Big\rangle-\Big\langle \xi\boldsymbol{n},\boldsymbol{v}^k_{n,3}-\boldsymbol{g}_{n,3}\Big\rangle_{\Gamma_D},$$

*where the semi-linear and linear forms are defined by*

$$\widetilde{\mathcal{F}}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\boldsymbol{\phi}):=\Big\langle \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\boldsymbol{\psi}\Big\rangle+\frac{\tau_n\nu}{2}\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\nabla\boldsymbol{\psi}\Big\rangle+\tau_n\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\boldsymbol{a}(\boldsymbol{x}^k),\boldsymbol{\psi}\Big\rangle$$
$$+\tau_n\Big\langle \nabla\boldsymbol{a}(\boldsymbol{x}^k)\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\boldsymbol{\psi}\Big\rangle-\frac{\tau_n\nu}{2}\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\boldsymbol{\psi}\Big\rangle_{\Gamma_D}-\frac{\tau_n\nu}{2}\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{\psi},\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\Big\rangle_{\Gamma_D}$$
$$+\frac{\eta_1\tau_n}{2h}\nu\Big\langle \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\boldsymbol{\psi}\Big\rangle_{\Gamma_D}+\frac{\eta_2\tau_n}{2h}\Big\langle \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\cdot\boldsymbol{n},\boldsymbol{\psi}\cdot\boldsymbol{n}\Big\rangle_{\Gamma_D},\quad(5.93)$$

$$\widetilde{\mathcal{F}}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}},\boldsymbol{\phi}):=-\frac{\tau_n\nu}{12}\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}},\nabla\boldsymbol{\psi}\Big\rangle+\tau_n\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}\boldsymbol{b}(\boldsymbol{x}^k),\boldsymbol{\psi}\Big\rangle+\tau_n\Big\langle \nabla\boldsymbol{b}(\boldsymbol{x}^k)\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}},\boldsymbol{\psi}\Big\rangle$$
$$+\frac{\tau_n\nu}{12}\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}},\boldsymbol{\psi}\Big\rangle_{\Gamma_D}+\frac{\tau_n\nu}{12}\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{\psi},\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}\Big\rangle_{\Gamma_D}-\frac{\eta_1\tau_n}{12h}\nu\Big\langle \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}},\boldsymbol{\psi}\Big\rangle_{\Gamma_D}$$
$$-\frac{\eta_2\tau_n}{12h}\Big\langle \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}\cdot\boldsymbol{n},\boldsymbol{\psi}\cdot\boldsymbol{n}\Big\rangle_{\Gamma_D},\quad(5.94)$$

$$\widetilde{\mathcal{F}}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\boldsymbol{\phi}):=\nu\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\nabla\boldsymbol{\psi}\Big\rangle+\Big\langle \nabla\boldsymbol{v}^k_{n,2}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\boldsymbol{\psi}\Big\rangle+\Big\langle \nabla\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\boldsymbol{v}^k_{n,2},\boldsymbol{\psi}\Big\rangle$$
$$-\nu\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}},\boldsymbol{\psi}\Big\rangle_{\Gamma_D}-\nu\Big\langle \partial_{\boldsymbol{n}}\boldsymbol{\psi},\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\Big\rangle_{\Gamma_D}+\frac{\eta_1}{h}\nu\Big\langle \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}},\boldsymbol{\psi}\Big\rangle_{\Gamma_D}$$
$$+\frac{\eta_2}{h}\Big\langle \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}\cdot\boldsymbol{n},\boldsymbol{\psi}\cdot\boldsymbol{n}\Big\rangle_{\Gamma_D}\quad(5.95)$$

*and, with* $\mathcal{B}_i$ *as well as* $\mathcal{B}_i^\top$, $i = 1 \dots 3$, *as defined in* (5.84),

$$\begin{aligned}
\widetilde{\mathcal{B}}_1(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}}, \xi) &\coloneqq \mathcal{B}_1 + \frac{\tau_n}{2}\left\langle \xi \boldsymbol{n}, \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,2}} \right\rangle_{\Gamma_D}, \quad \widetilde{\mathcal{B}}_1^\top(d^{k+1}_{p_{n,2}}, \boldsymbol{\phi}) \coloneqq \mathcal{B}_1^\top - \frac{\tau_n}{2}\left\langle d^{k+1}_{p_{n,2}} \boldsymbol{n}, \boldsymbol{\psi} \right\rangle_{\Gamma_D}, \\
\widetilde{\mathcal{B}}_2(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \xi) &\coloneqq \mathcal{B}_2 - \frac{\tau_n}{12}\left\langle \xi \boldsymbol{n}, \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}} \right\rangle_{\Gamma_D}, \quad \widetilde{\mathcal{B}}_2^\top(d^{k+1}_{p_{n,3}}, \boldsymbol{\phi}) \coloneqq \mathcal{B}_2^\top + \frac{\tau_n}{12}\left\langle d^{k+1}_{p_{n,3}} \boldsymbol{n}, \boldsymbol{\psi} \right\rangle_{\Gamma_D}, \\
\widetilde{\mathcal{B}}_3(\boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}}, \xi) &\coloneqq \mathcal{B}_3 + \left\langle \xi \boldsymbol{n}, \boldsymbol{d}^{k+1}_{\boldsymbol{v}_{n,3}} \right\rangle_{\Gamma}, \qquad \widetilde{\mathcal{B}}_3^\top(d^{k+1}_{p_{n,3}}, \boldsymbol{\phi}) \coloneqq \mathcal{B}_3^\top - \left\langle d^{k+1}_{p_{n,3}} \boldsymbol{n}, \boldsymbol{\psi} \right\rangle_{\Gamma_D}.
\end{aligned} \tag{5.96}$$

The fully discrete counterpart of Problem 5.9 is obtained along the lines of Sec. 5.2.2.2 with

$$\begin{aligned}
\widetilde{\boldsymbol{F}}_1 &\coloneqq \left(\widetilde{\mathcal{F}}_1(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\right)_{i,j=1}^{J}, \quad \widetilde{\boldsymbol{F}}_2 \coloneqq \left(\widetilde{\mathcal{F}}_2(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\right)_{i,j=1}^{J}, \quad \widetilde{\boldsymbol{F}}_3 \coloneqq \left(\widetilde{\mathcal{F}}_3(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\right)_{i,j=1}^{J}, \\
\widetilde{\boldsymbol{B}}_1 &\coloneqq \left(\widetilde{\mathcal{B}}_1(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\right)_{i,j=1}^{J}, \quad \widetilde{\boldsymbol{B}}_2 \coloneqq \left(\widetilde{\mathcal{B}}_2(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\right)_{i,j=1}^{J}, \quad \widetilde{\boldsymbol{B}}_3 \coloneqq \left(\widetilde{\mathcal{B}}_3(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\right)_{i,j=1}^{J}, \\
\widetilde{\boldsymbol{B}}_1^\top &\coloneqq \left(\widetilde{\mathcal{B}}_1^\top(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\right)_{i,j=1}^{J}, \quad \widetilde{\boldsymbol{B}}_2^\top \coloneqq \left(\widetilde{\mathcal{B}}_2^\top(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\right)_{i,j=1}^{J}, \quad \widetilde{\boldsymbol{B}}_3^\top \coloneqq \left(\widetilde{\mathcal{B}}_3^\top(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)\right)_{i,j=1}^{J},
\end{aligned} \tag{5.97}$$

replacing the corresponding quantities (5.86). The resulting block system of the Newton iteration has the same sparsity pattern as in (5.88), but is now based on the matrices (5.97).

### 5.2.3 Numerical experiments

In this section we study numerically the GCC$^1$(3) approach along with Nitsche's method for Dirichlet boundary conditions, presented before in Sec. 5.2.2.3. Firstly, this is done by a numerical convergence study. A study of the condition number of the arising linear algebraic systems is also included. Secondly, the GCC$^1$(3) approach is applied to one of the popular benchmark problems proposed in [46] of flow around a cylinder. The drag and lift coefficient are computed as goal quantities of physical interest. For the sake of comparison, calculations with the standard continuous Petrov–Galerkin method of piecewise linear polynomials in time, referred to as cGP(1), are also presented. For the implementation we used the deal.II finite element library [57] along with the Trilinos library [94] for parallel computations on multiple processors.

#### 5.2.3.1 Convergence study

For the solution $\{\boldsymbol{v}, p\}$ of the Navier–Stokes system (4.1) and its fully discrete GCC$^1$(3) approximation $\{\boldsymbol{v}_{\tau,h}, p_{\tau,h}\}$ we define the errors

$$\boldsymbol{e}^{\boldsymbol{v}}(t) \coloneqq \boldsymbol{v}(t) - \boldsymbol{v}_{\tau,h}(t)\,, \qquad e^p(t) \coloneqq p(t) - p_{\tau,h}(,t)\,. \tag{5.98}$$

We study the error $(\boldsymbol{e}^{\boldsymbol{v}}, e^p)$ with respect to the norms

$$\|e^w\|_{L^\infty(L^2)} \coloneqq \max_{t\in I}\left(\int_\Omega \|e^w\|^2 \mathrm{d}\boldsymbol{x}\right)^{\frac{1}{2}}, \quad \|e^w\|_{L^2(L^2)} \coloneqq \left(\int_I\int_\Omega \|e^w(t)\|^2 \mathrm{d}\boldsymbol{x}\,\mathrm{d}t\right)^{\frac{1}{2}}, \tag{5.99}$$

where $w \in (\boldsymbol{v}, p)$. The $L^\infty$-norm in time is computed on the discrete time grid

$$I = \left\{t_n^d \mid t_n^d = t_{n-1} + d\cdot k_n\cdot\tau_n, \quad k_n = 0.001\,,\ d = 0,\dots,999\,,\ n = 1,\dots,N\right\}. \tag{5.100}$$

In our experiment we study a test setting presented in [115] and choose the right-hand side function $\boldsymbol{f}$ on $\Omega\times I = (0,1)^2\times(0,1]$ in such a way, that the exact solution of the Navier–Stokes system (4.1) is given by

$$\begin{aligned}\boldsymbol{v}(\boldsymbol{x},t) &\coloneqq \begin{pmatrix}\cos(x_2\pi)\cdot\sin(t)\cdot\sin(x_1\pi)^2\cdot\sin(x_2\pi)\\ -\cos(x_1\pi)\cdot\sin(t)\cdot\sin(x_2\pi)^2\cdot\sin(x_1\pi)\end{pmatrix},\\ p(\boldsymbol{x},t) &\coloneqq \cos(x_2\pi)\cdot\sin(t)\cdot\sin(x_1\pi)\cdot\cos(x_1\pi)\cdot\sin(x_2\pi)\,.\end{aligned} \tag{5.101}$$

We prescribe a Dirichlet boundary condition, given by the solution (5.101), on the whole boundary such that $\Gamma_D = \partial\Omega$, i.e. $\boldsymbol{g} = \boldsymbol{0}$. The initial condition is also given by (5.101), i.e. $\boldsymbol{v}_0 = \boldsymbol{0}$. For the discretization in space the $\mathcal{Q}_4$–$\mathcal{Q}_3$ pair of the Taylor–Hood family is used; cf. Fig. 5.6. The Nitsche penalty parameters in (5.92) are fixed to $\eta_1 = \eta_2 = 35$. We also compute the spectral condition number $\kappa_2(\boldsymbol{S})$ of the corresponding system matrices $\boldsymbol{S}$ of the Newton linearization, using the largest singular value $\sigma_1$ and its smallest one $\sigma_n$, by means of

$$\kappa_2(\boldsymbol{S}) = \frac{\sigma_1}{\sigma_n}\,. \tag{5.102}$$

Table 5.6a shows the calculated errors as well as the experimental orders of convergence and the spectral condition numbers for a sequence of meshes that are successively refined in space and time. We note that a test of simultaneous convergence in space and time is thus performed. In all measured norms, we observe convergence of fourth order. This is the optimal order for the Galerkin–collocation approach GCC$^1$(3) with

Table 5.6: Step sizes, resulting degrees of freedom (DoF) on each time interval $I_n$, spectral condition number $\kappa_2(\boldsymbol{S})$, errors and experimental orders of convergence (EOC) for the approximation of (5.101) with $\mathcal{Q}_4$–$\mathcal{Q}_3$ elements in space and different time integration schemes under Nitsche's method; $\tau_0 = 1.000$, $h_0 = 1/\sqrt{2}$.

(a) $GCC^1(3)$ time integration scheme.

| $\tau$ | $h$ | $\mathrm{DoF}_{\lvert I_n}$ | $\kappa_2(\boldsymbol{S})$ | $\lVert e^{\boldsymbol{v}}\rVert_{L^2(L^2)}$ | $\lVert e^{p}\rVert_{L^2(L^2)}$ | $\lVert e^{\boldsymbol{v}}\rVert_{L^\infty(L^2)}$ | $\lVert e^{p}\rVert_{L^\infty(L^2)}$ |
|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 422 | 1.085e6 | 3.099e−4 | 9.240e−4 | 7.082e−3 | 3.139e−3 |
| $\tau_0/2^1$ | $h_0/2^1$ | 1 494 | 5.716e6 | 1.954e−5 | 5.516e−5 | 4.437e−5 | 1.973e−4 |
| $\tau_0/2^2$ | $h_0/2^2$ | 5 606 | 4.370e7 | 1.226e−6 | 3.468e−6 | 2.780e−6 | 1.264e−5 |
| $\tau_0/2^3$ | $h_0/2^3$ | 21 702 | 3.459e8 | 7.673e−8 | 2.177e−7 | 1.734e−7 | 8.016e−7 |
| EOC | | – | – | 4.00 | 3.99 | 4.00 | 3.98 |

(b) cGP(1) time integration scheme.

| $\tau$ | $h$ | $\mathrm{DoF}_{\lvert I_n}$ | $\kappa_2(\boldsymbol{S})$ | $\lVert e^{\boldsymbol{v}}\rVert_{L^2(L^2)}$ | $\lVert e^{p}\rVert_{L^2(L^2)}$ | $\lVert e^{\boldsymbol{v}}\rVert_{L^\infty(L^2)}$ | $\lVert e^{p}\rVert_{L^\infty(L^2)}$ |
|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 211 | 7.213e4 | 1.358e−2 | 1.170e−2 | 1.790e−2 | 1.790e−2 |
| $\tau_0/2^1$ | $h_0/2^1$ | 747 | 2.238e5 | 3.652e−3 | 3.394e−3 | 5.662e−3 | 5.662e−3 |
| $\tau_0/2^2$ | $h_0/2^2$ | 2 803 | 8.955e5 | 9.270e−4 | 8.278e−4 | 1.502e−3 | 1.502e−3 |
| $\tau_0/2^3$ | $h_0/2^3$ | 10 851 | 3.587e6 | 2.326e−4 | 2.057e−4 | 3.805e−4 | 3.805e−4 |
| EOC | | – | – | 1.99 | 2.01 | 1.93 | 1.98 |

piecewise polynomials of order three in time. For the mixed approximation of the Navier–Stokes system by the $\mathcal{Q}_4$–$\mathcal{Q}_3$ pair of the Taylor–Hood family convergence of order four in space can at most be expected. Thus, the application of the Nitsche method does not deteriorate the convergence behavior. The optimal rate of convergence in time is thus obtained for the approximation of the velocity field and of the pressure variable. For comparison and to illustrate the quality of the $\mathrm{GCC}^1(3)$ approach, Table 5.6b also shows the results for the standard cGP(1) discretization in time of second order accuracy. Of course, comparing the step sizes or number of the degrees of freedom with the errors for both approximation schemes, the higher order $\mathrm{GCC}^1(3)$ method is superior to the cGP(1) one. Comparing the condition of the linear algebraic systems with the computed errors for both approximation schemes, we observe that for fixed condition numbers the higher order $\mathrm{GCC}^1(3)$ method yields smaller errors than the cGP(1) one. Consequently, the conditioning of the linear systems, as a measure for the complexity of their iterative solution, does not suffer from the higher order combined Galerkin–collocation approximation.

#### 5.2.3.2 Impact of Nitsche's method for flow around a cylinder

In the second numerical example, we compare the effect of imposing the boundary conditions in a weak form by using Nitsche's method (cf. Sec. 5.2.2.3) with enforcing the boundary conditions by the definition of the underlying function space (cf. Sec. 5.2.2.1 and 5.2.2.2) and, then, condensing the algebraic system by eliminating the degrees of freedom corresponding to the nodes on the Dirichlet part of the boundary. For the experimental setting we use the well-known DFG benchmark problem "flow around a cylinder", given in [46]. A section of the mesh used for the computations is shown in Fig. 5.7.

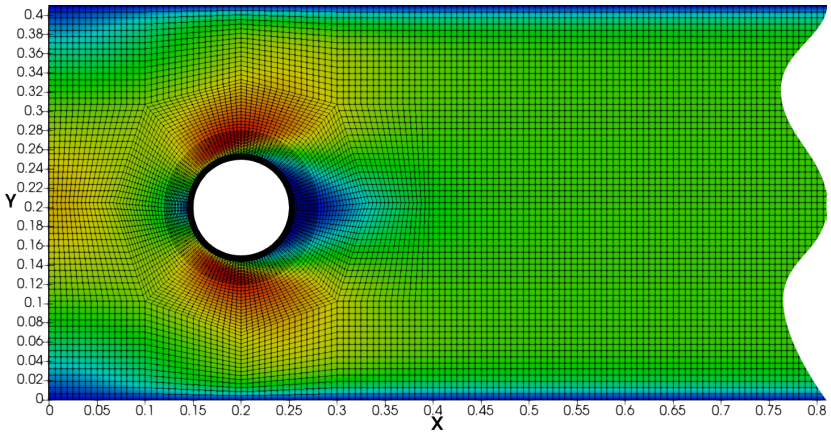



Figure 5.7: Inflow boundary at $x = 0$ and part of the spatial grid around the cylinder.

We consider the time interval $I = (0, 1]$, set $\nu = 0.01$ and let the velocity on the inflow boundary $\Gamma_i$ be given by $(\boldsymbol{x} = (x, y)^\top)$

$$\boldsymbol{g}(x, y, t) = \begin{pmatrix} -7.13861 \cdot (y - 0.41) \cdot y \cdot t^2 \\ 0 \end{pmatrix} . \tag{5.103}$$

The maximum mean velocity of the parabolic inflow profile is reached for $T = 1$ and is $\bar{U} = 0.2$. With the diameter of the cylinder as the characteristic length $L = 0.1$, this results in a Reynolds number $Re$ of

$$Re = \frac{\bar{U} \cdot L}{\nu} = \frac{0.2 \cdot 0.1}{0.01} = 2 . \tag{5.104}$$

In Fig. 5.8 we compare the computed solutions along the $y$-direction at $x = 0.2$ (cross section line through the ball's midpoint) and $T = 1$, that are obtained by the either methods. In the figures, the interval without having graphs is the cross section line that is covered by the ball. The computed profiles match perfectly such that no loss

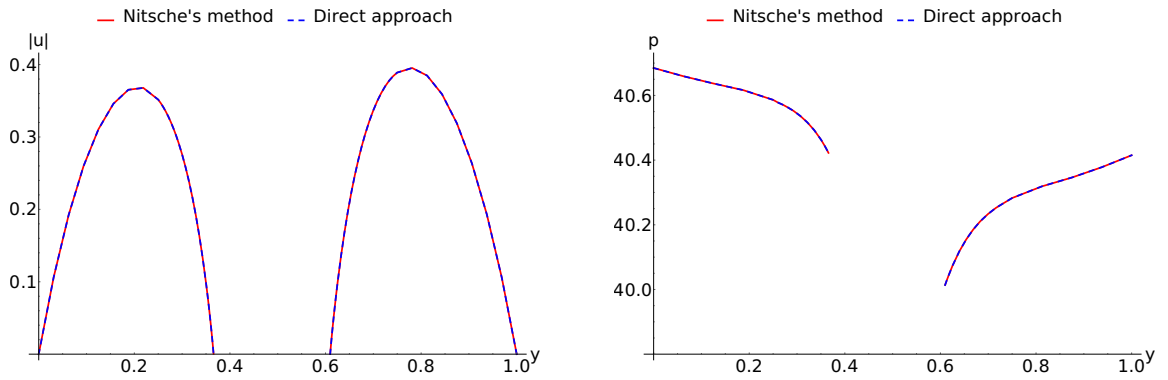



Figure 5.8: Comparison of computed velocity and pressure profiles along the $y$-axis at $x = 0.2$ (cross section line through the ball's midpoint) and $t = 1$ for Nitsche's method (weak form of Dirichlet boundary conditions) and for the enforcement of the Dirichlet boundary conditions by the definition of the function spaces with condensation of the algebraic system.

of accuracy is observed by the application of Nitsche's method of enforcing Dirichlet boundary conditions for this problem of viscous flow.

#### 5.2.3.3 GCC$^1$(3) versus cGP(1) for time-periodic flow around a cylinder

In the third numerical example we evaluate the performance properties of the Galerkin–collocation approach GCC$^1$(3) for the DFG benchmark (cf. [46]) of the time-periodic behavior of a fluid in a pipe with a circular obstacle as a more sophisticated test problem. It is set up in two space dimensions. The drag and lift coefficients of the flow on the circular cross section are computed as goal quantities of physical interest. We compare the computed results of the GCC$^1$(3) scheme with the ones obtained by the cGP(1) approximation (or Crank–Nicholson method). We use the geometrical setup as prescribed in Sec. 5.2.3.2, but consider the time interval $I = (0, 10]$, set the initial time step size $\tau = 0.01$, the viscosity $\nu = 0.001$ and prescribe the following Dirichlet boundary condition on $\Gamma_i$ by

$$g(x,y,t) = \begin{pmatrix} \frac{4 \cdot 1.5 \cdot y(0.41-y)}{0.41^2} \cdot \Big( (3t^2 - 2t^3) \cdot \big(1 - \Theta(t-1)\big) + \Theta(t-1) \Big) \\ 0 \end{pmatrix}, \qquad (5.105)$$

where $\Theta$ is the Heaviside function. Only for reasons of convenience and implementational issues, we altered the time step size slightly by choosing it as a constant, compared with the original benchmark design in [46]. Eq. (5.105) coincides with the inflow condition given in [46, p. 4], except that a smooth and non-instantaneous in-

crease in time of the profile until $t = 1$ is used. For $t \geq 1$ this results in a Reynold's number of $Re = 100$ and a time-periodic flow behavior. With the drag and lift forces $F_D$ and $F_L$ on the circle $S$ that are defined by

$$F_D = \int_S \left( \nu \frac{\partial v_t}{\partial \boldsymbol{n}} n_y - P n_x \right) \mathrm{d}S \,, \qquad F_L = - \int_S \left( \nu \frac{\partial v_t}{\partial \boldsymbol{n}} n_x - P n_y \right) \mathrm{d}S \,, \tag{5.106}$$

where $\boldsymbol{n}$ is the normal vector on $S$ and $v_t$ is the tangential velocity along $\boldsymbol{t} = (n_y, -n_x)^\top$, we can compute the drag and lift coefficients $c_D, c_L$ as our goal quantities by

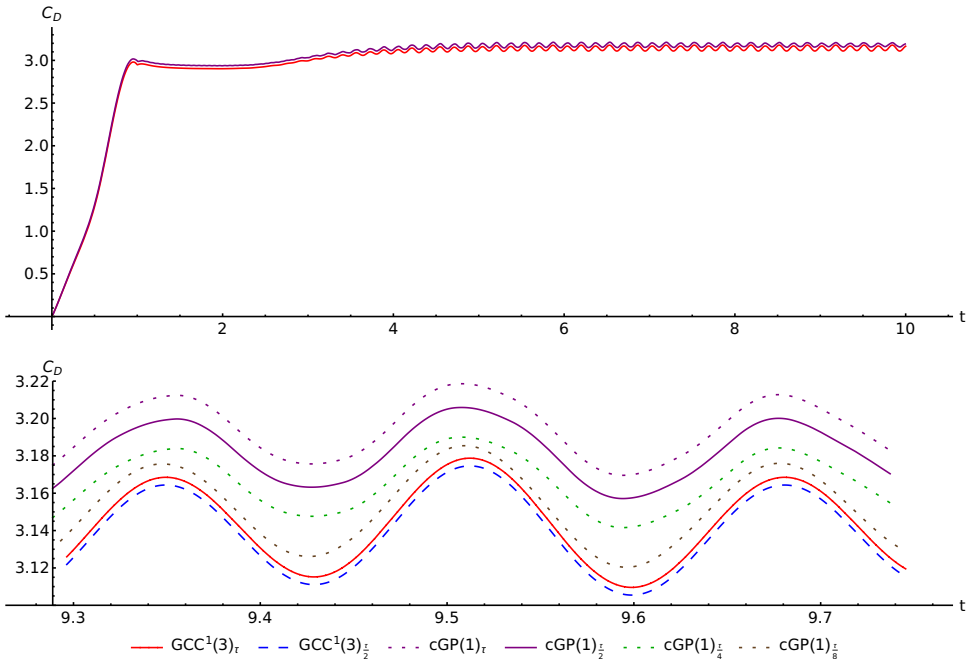
$$c_D = \frac{2}{\bar{U}^2 L} F_D \,, \qquad c_L = \frac{2}{\bar{U}^2 L} F_L \,. \tag{5.107}$$

For the spatial discretization we use a mesh consisting of 6 912 cells with higher order $\mathcal{Q}_5$–$\mathcal{Q}_4$ Taylor–Hood elements. This results in 919 104 degrees of freedom in each time interval for the GCC$^1$(3) scheme in contrast to 459 552 degrees of freedom for the cGP(1) method. In Fig. 5.9 we visualize the computed values for the drag and lift coefficient. The solid lines represent GCC$^1$(3) and cGP(1) simulations with same number of degrees of freedom, summarized over the whole simulation time. It can be observed that the cGP(1) solution converges towards the GCC$^1$(3) solution. But even with an eighth of the time step size of the GCC$^1$(3) solution it does not completely coincide with the GCC$^1$(3) solution yet. In contrast to this, the GCC$^1$(3) solution almost seems to be fully converged in the time domain with the basic time step size of $\tau$, since its further reduction to $\frac{\tau}{2}$ results in a minor change of the resulting drag and lift coefficients only. Summarizing, we can state that the GCC$^1$(3) approach is strongly superior to the cGP(1) method with respect to accuracy and efficiency.

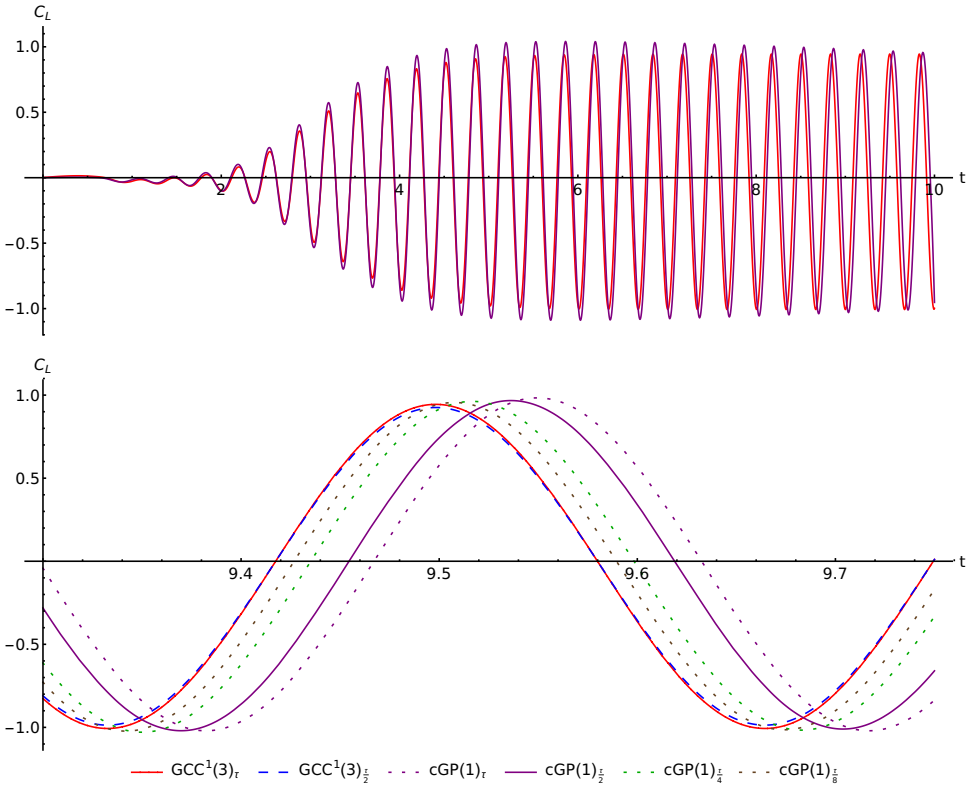
### 5.2.4 Definition of a full pressure trajectory

As noted in Sec. 5.2.1, when using continuous finite elements in the time domain one usually needs an initial value for the pressure, $p_0$. On a discrete level this need can be seen in Eq. (5.51). Hidden in the operator $A_h((\boldsymbol{v}_{\tau,h}, p_{\tau,h}), (\boldsymbol{\psi}_{\tau,h}, \xi_{\tau,h}))$ there occur integrals of the form

$$\int_{I_n} \langle p_{\tau,h}, \nabla \cdot \boldsymbol{\psi}_{\tau,h} \rangle_\Omega \, \mathrm{d}t \,. \tag{5.108}$$




(a) Drag coefficients $C_D$.




(b) Lift coefficients $c_L$.

Figure 5.9: Computed drag and lift coefficients for the example of Sec. 5.2.3.3 and different time discretization schemes with basic time step size $\tau$.

For brevity the arising problem is sketched for a cGP(1) discretization in the time domain only, although it can be generalized for any continuous time discretization scheme, that has support points on the beginning of a time interval $I_n$. If we insert the space-time expansion of the unknown variable $p_{\tau,h}$ (see Eq. (5.69)) and the test function $\boldsymbol{\psi}_{\tau,h}$ into Eq. (5.108) and utilizing that we just test with constant in time test functions, we get

$$\begin{aligned}\int_{I_n} \left\langle \sum_{l=0}^{1} p_{n,l}(\boldsymbol{x})\xi_l(t), \nabla\cdot\boldsymbol{\psi}_n \right\rangle_\Omega \mathrm{d}t &= \int_{I_n} \sum_{l=0}^{1} \xi_l(t)\,\langle p_{n,l}(\boldsymbol{x}), \nabla\cdot\boldsymbol{\psi}_n\rangle_\Omega \,\mathrm{d}t \\ &= \int_{I_n} \xi_0(t)\,\langle p_{n,0}(\boldsymbol{x}), \nabla\cdot\boldsymbol{\psi}_n\rangle_\Omega \,\mathrm{d}t + \int_{I_n} \xi_1(t)\,\langle p_{n,1}(\boldsymbol{x}), \nabla\cdot\boldsymbol{\psi}_n\rangle_\Omega \,\mathrm{d}t\,.\end{aligned}$$

It is clear, that only if

$$\int_{I_n} \xi_0(t)\,\mathrm{d}t = 0\,, \tag{5.109}$$

there is no initial pressure $p_0$ that contributes to the underlying system matrix. One natural choice for a cGP(1) system is the usage of a nodal, Lagrange time basis that uses the Gauss-Lobatto points as support points. This results in the following ansatz functions on the reference time interval:

$$\xi_0^{\mathrm{GL}}(t) = 1 - t\,, \qquad\qquad \xi_0^{\mathrm{GL}}(t) = t\,.$$

The advantage of this approach is, that the imposition of the continuity conditions in Eq. (5.49a) can be realized on a discrete level, without the need to evaluate the discrete finite element functions. This is computationally cheap. The disadvantage is, that Eq. (5.109) is not fulfilled and so, for $n = 1$, an initial pressure is needed for this kind of chosen ansatz functions in the time domain.

To overcome the issue on a discrete level, one way is to choose the trial functions in such a way, that Eq. (5.109) is fulfilled. According to [53] this is the case if one chooses the beginning of each time interval and adds $k-1$ Gauss quadrature points. For a cGP(1) scheme this results in the following ansatz functions on the reference time interval:

$$\xi_0^{\mathrm{GL}}(\hat{t}) = 1 - 2\hat{t}\,, \qquad\qquad \xi_1^{\mathrm{GL}}(\hat{t}) = 2\hat{t}\,. \tag{5.110}$$

With such ansatz functions, Eq. (5.109) is fulfilled and after the application of such a cGP(1) scheme one gets a complete set of velocity coefficient vectors, but just a reduced set of computed pressure coefficient vectors. The pressure coefficient belong to the mid points on each time interval, which is sketched in Fig. 5.10. In the next three

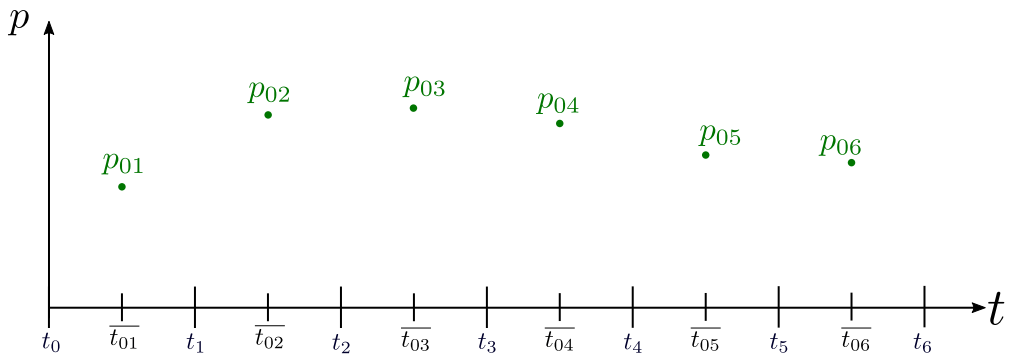



Figure 5.10: Position of the discrete pressure solution in the time mesh, using the ansatz functions of Eq. (5.110).

subsections, methods are proposed to define a complete pressure trajectory by these reduced set of pressure coefficient vectors. For a rigorous study of the initial pressure problem for continuous finite elements in the time domain, we refer to [133]. In [134] second order pressure estimates are given for the Crank-Nicolson scheme under reduced regularity assumptions on the initial data, utilizing an implicit Euler step in the first time step.

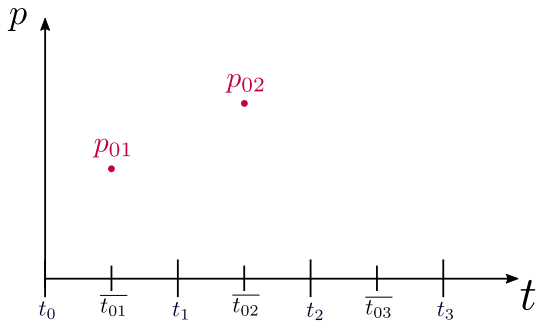
#### 5.2.4.1 By extrapolation

The first scheme is based on a linear extrapolation of the initial pressure $p_0$ after the first two time steps, using the first two solutions at the midpoints of the first two time intervals as support points:

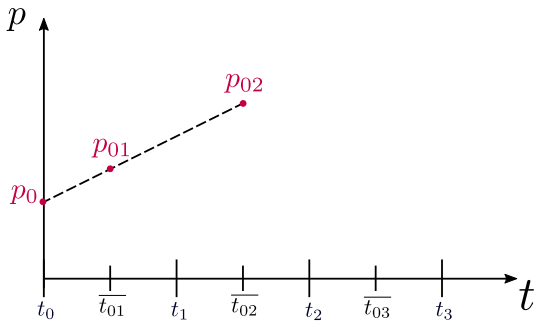
$$p_0 = \frac{3}{2}\, p_{01} - \frac{1}{2}\, p_{02}\,.$$

Afterwards we have a complete set of pressure coefficient functions in the first time interval and can compute the corresponding pressure $p_1$ at the end of the first time interval, $t_1$, which is then forwarded to the second time interval and completes the set of pressure coefficient functions there. Beginning from time interval three, we can start a usual cGP(1) scheme and output or store the two pressure coefficient functions in each time step. The procedure is sketched in Fig. 5.11.

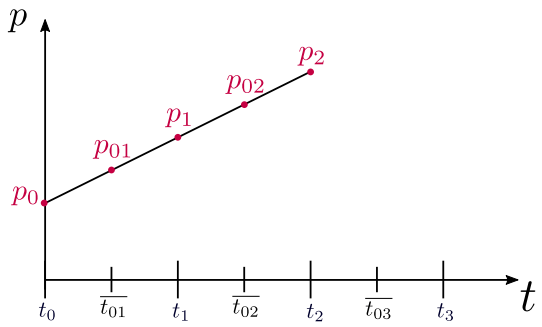
To numerically test this approach, we use the convergence test, we use a similar setup as in Sec. 5.2.3.1, with the exact solution given in Eq. (5.101). This time $\Omega \times I$ is set to $(0,1)^2 \times (1,2]$, so that the pressure $p_0$ is explicitly not 0 at the start of the simulation. $(\mathbb{P}_1(I_n; H_h^2))^2 \times \mathbb{P}_1(I_n; H_h^1)$ elements are used on each $I_n$. For $\tau_0 = 0.5$ and $h_0 = \frac{1}{2\cdot\sqrt{2}}$, Table 5.7 shows the corresponding experimental order of convergence for $\boldsymbol{v}$ and $p$. All experimental orders of convergence are as expected.



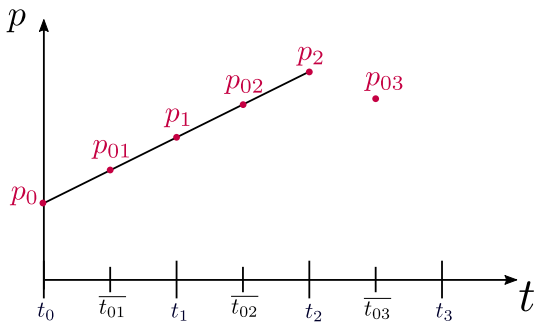

(a) Available solutions after two time steps.



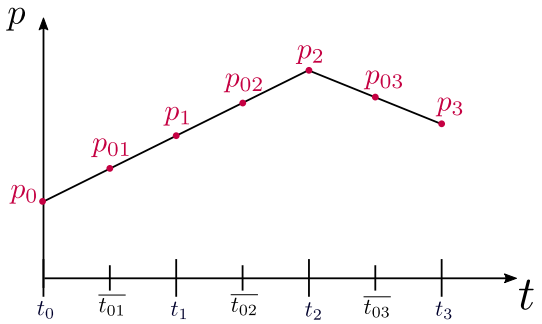

(b) Extrapolating initial solution $p_0$.




(c) Evaluating FE solution in time interval 1 at $t_1$.




(d) Continuing to time step 3.




(e) Evaluating FE solution in time interval 3 at $t_3$.

Figure 5.11: Solution of the CutFEM simulation at time $t = 10.12$.

Table 5.7: Errors and experimental order of convergence for $\tau_0 = 1.0$ and $h_0 = 1/(2\sqrt{2})$.

| $\tau$ | $h$ | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC | $\|e^{\boldsymbol{v}}\|_{L^\infty(L^2)}$ | EOC | $\|e^{p}\|_{L^\infty(L^2)}$ | EOC |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 7.899e−3 | – | 2.924e−2 | – | 9.385e−3 | – | 2.989e−2 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 1.162e−3 | 2.77 | 4.536e−3 | 2.69 | 2.253e−3 | 2.06 | 5.103e−3 | 2.55 |
| $\tau_0/2^2$ | $h_0/2^2$ | 2.099e−4 | 2.47 | 1.041e−3 | 2.12 | 5.202e−4 | 2.11 | 1.210e−3 | 2.08 |
| $\tau_0/2^3$ | $h_0/2^3$ | 4.589e−5 | 2.19 | 2.541e−4 | 2.03 | 1.056e−4 | 2.30 | 2.962e−4 | 2.03 |
| $\tau_0/2^4$ | $h_0/2^4$ | 1.102e−5 | 2.06 | 6.299e−5 | 2.01 | 2.487e−5 | 2.09 | 7.324e−5 | 2.02 |
| $\tau_0/2^4$ | $h_0/2^4$ | 2.727e−6 | 2.02 | 1.572e−5 | 2.00 | 6.446e−6 | 1.95 | 1.827e−5 | 2.00 |

#### 5.2.4.2 By Local interpolation

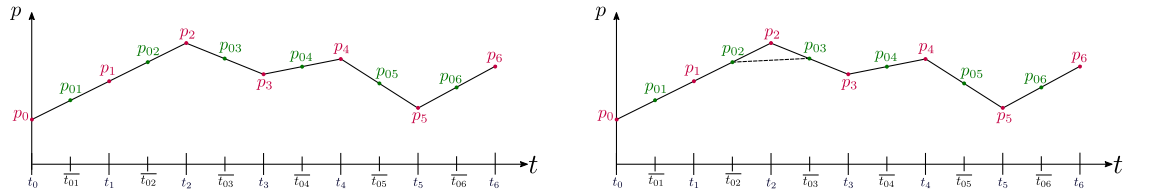
Another way, which avoids propagating extrapolated information through all time intervals is to apply just a local interpolation between two midpoints of each time interval. The idea is to firstly apply the steps after two time steps, explained in Sec. 5.2.4.1, to get an extrapolated initial pressure $p_0$ and $p_1$. Afterwards the time marching scheme is continued. Beginning in time step 3, we compute then the solution at the end point of the last time interval by local interpolation:

$$p_{n-1} = \frac{1}{2}\, p_{0(n)} + \frac{1}{2}\, p_{0(n-1)} \quad \forall n \in [3, \dots, N]\,.$$

This alone would result in a kink in the pressure solution in time step $n-1$. Therefore, afterwards also the solution $p_{0(n-1)}$ is corrected by a local interpolation:

$$p_{0(n-1)} = \frac{1}{2}\, p_{n-2} + \frac{1}{2}\, p_{n-1} \quad \forall n \in [3, \dots, N]\,.$$

The scheme is sketched in Fig. 5.12. Table 5.8 shows the results for the same conver-



(a) Solution after applying extrapolation of Sec. 5.2.4.1.

(b) After time step 3, pressure $p_2$ is updated...

(c) ... by interpolation between $p_{02}$ and $p_{03}$.

(d) Afterwards $p_{02}$ gets corrected...

(e) ... so that the solution is linear on $\overline{t_{02}}$.

(f) In the last time step, the final solution is extrapolated.

Figure 5.12: Sketch of the local extrapolation scheme.

gence test as in Sec. 5.2.4.1. Again, the experimental orders of convergence are as

Table 5.8: Errors and experimental order of convergence for $\tau_0 = 1.0$ and $h_0 = 1/(2\sqrt{2})$.

| $\tau$ | $h$ | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC | $\|e^{\boldsymbol{v}}\|_{L^\infty(L^2)}$ | EOC | $\|e^{p}\|_{L^\infty(L^2)}$ | EOC |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 8.029e−3 | – | 2.127e−2 | – | 9.622e−3 | – | 2.316e−2 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 1.163e−3 | 2.79 | 4.562e−3 | 2.22 | 2.223e−3 | 2.11 | 5.083e−3 | 2.19 |
| $\tau_0/2^2$ | $h_0/2^2$ | 2.100e−4 | 2.47 | 1.092e−3 | 2.06 | 5.171e−4 | 2.10 | 1.193e−3 | 2.09 |
| $\tau_0/2^3$ | $h_0/2^3$ | 4.590e−5 | 2.19 | 2.705e−4 | 2.01 | 1.052e−4 | 2.30 | 2.930e−4 | 2.03 |
| $\tau_0/2^4$ | $h_0/2^4$ | 1.102e−5 | 2.06 | 6.756e−5 | 2.00 | 2.487e−5 | 2.08 | 7.292e−5 | 2.01 |
| $\tau_0/2^4$ | $h_0/2^4$ | 2.727e−6 | 2.02 | 1.690e−5 | 2.00 | 6.449e−6 | 1.95 | 1.821e−5 | 2.00 |

expected.

#### 5.2.4.3 By Lifting

The third approach is based on a post-processing of the fully discrete solution of Problem 5.3. This approach is based on [9], where such a post-processing was applied to the wave equation in almost the same manner. The idea is to use an arbitrary initial pressure $p_0$, for example in our numerical tests we set $p_0 = 0$. Then the fully discrete cGP(1) solution is extended on each time interval. With the notation of Sec. 5.2.2.2 we define:

$$\widetilde{\boldsymbol{v}}_{\tau,h|I_n} \coloneqq \boldsymbol{v}_{\tau,h|I_n} + \sum_{j=1}^{J} \boldsymbol{v}_{n,2,j} \boldsymbol{\psi}_j(\boldsymbol{x}) \xi_2(t)\,,$$
$$\widetilde{p}_{\tau,h|I_n} \coloneqq p_{\tau,h|I_n} + \sum_{m=1}^{M} p_{n,2,m} \phi_m(\boldsymbol{x}) \frac{\mathrm{d}\xi_2(t)}{\mathrm{d}t}\,,$$

with $\xi_2(t) \in \mathbb{P}_2(I_n;\mathbb{R})$. In the computational tests this additional pressure term corrects the (possibly) wrong guess of the initial pressure and leads to the expected experimental order of convergence.

The coefficients of $\xi_2(\hat{t})$ on the reference time interval are defined in such a way, that the function values vanish on the boundary of each time interval $I_n$. Further the time derivative is set to one on the beginning of each time slab. On the reference time interval this leads to the following conditions:

$$\xi_2(0) = 0\,, \quad \xi_2(1) = 0\,, \quad \frac{\mathrm{d}\xi_2(t=0)}{\mathrm{d}t} = 0\,. \tag{5.111}$$

With these, $\xi_2(\hat{t})$ can be expressed as:

$$\xi_2(\hat{t}) = \hat{t} - \hat{t}^2\,.$$

The spatial unknown coefficient vectors $\boldsymbol{v}_{n,2}$ and $\boldsymbol{p}_{n,2}$ are defined by a collocation condition at $t_{n-1}$ on each time interval $I_n$:

$$\langle \partial_t \widetilde{\boldsymbol{v}}_{\tau,h}(t_{n-1}), \boldsymbol{\psi}_h \rangle_{\Omega_f} = L(\psi_h; \boldsymbol{f}(t_{n-1}), \boldsymbol{g}(t_{n-1})) - A((\widetilde{\boldsymbol{v}}_{\tau,h}(t_{n-1}), \widetilde{p}_{\tau,h}(t_{n-1})), (\boldsymbol{\psi}_h, \xi_h)) \,,$$

for all $\boldsymbol{\phi}_h \in \boldsymbol{V}_h^0 \times Q_h$. With the conditions of Eq. (5.111) and using $\nabla \cdot \partial_t \widetilde{\boldsymbol{v}}_{\tau,h} = 0$ this can be rearranged to:

$$\begin{aligned} \langle \partial_t \widetilde{\boldsymbol{v}}_{\tau,h}(t_{n-1}), \boldsymbol{\psi}_h \rangle_{\Omega_f} - \langle \widetilde{p}_{\tau,h}(t_{n-1}), \nabla \cdot \boldsymbol{\psi} \rangle_{\Omega_f} + \langle \nabla \cdot \partial_t \widetilde{\boldsymbol{v}}_{\tau,h}(t_{n-1}), \xi \rangle_{\Omega_f} \\ = L(\psi_h; \boldsymbol{f}(t_{n-1}), \boldsymbol{g}(t_{n-1})) - A((\boldsymbol{v}_{\tau,h}(t_{n-1}), p_{\tau,h}(t_{n-1})), (\boldsymbol{\psi}_h, \xi_h)) \,, \end{aligned}$$

for all $\boldsymbol{\phi}_h \in \boldsymbol{V}_h^0 \times Q_h$. This results is a linear problem for the unknown coefficient vectors $\boldsymbol{v}_{n,2}$ and $\boldsymbol{p}_{n,2}$. The resulting system matrix for the vector of unknowns $(\boldsymbol{v}_{n,2}, \boldsymbol{p}_{n,2})^\top$, with which we have to solve to post-process the solution, is:

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{M} & \boldsymbol{B}^\top \\ -\boldsymbol{B} & \boldsymbol{0} \end{pmatrix} . \tag{5.112}$$

The additional computational effort to gain the post-processed solution this solution is small, regarding that the system matrix is of similar structure as the cGP(1) system matrix with an even simpler first block ($\boldsymbol{M}$). This approach can be applied after every time step of the time marching scheme. For the numerical convergence test of Sec. 5.2.4.1 this results in the experimental orders of convergence shown in Table 5.9. The results show no increase in the experimental order of convergence of the velocity.

Table 5.9: Errors and experimental order of convergence, when applying the post-processing in every time step.

| $\tau$ | $h$ | $\|e^{\boldsymbol{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC | $\|e^{\boldsymbol{v}}\|_{L^\infty(L^2)}$ | EOC | $\|e^{p}\|_{L^\infty(L^2)}$ | EOC |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 5.481e−2 | – | 2.249e−2 | – | 8.460e−2 | – | 2.626e−2 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 7.292e−3 | 2.91 | 4.529e−3 | 2.31 | 1.118e−2 | 2.92 | 5.653e−3 | 2.22 |
| $\tau_0/2^2$ | $h_0/2^2$ | 9.587e−4 | 2.93 | 1.027e−3 | 2.14 | 1.380e−3 | 3.02 | 1.237e−3 | 2.19 |
| $\tau_0/2^3$ | $h_0/2^3$ | 1.317e−4 | 2.86 | 2.498e−4 | 2.04 | 1.725e−4 | 3.00 | 2.852e−4 | 2.12 |
| $\tau_0/2^4$ | $h_0/2^4$ | 2.331e−5 | 2.50 | 6.182e−5 | 2.01 | 2.594e−5 | 2.73 | 6.830e−5 | 2.06 |
| $\tau_0/2^4$ | $h_0/2^4$ | 5.456e−6 | 2.10 | 1.539e−5 | 2.01 | 6.226e−6 | 2.06 | 1.829e−5 | 1.90 |

This is in accordance with [53], where the cGP(1) scheme showed no superconvergence properties, but an order 2 of convergence on the whole time interval. Post-processing the solution in every time step is therefore a computational expensive way of solving the initial pressure problem. Instead of post-processing the solution and solving a problem with the system matrix of Eq. (5.112) in every time step, our computational results

indicate, that it is enough to just post-process the very first time step. Table 5.10 shows the results for such a simulation. Here the initial pressure was set to 0. After

Table 5.10: Errors and experimental order of convergence, when applying the post-processing only in the very first time step.

| $\tau$ | $h$ | $\|e^{\mathfrak{v}}\|_{L^2(L^2)}$ | EOC | $\|e^{p}\|_{L^2(L^2)}$ | EOC | $\|e^{\mathfrak{v}}\|_{L^\infty(L^2)}$ | EOC | $\|e^{p}\|_{L^\infty(L^2)}$ | EOC |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_0/2^0$ | $h_0/2^0$ | 3.392e−2 | – | 2.294e−2 | – | 6.410e−2 | – | 2.629e−2 | – |
| $\tau_0/2^1$ | $h_0/2^1$ | 3.457e−3 | 3.29 | 4.639e−3 | 2.31 | 8.974e−3 | 2.84 | 5.588e−3 | 2.23 |
| $\tau_0/2^2$ | $h_0/2^2$ | 3.851e−4 | 3.17 | 1.042e−3 | 2.15 | 1.264e−3 | 2.83 | 1.224e−3 | 2.19 |
| $\tau_0/2^3$ | $h_0/2^3$ | 5.418e−5 | 2.83 | 2.514e−4 | 2.05 | 1.679e−4 | 2.91 | 2.829e−4 | 2.11 |
| $\tau_0/2^4$ | $h_0/2^4$ | 1.110e−5 | 2.29 | 6.196e−5 | 2.02 | 2.487e−5 | 2.75 | 6.830e−5 | 2.05 |
| $\tau_0/2^4$ | $h_0/2^4$ | 2.685e−6 | 2.05 | 1.539e−5 | 2.01 | 6.221e−6 | 2.00 | 1.829e−5 | 1.90 |

solving the first time step, the solution was lifted and the initial pressure was updated with this lifting solution. Again we see the expected orders of convergence. For the velocity, the absolute error is even smaller than when applying the post-processing in every timestep (see Table 5.9).

# 6 Discussion

## 6.1 Summary

In this work we computationally analyzed higher order space-time discretizations for the Navier–Stokes equations and wave problems.

In particular, we presented a highly parallelized CutFEM approach for the nonstationary, incompressible Navier–Stokes equations in Chapter 3. It is based on mainly three ingredients:

1. A weak imposition of Dirichlet boundary conditions, using Nitsche's method
2. An iterated integration formula over all kind of cut cells
3. A ghost penalty stabilization and implicit extension of the fluid quantities to a fixed computational domain

In various numerical examples we demonstrated, that the results of simulations, utilizing this approach, converge to the results of fixed grid computations. Further we observed higher orders of convergence, even in complex scenarios with a time-dependent rigid domain.

In Chapter 4 we applied a geometric multigrid method for time-independent domains, based on a local Vanka smoother, to our higher order space-time systems. To evade the need for initial pressure values, this GMG method was applied to a discontinuous in time discretization scheme. The efficiency of this approach was demonstrated in a fixed grid flow around a cylinder benchmark in two and three space dimensions. Combining this GMG method with our CutFEM scheme on evolving domains, resulted in increased iteration numbers in the GMRES solver, compared to body fitted grids. The situation gets worse, if the size of the ghost penalty extension zone is increased. Nevertheless, for many applications, when the fluid domain is large in comparison to the rigid domain, the results of this approach look promising.

In Chapter 5 we presented a combination of the continuous Galerkin-Petrov in time discretization schemes with the classical collocation schemes to the so called Galerkin-Collocation shemes. On a discrete level, these schemes result in a solution of higher regularity in the time domain. The corresponding linear systems are smaller and less complex, than the ones of traditional continuous Galerkin-Petrov schemes. We applied such Galerkin-Collocation of $C^1$ regularity and order $k$ schemes to the wave equation and demonstrated its efficiency when we applied it to a challenging example of structural health monitoring. This was the motivation for us to also apply this scheme to flow problems. We presented the resulting linear systems and the experimental orders of convergence were as expected. Nevertheless, the GCC schemes rely, like the continuous Galerkin-Petrov schemes, on initial values and even initial values for the derivatives of the physical quantities. When it comes to the Navier–Stokes equations, these initial values for the pressure are not defined by the problem and we proposed three remedies for cGP schemes, which can similarly be applied to GCC schemes. Especially in coupled problems, when the coefficients of (time) derivatives of one problem, enter the other problem, we see a potential for the application of the Galerkin-Collocation schemes, since these coefficients are derived naturally by the scheme.

## 6.2 Outlook

In this subsection we want to give some perspectives for possibilities of research in the future on the topics, presented in this work.

The CutFEM results of Chapter 3 showed, that this approach is suitable for generating higher order space-time solutions on evolving domains. A possible extension has to be avoiding the prescription of the motion of the rigid body explicitly (cf., e.g. [67]), such that the interface has to be computed simultaneously or is given by, for instance fluid-structure interaction. In a more sophisticated scenario, the rigid body could also be replaced by an elastic body, that is modeled by the elastic wave equation. One advantage of our approach is, that the background mesh is fixed over the whole simulation time, which simplifies the further adaption towards such problems. This fixed background mesh also enables the usage of continuous or even continuous differentiable finite element methods in the time domain, for example the Galerkin-Collocation schemes. The better degrees of freedom to convergence rate of such schemes is a big advantage of these, but (especially in scenarios with moving domains) the problem of the need for initial values for the pressure variable remains.

Another topic of interest is improving the GMG solver of Chapter 4 for applications with evolving domains, utilizing CutFEM techniques. Especially for small rigid structures the results in this work look promising. It is of high interest to further reduce the GMRES iterations in cut scenarios, so that such GMG schemes can also be applied to three dimensional CutFEM problems. A further possibility for research is the tuning of the ghost penalty parameters $\gamma_v$ and $\gamma_p$. Even an automated algorithm utilizing neuronal networks is a possibility here, but out of the scope in this work.
To reduce the GMRES iterations further it might be also of interest to resolve the interface zone better, by maybe implementing an adaptive refinement strategy on all multigrid levels, so that especially the interface resolution could be increased. Of course the success of such an approach cannot be predicted and is uncertain.
If the GMRES iterations in 2d CutFEM scenarios can be drastically reduced, it is computationally within scope to adapt the CutFEM method further to 3d problems. The main open implementational challenge here is to adjust the 2d iterated integration scheme of Sec. 3.4 to the (far more complex) 3d cases.

# Bibliography


[1] S. Hussain, F. Schieweck and S. Turek, "Efficient Newton-multigrid solution techniques for higher order space–time Galerkin discretizations of incompressible flow", *Applied Numerical Mathematics*, vol. 83, pp. 51–71, 09/2014. DOI: 10.1016/j.apnum.2014.04.011.

[2] W. Dörfler, S. Findeisen and C. Wieners, "Space-Time Discontinuous Galerkin Discretizations for Linear First-Order Hyperbolic Evolution Systems", *Computational Methods in Applied Mathematics*, vol. 16, no. 3, pp. 409–428, 07/01/2016. DOI: 10.1515/cmam-2016-0015.

[3] J. Ernesti and C. Wieners, "A space-time discontinuous Petrov- Galerkin method for acousticwaves", Karlsruhe, Research Report, 2018. DOI: 10.5445/IR/1000085443.

[4] O. Steinbach, "Space-Time Finite Element Methods for Parabolic Problems", *Computational Methods in Applied Mathematics*, vol. 15, no. 4, pp. 551–566, 10/01/2015. DOI: 10.1515/cmam-2015-0026.

[5] W. Bangerth, M. Geiger and R. Rannacher, "Adaptive Galerkin Finite Element Methods for the Wave Equation", *Computational Methods in Applied Mathematics*, vol. 10, no. 1, pp. 3–48, 2010. DOI: 10.2478/cmam-2010-0001.

[6] G. Matthies and F. Schieweck, "Higher order variational time discretizations for nonlinear systems of ordinary differential equations", Otto-von-Guericke-Universität, Magdeburg, 2011. [Online]. Available: https://www-ian.math.uni-magdeburg.de/home/schieweck/Prepr_23_11.pdf.

[7] S. Zhao and G. W. Wei, "A unified discontinuous Galerkin framework for time integration", *Mathematical Methods in the Applied Sciences*, vol. 37, no. 7, pp. 1042–1071, 05/15/2014. DOI: 10.1002/mma.2863.

[8] S. Becher, G. Matthies and D. Wenzel, "Variational Methods for Stable Time Discretization of First-Order Differential Equations", in *Advanced Computing in Industrial Mathematics*, ser. Studies in Computational Intelligence, K. Georgiev, M. Todorov and I. Georgiev, Eds., vol. 793, Cham: Springer International Publishing, 2019, pp. 63–75. DOI: 10.1007/978-3-319-97277-0_6.

[9] M. Bause, M. Anselmann, G. Matthies and S. Becher, "Galerkin–collocation approximation in time for the wave equation and its post-processing", *ESAIM: Mathematical Modelling and Numerical Analysis*, 05/05/2020. DOI: 10.1051/m2an/2020033.

[10] T. J. Hughes, W. K. Liu and T. K. Zimmermann, "Lagrangian-Eulerian finite element formulation for incompressible viscous flows", *Computer Methods in Applied Mechanics and Engineering*, vol. 29, no. 3, pp. 329–349, 12/1981. DOI: 10.1016/0045-7825(81)90049-9.

[11] J. Donea, S. Giuliani and J. Halleux, "An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions", *Computer Methods in Applied Mechanics and Engineering*, vol. 33, no. 1-3, pp. 689–723, 09/1982. DOI: 10.1016/0045-7825(82)90128-1.

[12] T. Richter and T. Wick, "Finite elements for fluid–structure interaction in ALE and fully Eulerian coordinates", *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 41-44, pp. 2633–2642, 10/2010. DOI: 10.1016/j.cma.2010.04.016.

[13] T. Richter, "A monolithic geometric multigrid solver for fluid-structure interactions in ALE formulation: Monolithic Geometric Multigrid Solver for Fluid-Structure Interactions", *International Journal for Numerical Methods in Engineering*, vol. 104, no. 5, pp. 372–390, 11/02/2015. DOI: 10.1002/nme.4943.

[14] H. Braess and P. Wriggers, "Arbitrary Lagrangian Eulerian finite element analysis of free surface flow", *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 1-2, pp. 95–109, 10/2000. DOI: 10.1016/S0045-7825(99)00416-8.

[15] W. A. Wall, A. Gerstenberger, P. Gamnitzer, C. Förster and E. Ramm, "Large Deformation Fluid-Structure Interaction – Advances in ALE Methods and New Fixed Grid Approaches", in *Fluid-Structure Interaction*, ser. Lecture Notes in Computational Science and Engineering, H.-J. Bungartz and M. Schäfer, Eds., vol. 53, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 195–232. DOI: 10.1007/3-540-34596-5_9.

[16] M. Nazem, D. Sheng and J. P. Carter, "Stress integration and mesh refinement for large deformation in geomechanics", *International Journal for Numerical Methods in Engineering*, vol. 65, no. 7, pp. 1002–1027, 02/12/2006. DOI: 10.1002/nme.1470.

[17] J. L. Farinatti Aymone, "Mesh motion techniques for the ALE formulation in 3D large deformation problems: MESH MOTION TECHNIQUES", *International*

*Journal for Numerical Methods in Engineering*, vol. 59, no. 14, pp. 1879–1908, 04/14/2004. DOI: 10.1002/nme.939.

[18] A. Brenner, E. Bänsch and M. Bause, "A priori error analysis for finite element approximations of the Stokes problem on dynamic meshes", *IMA Journal of Numerical Analysis*, vol. 34, no. 1, pp. 123–146, 01/01/2014. DOI: 10.1093/imanum/drt001.

[19] J. Dolbow, N. Moës and T. Belytschko, "Discontinuous enrichment in finite elements with a partition of unity method", *Finite Elements in Analysis and Design*, vol. 36, no. 3-4, pp. 235–260, 11/2000. DOI: 10.1016/S0168-874X(00)00035-4.

[20] J. Chessa and T. Belytschko, "An Extended Finite Element Method for Two-Phase Fluids", *Journal of Applied Mechanics*, vol. 70, no. 1, pp. 10–17, 01/01/2003. DOI: 10.1115/1.1526599.

[21] E. Burman, S. Claus, P. Hansbo, M. G. Larson and A. Massing, "CutFEM: Discretizing geometry and partial differential equations", *International Journal for Numerical Methods in Engineering*, vol. 104, no. 7, pp. 472–501, 11/16/2015. DOI: 10.1002/nme.4823.

[22] B. Schott, C. Ager and W. A. Wall, "Monolithic cut finite element–based approaches for fluid-structure interaction", *International Journal for Numerical Methods in Engineering*, vol. 119, no. 8, pp. 757–796, 2019. DOI: 10.1002/nme.6072.

[23] S. Zahedi, "A Space-Time Cut Finite Element Method with Quadrature in Time", in *Geometrically Unfitted Finite Element Methods and Applications*, ser. Lecture Notes in Computational Science and Engineering, S. P. A. Bordas, E. Burman, M. G. Larson and M. A. Olshanskii, Eds., vol. 121, Cham: Springer International Publishing, 2017, pp. 281–306. DOI: 10.1007/978-3-319-71431-8_9.

[24] B. Schott, "Stabilized Cut Finite Element Methods for Complex Interface Coupled Flow Problems", Ph.D. dissertation, Technische Universität München, München, 03/15/2017. [Online]. Available: https://mediatum.ub.tum.de/doc/1304754/1304754.pdf.

[25] C. Lehrenfeld and M. Olshanskii, "An Eulerian finite element method for PDEs in time-dependent domains", *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 53, no. 2, pp. 585–614, 03/2019. DOI: 10.1051/m2an/2018068.

[26] H. von Wahl, T. Richter and C. Lehrenfeld, "An unfitted Eulerian finite element method for the time-dependent Stokes problem on moving domains", *IMA Journal of Numerical Analysis*, drab044, 07/05/2021. DOI: 10.1093/imanum/drab044.

[27] E. Burman, S. Frei and A. Massing. "Eulerian time-stepping schemes for the non-stationary Stokes equations on time-dependent domains". arXiv: 1910.03054. (12/01/2020).

[28] M. Winter, B. Schott, A. Massing and W. Wall, "A Nitsche cut finite element method for the Oseen problem with general Navier boundary conditions", *Computer Methods in Applied Mechanics and Engineering*, vol. 330, pp. 220–252, 03/2018. DOI: 10.1016/j.cma.2017.10.023.

[29] E. Burman and P. Hansbo, "Fictitious domain methods using cut elements: III. A stabilized Nitsche method for Stokes' problem", *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 48, no. 3, pp. 859–874, 05/2014. DOI: 10.1051/m2an/2013123.

[30] B. Schott and W. Wall, "A new face-oriented stabilized XFEM approach for 2D and 3D incompressible Navier–Stokes equations", *Computer Methods in Applied Mechanics and Engineering*, vol. 276, pp. 233–265, 07/2014. DOI: 10.1016/j.cma.2014.02.014.

[31] S. Court, M. Fournié and A. Lozinski, "A fictitious domain approach for the Stokes problem based on the extended finite element method", *International Journal for Numerical Methods in Fluids*, vol. 74, pp. 73–99, 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01978110.

[32] B. Schott, C. Ager and W. Wall, "A monolithic approach to fluid-structure interaction based on a hybrid Eulerian-ALE fluid domain decomposition involving cut elements", *International Journal for Numerical Methods in Engineering*, vol. 119, no. 3, pp. 208–237, 07/20/2019. DOI: 10.1002/nme.6047.

[33] S. Court and M. Fournié, "A fictitious domain finite element method for simulations of fluid–structure interactions: The Navier–Stokes equations coupled with a moving solid", *Journal of Fluids and Structures*, vol. 55, pp. 398–408, 05/2015. DOI: 10.1016/j.jfluidstructs.2015.03.013.

[34] E. Burman, M. A. Fernández and S. Frei, "A Nitsche-based formulation for fluid-structure interactions with contact", *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 54, no. 2, pp. 531–564, 03/2020. DOI: 10.1051/m2an/2019072.

[35] E. Burman, "Ghost penalty", *Comptes Rendus Mathematique*, vol. 348, no. 21-22, pp. 1217–1220, 11/2010. DOI: 10.1016/j.crma.2010.10.006.

[36] J. Preuß, "Higher order unfitted isoparametric space-time FEM on moving domains", M.S. thesis, University of Göttingen, 02/28/2018. [Online]. Available: http://cpde.math.uni-goettingen.de/data/Pre18_Ma.pdf.

[37] M. Benzi, G. H. Golub and J. Liesen, "Numerical solution of saddle point problems", *Acta Numerica*, vol. 14, pp. 1–137, 05/2005. DOI: 10.1017/S0962492904000212.

[38] H. C. Elman, D. J. Silvester and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics* (Numerical Mathematics and Scientific Computation), Second edition. Oxford, United Kingdom: Oxford University Press, 2014, 479 pp.

[39] Y. Saad and M. H. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems", *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 07/1986. DOI: 10.1137/0907058.

[40] S. Turek, *Efficient Solvers for Incompressible Flow Problems* (Lecture Notes in Computational Science and Engineering), M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose and T. Schlick, Eds. Berlin, Heidelberg: Springer, 1999, vol. 6. [Online]. Available: http://link.springer.com/10.1007/978-3-642-58393-3.

[41] A. J. Wathen, "Preconditioning", *Acta Numerica*, vol. 24, pp. 329–376, 05/01/2015. DOI: 10.1017/S0962492915000021.

[42] D. Kay, D. Loghin and A. Wathen, "A Preconditioner for the Steady-State Navier–Stokes Equations", *SIAM Journal on Scientific Computing*, vol. 24, no. 1, pp. 237–256, 01/2002. DOI: 10.1137/S106482759935808X.

[43] E. C. Cyr, J. N. Shadid and R. S. Tuminaro, "Stabilization and scalable block preconditioning for the Navier–Stokes equations", *Journal of Computational Physics*, vol. 231, no. 2, pp. 345–363, 01/2012. DOI: 10.1016/j.jcp.2011.09.001.

[44] M. A. Olshanskii and Y. V. Vassilevski, "Pressure Schur Complement Preconditioners for the Discrete Oseen Problem", *SIAM Journal on Scientific Computing*, vol. 29, no. 6, pp. 2686–2704, 01/2007. DOI: 10.1137/070679776.

[45] Y. Notay, "A new algebraic multigrid approach for Stokes problems", *Numerische Mathematik*, vol. 132, no. 1, pp. 51–84, 01/2016. DOI: 10.1007/s00211-015-0710-0.

[46] M. Schäfer, S. Turek, F. Durst, E. Krause and R. Rannacher, "Benchmark Computations of Laminar Flow Around a Cylinder", in *Flow Simulation with High-Performance Computers II*, ser. Notes on Numerical Fluid Mechanics (NNFM), E. H. Hirschel, Ed., red. by E. H. Hirschel, K. Fujii, B. van Leer, M. A. Leschziner, M. Pandolfi, A. Rizzi and B. Roux, vol. 48, Wiesbaden: Vieweg+Teubner Verlag, 1996, pp. 547–566. DOI: 10.1007/978-3-322-89849-4_39.

[47] V. John and L. Tobiska, "Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier–Stokes equations", *International Journal for Numerical Methods in Fluids*, vol. 33, no. 4, pp. 453–473, 2000. DOI: 10.1002/1097-0363(20000630)33:4<453::AID-FLD15>3.0.CO;2-0.

[48] V. John, "Higher order finite element methods and multigrid solvers in a benchmark problem for the 3D Navier-Stokes equations", *International Journal for Numerical Methods in Fluids*, vol. 40, no. 6, pp. 775–798, 10/30/2002. DOI: 10.1002/fld.377.

[49] S. Turek and H. Wobker, "Numerical Studies of Vanka-Type Smoothers in Computational Solid Mechanics", *Advances in Applied Mathematics and Mechanics*, vol. 1, no. 1, pp. 29–55, 2009.

[50] V. John, "On the efficiency of linearization schemes and coupled multigrid methods in the simulation of a 3D flow around a cylinder", *International Journal for Numerical Methods in Fluids*, vol. 50, no. 7, pp. 845–862, 03/10/2006. DOI: 10.1002/fld.1080.

[51] J. van der Vegt and S. Rhebergen, "Hp-Multigrid as Smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows: Part I. Multilevel analysis", *Journal of Computational Physics*, vol. 231, no. 22, pp. 7537–7563, 09/2012. DOI: 10.1016/j.jcp.2012.05.038.

[52] J. van der Vegt and S. Rhebergen, "Hp-Multigrid as Smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. Part II: Optimization of the Runge–Kutta smoother", *Journal of Computational Physics*, vol. 231, no. 22, pp. 7564–7583, 09/2012. DOI: 10.1016/j.jcp.2012.05.037.

[53] S. Hussain, F. Schieweck and S. Turek, "A Note on Accurate and Efficient Higher Order Galerkin Time Stepping Schemes for the Nonstationary Stokes Equations", *The Open Numerical Methods Journal*, vol. 4, no. 1, pp. 35–45, 01/23/2012. DOI: 10.2174/1876389801204010035.

[54] S. Hussain, F. Schieweck and S. Turek, "An efficient and stable finite element solver of higher order in space and time for nonstationary incompressible flow", *International Journal for Numerical Methods in Fluids*, vol. 73, no. 11, pp. 927–952, 12/20/2013. DOI: 10.1002/fld.3831.

[55] S. Vanka, "Block-implicit multigrid calculation of two-dimensional recirculating flows", *Computer Methods in Applied Mechanics and Engineering*, vol. 59, no. 1, pp. 29–48, 11/1986. DOI: 10.1016/0045-7825(86)90022-8.

[56] D. Braess and R. Sarazin, "An efficient smoother for the Stokes problem", *Applied Numerical Mathematics*, vol. 23, no. 1, pp. 3–19, 02/1997. DOI: 10.1016/S0168-9274(96)00059-1.

[57] D. Arndt, W. Bangerth, B. Blais, T. C. Clevenger, M. Fehling, A. V. Grayver, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, R. Rastak, I. Tomas, B. Turcksin, Z. Wang and D. Wells, "The deal.II library, Version 9.2", *Journal of Numerical Mathematics*, vol. 28, no. 3, pp. 131–146, 09/25/2020. DOI: 10.1515/jnma-2020-0043.

[58] J. Baiges and R. Codina, "The fixed-mesh ALE approach applied to solid mechanics and fluid–structure interaction problems", *International Journal for Numerical Methods in Engineering*, vol. 81, no. 12, pp. 1529–1557, 2010. DOI: 10.1002/nme.2740.

[59] 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*, vol. 36, no. 1, pp. 9–15, 07/1971. DOI: 10.1007/BF02995904.

[60] E. Burman and P. Hansbo, "Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method", *Applied Numerical Mathematics*, vol. 62, no. 4, pp. 328–341, 04/2012. DOI: 10.1016/j.apnum.2011.01.008.

[61] R. Becker, "Mesh adaptation for Dirichlet flow control via Nitsche's method", *Communications in Numerical Methods in Engineering*, vol. 18, no. 9, pp. 669–680, 2002. DOI: 10.1002/cnm.529.

[62] T. Boiveau and E. Burman, "A penalty-free Nitsche method for the weak imposition of boundary conditions in compressible and incompressible elasticity",

*IMA Journal of Numerical Analysis*, vol. 36, no. 2, pp. 770–795, 04/2016. DOI: 10.1093/imanum/drv042.

[63] A. Massing, M. G. Larson, A. Logg and M. E. Rognes, "A Stabilized Nitsche Fictitious Domain Method for the Stokes Problem", *Journal of Scientific Computing*, vol. 61, no. 3, pp. 604–628, 12/01/2014. DOI: 10.1007/s10915-014-9838-9.

[64] H. Xie, Z. Li and Z. Qiao, "A Finite Element Method for Elasticity Interface Problems with Locally Modified Triangulations", *International Journal of Numerical Analysis and Modeling*, vol. 8, no. 2, pp. 189–200, 2011. pmid: 24058368.

[65] E. S. Gawlik and A. J. Lew, "High-order finite element methods for moving boundary problems with prescribed boundary evolution", *Computer Methods in Applied Mechanics and Engineering*, vol. 278, pp. 314–346, 08/2014. DOI: 10.1016/j.cma.2014.05.008.

[66] T. Carraro and S. Wetterauer, "On the implementation of the eXtended Finite Element Method (XFEM) for interface problems", *Archive of Numerical Software*, vol. 4, no. 2, pp. 1–23, 2016. DOI: 10.11588/ANS.2016.2.22317.

[67] S. Frei, T. Richter and T. Wick, "LocModFE: Locally modified finite elements for approximating interface problems in deal.II", *Software Impacts*, vol. 8, p. 100 070, 05/2021. DOI: 10.1016/j.simpa.2021.100070.

[68] A. Massing, M. G. Larson and A. Logg, "Efficient Implementation of Finite Element Methods on Nonmatching and Overlapping Meshes in Three Dimensions", *SIAM Journal on Scientific Computing*, vol. 35, no. 1, pp. C23–C47, 01/2013. DOI: 10.1137/11085949X.

[69] Y. Sudhakar, J. Moitinho de Almeida and W. A. Wall, "An accurate, robust, and easy-to-implement method for integration over arbitrary polyhedra: Application to embedded interface methods", *Journal of Computational Physics*, vol. 273, pp. 393–415, 09/2014. DOI: 10.1016/j.jcp.2014.05.019.

[70] E. Burman and M. A. Fernández, "Continuous interior penalty finite element method for the time-dependent Navier–Stokes equations: Space discretization and convergence", *Numerische Mathematik*, vol. 107, no. 1, pp. 39–77, 07/01/2007. DOI: 10.1007/s00211-007-0070-5.

[71] H.-G. Roos, M. Stynes, L. Tobiska and H.-G. Roos, *Robust Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion-Reaction and Flow Problems* (Springer Series in Computational Mathematics 24), 2nd ed. Berlin: Springer-Verlag, 2008, 604 pp.

[72] E. Burman and M. A. Fernández, "An unfitted Nitsche method for incompressible fluid–structure interaction using overlapping meshes", *Computer Methods in Applied Mechanics and Engineering*, vol. 279, pp. 497–514, 09/2014. DOI: 10.1016/j.cma.2014.07.007.

[73] P. Hansbo, M. G. Larson and S. Zahedi, "A cut finite element method for coupled bulk-surface problems on time-dependent domains", *Computer Methods in Applied Mechanics and Engineering*, vol. 307, pp. 96–116, 08/2016. DOI: 10.1016/j.cma.2016.04.012.

[74] E. Burman, D. Elfverson, P. Hansbo, M. G. Larson and K. Larsson, "A cut finite element method for the Bernoulli free boundary value problem", *Computer Methods in Applied Mechanics and Engineering*, vol. 317, pp. 598–618, 04/2017. DOI: 10.1016/j.cma.2016.12.021.

[75] T. Frachon and S. Zahedi, "A cut finite element method for incompressible two-phase Navier–Stokes flows", *Journal of Computational Physics*, vol. 384, pp. 77–98, 05/2019. DOI: 10.1016/j.jcp.2019.01.028.

[76] A. Limache and S. Idelsohn, "Laplace Form Of Navier-Stokes Equations: A Safe Path Or A Wrong Way?", *Asociación Argentina de Mecánica Computacional*, vol. XXV, no. 2, pp. 151–168, 2006.

[77] V. John, *Finite Element Methods for Incompressible Flow Problems* (Springer Series in Computational Mathematics). Cham: Springer International Publishing, 2016, vol. 51. DOI: 10.1007/978-3-319-45750-5.

[78] J. G. Heywood, R. Rannacher and S. Turek, "Artificial boundaries and flux and pressure conditions for the incompressible Navier–Stokes equations", *International Journal for Numerical Methods in Fluids*, vol. 22, no. 5, pp. 325–352, 1996. DOI: 10.1002/(SICI)1097-0363(19960315)22:5<325::AID-FLD307>3.0.CO;2-Y.

[79] R. Salvi, "On the Navier-Stokes equations in non-cylindrical domains: On the existence and regularity", *Mathematische Zeitschrift*, vol. 199, no. 2, pp. 153–170, 06/1988. DOI: 10.1007/BF01159649.

[80] D. N. Bock, "On the Navier-Stokes equations in noncylindrical domains", *Journal of Differential Equations*, vol. 25, no. 2, pp. 151–162, 08/1977. DOI: 10.1016/0022-0396(77)90197-8.

[81] J. G. Heywood and R. Rannacher, "Finite Element Approximation of the Nonstationary Navier–Stokes Problem. I. Regularity of Solutions and Second-Order

Error Estimates for Spatial Discretization", *SIAM Journal on Numerical Analysis*, vol. 19, no. 2, pp. 275–311, 04/1982. DOI: 10.1137/0719018.

[82] M. Bause, "On optimal convergence rates for higher-order Navier–Stokes approximations. I. Error estimates for the spatial discretization", *IMA Journal of Numerical Analysis*, vol. 25, no. 4, pp. 812–841, 10/01/2005. DOI: 10.1093/imanum/dri019.

[83] F. Sonner and T. Richter, "Second Order Pressure Estimates for the Crank–Nicolson Discretization of the Incompressible Navier–Stokes Equations", *SIAM Journal on Numerical Analysis*, vol. 58, no. 1, pp. 375–409, 01/2020. DOI: 10.1137/18M1234813.

[84] A. Alphonse, C. Elliott and B. Stinner, "An abstract framework for parabolic PDEs on evolving spaces", *Portugaliae Mathematica*, vol. 72, no. 1, pp. 1–46, 2015. DOI: 10.4171/PM/1955.

[85] J. Benk, M. Ulbrich and M. Mehl, "The Nitsche Method of the Navier Stokes Equations for Immersed and Moving Boundaries", in *Seventh International Conference on Computational Fluid Dynamics*, 2012.

[86] C. Ager, B. Schott, M. Winter and W. Wall, "A Nitsche-based cut finite element method for the coupling of incompressible fluid flow with poroelasticity", *Computer Methods in Applied Mechanics and Engineering*, vol. 351, pp. 253–280, 07/2019. DOI: 10.1016/j.cma.2019.03.015.

[87] S. Hussain, F. Schieweck and S. Turek, "Higher order Galerkin time discretizations and fast multigrid solvers for the heat equation", *Journal of Numerical Mathematics*, vol. 19, no. 1, pp. 41–61, 01/2011. DOI: 10.1515/jnum.2011.003.

[88] U. Köcher and M. Bause, "Variational Space–Time Methods for the Wave Equation", *Journal of Scientific Computing*, vol. 61, no. 2, pp. 424–453, 11/01/2014. DOI: 10.1007/s10915-014-9831-3.

[89] X. S. Li and J. W. Demmel, "SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems", *ACM Transactions on Mathematical Software*, vol. 29, no. 2, pp. 110–140, 06/2003. DOI: 10.1145/779359.779361.

[90] R. P. Pawlowski, J. P. Simonis, H. F. Walker and J. N. Shadid, "Inexact Newton Dogleg Methods", *SIAM Journal on Numerical Analysis*, vol. 46, no. 4, pp. 2112–2132, 01/2008. DOI: 10.1137/050632166.

[91] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice-Hall Series in Computational Mathematics). Englewood Cliffs, N.J: Prentice-Hall, 1983, 378 pp.

[92] R. S. Tuminaro, H. F. Walker and J. N. Shadid, "On Backtracking Failure in Newton–GMRES Methods with a Demonstration for the Navier–Stokes Equations", *Journal of Computational Physics*, vol. 180, no. 2, pp. 549–558, 08/2002. DOI: 10.1006/jcph.2002.7102.

[93] R. P. Pawlowski, J. N. Shadid, J. P. Simonis and H. F. Walker, "Globalization Techniques for Newton–Krylov Methods and Applications to the Fully Coupled Solution of the Navier–Stokes Equations", *SIAM Review*, vol. 48, no. 4, pp. 700–721, 01/2006. DOI: 10.1137/S0036144504443511.

[94] The Trilinos Project Team, *The Trilinos Project Website*, manual, 07/18/2020. [Online]. Available: https://trilinos.github.io/.

[95] V. John and G. Matthies, "Higher-order finite element discretizations in a benchmark problem for incompressible flows", *International Journal for Numerical Methods in Fluids*, vol. 37, no. 8, pp. 885–903, 12/30/2001. DOI: 10.1002/fld.195.

[96] M. Braack and T. Richter, "Solutions of 3D Navier–Stokes benchmark problems with adaptive finite elements", *Computers & Fluids*, vol. 35, no. 4, pp. 372–392, 05/2006. DOI: 10.1016/j.compfluid.2005.02.001.

[97] T. Richter, *Fluid-Structure Interactions: Models, Analysis and Finite Elements* (Lecture Notes in Computational Science and Engineering 118). Cham: Springer, 2017, 436 pp.

[98] T. C. Clevenger, T. Heister, G. Kanschat and M. Kronbichler, "A Flexible, Parallel, Adaptive Geometric Multigrid Method for FEM", *ACM Transactions on Mathematical Software*, vol. 47, no. 1, pp. 1–27, 01/06/2021. DOI: 10.1145/3425193.

[99] M. Kronbichler, A. Diagne and H. Holmgren, "A fast massively parallel two-phase flow solver for microfluidic chip simulation", *The International Journal of High Performance Computing Applications*, vol. 32, no. 2, pp. 266–287, 03/2018. DOI: 10.1177/1094342016671790.

[100] G. Kanschat and Y. Mao, "Multigrid methods for $H^{\mathrm{div}}$-conforming discontinuous galerkin methods for the stokes equations", *Journal of Numerical Mathematics*, vol. 23, no. 1, 01/01/2015. DOI: 10.1515/jnma-2015-0005.

[101] S. Basting and E. Bänsch, "Preconditioners for the Discontinuous Galerkin time-stepping method of arbitrary order", *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 51, no. 4, pp. 1173–1195, 07/2017. DOI: 10.1051/m2an/2016055.

[102] U. Köcher, M. P. Bruchhäuser and M. Bause, "Efficient and scalable data structures and algorithms for goal-oriented adaptivity of space–time FEM codes", *SoftwareX*, vol. 10, p. 100 239, 07/2019. DOI: 10.1016/j.softx.2019.100239.

[103] M. Benzi, G. H. Golub and J. Liesen, "Numerical solution of saddle point problems", *Acta Numerica*, vol. 14, pp. 1–137, 05/2005. DOI: 10.1017/S0962492904000212.

[104] S. Vanka, "Block-implicit multigrid solution of Navier-Stokes equations in primitive variables", *Journal of Computational Physics*, vol. 65, no. 1, pp. 138–158, 07/1986. DOI: 10.1016/0021-9991(86)90008-2.

[105] J. Molenaar, "A two-grid analysis of the combination of mixed finite elements and Vanka-type relaxation", in *Multigrid Methods III*, W. Hackbusch and U. Trottenberg, Eds., Basel: Birkhäuser Basel, 1991, pp. 313–323. DOI: 10.1007/978-3-0348-5712-3_23.

[106] G. Matthies and L. Tobiska, "Mass conservation of finite element methods for coupled flow-transport problems", *International Journal of Computing Science and Mathematics*, vol. 1, p. 293, 2/3/4 2007. DOI: 10.1504/IJCSM.2007.016537.

[107] G. Matthies and L. Tobiska, "The inf-sup condition for the mapped $Q_k$ / $P_{k-1}^{\text{disc}}$ element in arbitrary apace dimensions", *Computing*, vol. 69, no. 2, pp. 119–139, 10/2002. DOI: 10.1007/s00607-002-1451-3.

[108] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities", in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference on - AFIPS '67 (Spring)*, Atlantic City, New Jersey: ACM Press, 1967, p. 483. DOI: 10.1145/1465482.1465560.

[109] M. Emami, "Efficient Multigrid Solvers for the Stokes Equations using Finite Elements", M.S. thesis, Friedrich-Alexander-Universität, Erlangen-Nürnberg, 02/12/2013.

[110] N. Ahmed, S. Becher and G. Matthies, "Higher-order discontinuous Galerkin time stepping and local projection stabilization techniques for the transient Stokes problem", *Computer Methods in Applied Mechanics and Engineering*, vol. 313, pp. 28–52, 01/2017. DOI: 10.1016/j.cma.2016.09.026.

[111] N. Ahmed and G. Matthies, "Numerical Studies of Higher Order Variational Time Stepping Schemes for Evolutionary Navier-Stokes Equations", in *Boundary and Interior Layers, Computational and Asymptotic Methods BAIL 2016*, ser. Lecture Notes in Computational Science and Engineering, Z. Huang, M. Stynes and Z. Zhang, Eds., vol. 120, Cham: Springer International Publishing, 2017, pp. 19–33. DOI: 10.1007/978-3-319-67202-1_2.

[112] O. Karakashian and C. Makridakis, "Convergence of a continuous galerkin method with mesh modification for nonlinear wave equations", *Mathematics of Computation*, vol. 74, no. 249, pp. 85–102, 2005. JSTOR: 4100238.

[113] O. Steinbach and H. Yang, "An Algebraic Multigrid Method for an Adaptive Space–Time Finite Element Discretization", in *Large-Scale Scientific Computing*, ser. Lecture Notes in Computer Science, I. Lirkov and S. Margenov, Eds., vol. 10665, Cham: Springer International Publishing, 2018, pp. 66–73. DOI: 10.1007/978-3-319-73441-5_6.

[114] U. Langer, M. Neumüller and A. Schafelner, "Space-Time Finite Element Methods for Parabolic Evolution Problems with Variable Coefficients", in *Advanced Finite Element Methods with Applications*, ser. Lecture Notes in Computational Science and Engineering, T. Apel, U. Langer, A. Meyer and O. Steinbach, Eds., vol. 128, Cham: Springer International Publishing, 2019, pp. 247–275. DOI: 10.1007/978-3-030-14244-5_13.

[115] M. Besier and R. Rannacher, "Goal-oriented space–time adaptivity in the finite element Galerkin method for the computation of nonstationary incompressible flow", *International Journal for Numerical Methods in Fluids*, vol. 70, no. 9, pp. 1139–1166, 2012. DOI: 10.1002/fld.2735.

[116] M. P. Bruchhäuser, K. Schwegler and M. Bause, "Dual weighted residual based error control for nonstationary convection-dominated equations: Potential or ballast?", in *Boundary and Interior Layers, Computational and Asymptotic Methods BAIL 2018*, G. R. Barrenechea and J. Mackenzie, Eds., Cham: Springer International Publishing, 2020, pp. 1–17.

[117] G. Akrivis, C. Makridakis and R. H. Nochetto, "Galerkin and Runge–Kutta methods: Unified formulation, a posteriori error estimates and nodal superconvergence", *Numerische Mathematik*, vol. 118, no. 3, pp. 429–456, 07/2011. DOI: 10.1007/s00211-011-0363-6.

[118] C. Makridakis and R. H. Nochetto, "A posteriori error analysis for higher order dissipative methods for evolution problems", *Numerische Mathematik*, vol. 104, no. 4, pp. 489–514, 09/25/2006. DOI: 10.1007/s00211-006-0013-6.

[119] M. Bause, U. Köcher, F. A. Radu and F. Schieweck, "Post-processed Galerkin approximation of improved order for wave equations", *Mathematics of Computation*, vol. 89, no. 322, pp. 595–627, 08/30/2019. DOI: 10.1090/mcom/3464.

[120] J. Ernesti, "Space-Time Methods for Acoustic Waves with Applications to Full Waveform Inversion", Ph.D. dissertation, Karlsruhe, 2018. [Online]. Available: https://publikationen.bibliothek.kit.edu/1000082807.

[121] A. Mikelić and M. F. Wheeler, "Theory of the dynamic Biot-Allard equations and their link to the quasi-static Biot system", *Journal of Mathematical Physics*, vol. 53, no. 12, p. 123 702, 12/2012. DOI: 10.1063/1.4764887.

[122] U. Köcher, "Variational Space-Time Methods for the Elastic Wave Equation and the Diffusion Equation", Ph.D. dissertation, Helmut-Schmidt-University, University of the German Federal Armed Forces Hamburg, 2015. [Online]. Available: https://openhsu.ub.hsu-hh.de/bitstream/10.24405/536/1/3112-pdf-Koecher2015_thesis_final.pdf.

[123] J. D. De Basabe, M. K. Sen and M. F. Wheeler, "The interior penalty discontinuous Galerkin method for elastic wave propagation: Grid dispersion", *Geophysical Journal International*, vol. 175, no. 1, pp. 83–93, 2008. DOI: 10.1111/j.1365-246X.2008.03915.x.

[124] M. J. Grote, A. Schneebeli and D. Schötzau, "Discontinuous galerkin finite element method for the wave equation", *SIAM Journal on Numerical Analysis*, vol. 44, no. 6, pp. 2408–2431, 2006. DOI: 10.1137/05063194X.

[125] J.-L. Lions and E. Magenes, *Problèmes Aus Limites Non Homogènes et Applications*. Paris: Dunod, 1968, vol. 1, 2, 3.

[126] J. L. Lions, *Optimal Control of Systems Governed by Partial Differential Equations*. Berlin: Springer, 1971.

[127] S. Hussain, F. Schieweck and S. Turek, "Higher order galerkin time discretization for nonstationary incompressible flow", in *Numerical Mathematics and Advanced Applications 2011*, A. Cangiani, R. L. Davidchack, E. Georgoulis, A. N. Gorban, J. Levesley and M. V. Tretyakov, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 509–517.

[128] L. Berger-Vergiat, C. A. Glusa, J. J. Hu, C. Siefert, R. S. Tuminaro, M. Matthias, P. Andrey and W. Tobias, "MueLu User's Guide.", SAND2019-0537, 1491860, 01/01/2019, SAND2019–0537, 1 491 860. DOI: 10.2172/1491860.

[129] W. Varnhorn, *The Stokes Equations* (Mathematical Research v. 76), 1st ed. Berlin: Akademie Verlag, 1994, 153 pp.

[130] H. Joulak and B. Beckermann, "On Gautschi's conjecture for generalized Gauss–Radau and Gauss–Lobatto formulae", *Journal of Computational and Applied Mathematics*, vol. 233, no. 3, pp. 768–774, 12/2009. DOI: 10.1016/j.cam.2009.02.083.

[131] A. Massing, B. Schott and W. Wall, "A stabilized Nitsche cut finite element method for the Oseen problem", *Computer Methods in Applied Mechanics and Engineering*, vol. 328, pp. 262–300, 01/2018. DOI: 10.1016/j.cma.2017.09.003.

[132] E. Burman, "A Penalty-Free Nonsymmetric Nitsche-Type Method for the Weak Imposition of Boundary Conditions", *SIAM Journal on Numerical Analysis*, vol. 50, no. 4, pp. 1959–1981, 01/2012. DOI: 10.1137/10081784X.

[133] M. Anselmann, M. Bause and F. Schieweck, "Pressure approximation in continuous in time variational discretizations of the Stokes system.", *in progress*, 2022.

[134] F. Sonner and T. Richter, "Second Order Pressure Estimates for the Crank–Nicolson Discretization of the Incompressible Navier–Stokes Equations", *SIAM Journal on Numerical Analysis*, vol. 58, no. 1, pp. 375–409, 01/2020. DOI: 10.1137/18M1234813.