TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Mesh Deformation in the Context of Isogeometric Analysis

Alexander Shamanskiy

Vom Fachbereich Mathematik der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades

Doktor der Naturwissenschaften (Doktor rerum naturalium, Dr. rer. nat.)

genehmigte Dissertation.

1. Gutachter: Prof. Dr. Bernd Simeon, Techische Universität Kaiserslautern
2. Gutachter: Prof. Dr. Bert Jüttler, Johannes Kepler Universität Linz

Disputation: 11. September 2020

D 386

Alexander Shamanskiy

Mesh Deformation in the Context of Isogeometric Analysis

Supervised by Prof. Dr. Bernd Simeon

2020

Abstract

This thesis introduces a novel deformation method for computational meshes. It is based on the numerical path following for the equations of nonlinear elasticity. By employing a logarithmic variation of the neo-Hookean hyperelastic material law, the method guarantees that the mesh elements do not become inverted and remain well-shaped. In order to demonstrate the performance of the method, this thesis addresses two areas of active research in isogeometric analysis: volumetric domain parametrization and fluid-structure interaction. The former concerns itself with construction of a parametrization for a given computational domain provided only a parametrization of the domain's boundary. The proposed mesh deformation method gives rise to a novel solution approach to this problem. Within it, the domain parametrization is constructed as a deformed configuration of a simplified domain. In order to obtain the simplified domain, the boundary of the target domain is projected in the L^2-sense onto a coarse NURBS basis. Then, the Coons patch is applied to parametrize the simplified domain. As a range of 2D and 3D examples demonstrates, the mesh deformation approach is able to produce high-quality parametrizations for complex domains where many state-of-the-art methods either fail or become unstable and inefficient. In the context of fluid-structure interaction, the proposed mesh deformation method is applied to robustly update the computational mesh in situations when the fluid domain undergoes large deformations. In comparison to the state-of-the-art mesh update methods, it is able to handle larger deformations and does not result in an eventual reduction of mesh quality. The performance of the method is demonstrated on a classic 2D fluid-structure interaction benchmark reproduced by using an isogeometric partitioned solver with strong coupling.

Zusammenfassung

Diese Dissertation führt eine neue Methode zur Verformung von Berechnungsgittern ein. Sie basiert auf der Fortsetzungsmethode für die Gleichungen der nichtlinearen Elastizität. Durch den Einsatz einer logarithmischen Variante des Neo-Hooke-Modells garantiert diese Methode, dass die Gitterelemente nicht invertiert werdern und wohlgeformt bleiben. Um die Leistungsfähigkeit der Methode zu demonstrieren, behandelt die Dissertation zwei Bereiche der aktuellen Forschung in der isogeometrischen Analysis: volumetrische Gebietsparametrisierung und Fluid-Struktur-Kopplung. Ersterer Bereich beschäftigt sich mit der Aufgabe zu einer gegebenen Parametrisierung des Gebietsrandes eine konsistente Parametrisierung für das Gebietsinnere zu finden. Die vorgeschlagene Methode zur Gittersverformung erlaubt einen neuartigen Ansatz für das Problem. Dabei wird die Gebietsparametrisierung als verformte Konfiguration eines vereinfachten Gebietes konstruiert. Um das vereinfachte Gebiet zu erhalten, wird der Rand des Zielgebietes im L^2-Sinne auf eine gröbere NURBS-Basis projeziert. Daraufhin wird die Coons-Fläche-Methode genuztz, um das vereinfachte Gebiet zu parametrisieren. Wie ein breites Spektrum von 2D und 3D Beispielen demonstriert, ist dieser Ansatz der Gitterverformung in der Lage hochwertige Parametrisierungen für komplexe Gebiete zu erzeugen, auf denen viele häufig genutzte Methoden entweder scheitern oder instabil und ineffizient werden. Im Bereich der Fluid-Struktur-Kopplung wird die vorgeschlagene Methode zur Gitterverformung genutzt, um das Berechnungsgitter auf robuste Weise in Situationen zu aktualisieren, in denen das Gebiet des Fluids große Verformungen erfährt. Im Vergleich zu anderen oft genutzten Verfahren der Gitteraktualisierung ist sie in der Lage größere Verformungen zu behandeln und vermeidet den Verlust der Gitterqualität. Die Performance der Methode wird anhand eines klassischen 2D Benchmarks der Fluid-Struktur-Kopplung illustriert, welcher durch einen isogeometrischen partitioniertem Löser mit Starker Kopplung implementiert wird.

Acknowledgments

As numerous Ph.D. students before me, I have to admit that this thesis could not have come into existence without the help and encouragement of many people. First and foremost, I am eternally grateful to my supervisor Prof. Dr. Bernd Simeon. Before he gave me a chance to join his group, I had come to know him as an inspiring teacher. As a supervisor, he let me try and learn to swim on my own, but was always there whenever I felt like drowning. I deeply enjoyed the atmosphere of honesty and calm professionalism that reigned during our discussions.

I would also like to thank many brilliant scientists whom I met through participating in the European Union project MOTOR. Special thanks go to: Dr. Matthias Möller for leading the project; Prof. Dr. Bert Jüttler and Dr. Angelos Mantzaflaris for introducing me to G+Smo, my primary source of pride and frustration over the last three and a half years; Jochen Hinz for countable but still enormous number of hours spent discussing parametrization techniques; and Prof. Dr. Andreas Brümmer and Matthias Utri for giving me a sneak peek at real engineering.

Of course, I owe many thanks to all of my former and current colleagues at the Felix-Klein-Zentrum: Clarissa Arioli, Felix Dietrich, Mané Harutyunyan, Doris Hemmer-Kolb, Kirsten Höffler, Tobias Mantel, Dennis Merkert, Steffen Plunder and Aydar Uatay. I am especially indebted to my former colleague Dr. Michael Helmut Gfrerer for always being ready to dive headfirst into long discussions on any mathematical topic I happened to bring to him.

I also must thank Prof. Dr. Grigory Panasenko, Dr. Julia Orlik, Prof. Dr. Oleg Iliev and the Fraunhofer ITWM for bringing me into the scientific world and letting me go when I decided it would be best for us all.

At this point, I could have filled the next two pages describing all the ways in which my beautiful wife Sofya supported and inspired me. It is only her explicit prohibition that kept me from doing so. Instead, I would like to thank all of my family members and friends. Although many of them live in Moscow, I could always rely on their support even from far away.

And finally, I thank you, careful reader, for being curious enough to go through the acknowledgments. Your attention to details is what made me do my best when writing this thesis.

Support of this work by the DFG (German Research Council), BMBF (German Federal Ministry of Education and Research) and European Union is gratefully acknowledged.

Stylistic remark

Throughout this work, I extensively use the personal pronoun *I*. The reasoning behind this practice, which some in the mathematical community can find controversial, is two-fold. For one, I wanted to make a process of reading this thesis a little bit more enjoyable for the readers by saving them from excessive use of passive voice. One good way to do that is by writing in first person. But since I am the sole author of this work and have no rights to use the royal *we*, writing in first person means using *I*. For another, writing *I* instead of a vague *we* allows me to emphasize when a certain idea, assumption or opinion belongs to me and not to the scientific community. This last point, I believe, is essential when it comes to writing a thesis. Still, you will find the word *we* used from time to time. Please note that this is not a stylistic inconsistency. In these situations, *we* has a meaning of *the reader and I*. And away we go!

Contents

1 | Introduction

What comes to your mind when you hear the word *innovation*? In year 2020, the answer is very likely to be some combination of machine learning, the Internet of things and mobile solutions. Indeed, these fields have received a lot of attention over the last ten years and have changed the way many people communicate, access information, manage their finances and entertain themselves. In other words, innovation may seem something *digital*. But another less flashy innovation has been happening in the industrial engineering at the same time. Our means of transport, energy sources and buildings have become safer, cheaper and more efficient. Since these high-fidelity engineering areas are heavily regulated and require precision and robustness, progress here tends to happen slower (for example, it took Boeing, a major airplane manufacturer, eight years to develop its new 787 Dreamliner plane and two more years before its first commercial flight [1]) and often escapes public attention. Still, products of industrial engineering form the backbone of our society, and their failure can cause large economic damage and cost people lives.

In order to understand the complexity of modern engineering, one has to distinguish between four product development stages: design, analysis, optimization and manufacturing. In a traditional development process, where physical prototyping plays a key role, a designed prototype is first manufactured, then analyzed, and finally the analysis results are used to optimize the design. If necessary, the sequence is repeated. It goes without saying that this process can be extremely time and cost consuming. Luckily, computer technologies offer a wide range of tools that make it possible to accelerate the development: computer aided design (CAD) for fast geometry modeling; finite element (FEM) and finite volume (FVM) methods for accurate numerical simulation and optimization, usually collectively referred to as computer aided engineering (CAE); and computer aided manufacturing (CAM) for planning and automating the manufacturing process. CAD and CAE form a digital prototyping loop, which renders manufacturing of a physical prototype unnecessary until a final testing stage. Together, these two technologies allow engineers to discover better designs in less time and at lower costs.

Despite being around for more than fifty years, both CAD and CAE remain active fields of research. For one thing, the insatiable demand of industry for efficiency and complexity keeps driving the development of new methods and algorithms. For another, a new trend in CAx technologies seeks to unify CAD and CAE, which historically have evolved separately. As a result, they tend to describe the same object from different, often incompatible, points of view. In particular, CAD systems often use non-uniform rational B-splines (NURBS) to model smooth surfaces of arbitrary shape, whereas in CAE, objects are represented by piecewise linear computational meshes. Conversion between the two formats is far from being fully automated and, according to [2], consumes more than 65% of total digital prototyping time. As an attempt to breach this gap between CAD and CAE, isogeometric analysis (IGA) was introduced in 2005 [3]. The core concept of IGA is to eliminate the conversion step by performing numerical analysis directly on CAD models using the underlying NURBS function spaces in CAE simulations. The idea got traction, and the last fifteen

years have witnessed a research boom in this field. IGA has been successfully applied to many engineering problems, including structural mechanics [4, 5, 6], fluid dynamics [7, 8, 9], fluid-structure interaction [10, 11, 12], electromagnetics [13, 14], topology optimization and phase-field problems [15, 16]. Moreover, further research has shown that IGA offers more accuracy per degree of freedom than standard high-order FEM due to high continuity of NURBS, although rates of convergence are the same [17, 18].

Yet, despite all the progress, the original conversion problem has not been fully solved. While IGA provides a straightforward and natural way to perform numerical simulations on simple tensor-product NURBS objects like single-patch surfaces and volumes, real CAD models still represent a serious challenge. One reason for that are the binary geometric operations that are used in CAD software to create complex geometric objects as a union, intersection or difference of simple ones. From the analysis point of view, these operations result in incomplete, or trimmed, objects which require special treatment [19, 20] or alternative methods to circumvent them [21, 22]. Another reason is that objects in CAD are often described only in terms of their surfaces. While this is not a problem for thin structures like shells and membranes (hence a great success of IGA in these applications [6, 23, 24]), a description of an object's interior is necessary in many other applications, for example, solid body mechanics or fluid dynamics. Construction of the interior description is a challenging problem called volumetric domain parametrization [25, 26]. Unfortunately, existing methods offer imperfect or incomplete solutions to the described problems, and creation of analysis-suitable NURBS models from CAD data still requires lots of time consuming user intervention.

And yet, it would be unfair to say that IGA has done nothing to breach the gap between CAD and CAE. For one, it has triggered a surge of research and software development in the CAE community which is interested in deeper integration with CAD. For another, IGA has started a slow process of raising awareness of the analysis needs among the CAD community. One can imagine designers changing their traditional workflow to create analysis-suitable geometric models from the beginning of the product development process. Unification of CAD and CAE cannot happen over night, but IGA may be setting a right course for incremental progress towards it.

1.1 Scope of this work

This thesis addresses two at first sight unconnected areas of active research in IGA: volumetric domain parametrization and fluid-structure interaction. The former includes methods for constructing NURBS description of an interior of a given computational domain provided only a description of its boundary. This task usually involves two steps: first, the domain is segmented into a number of topological hexahedra, called patches, and then a tensor-product NURBS parametrization for each patch is constructed. I refer to this second step as a single-patch domain parametrization problem. In comparison to the segmentation step, which is currently too complex to be fully automated [26], it has received much more attention from the scientific community. As a result, a number of solution approaches exist which range in their complexity, robustness and flexibility [25, 27, 28, 29].

In this work, I propose a novel *mesh deformation approach* to the single-patch domain parametrization problem, which offers great robustness and flexibility at medium complexity. Based on equations of nonlinear elasticity [30], it constructs a parametrization for the target domain as a deformed configuration of a simplified domain [31]. Simplification is performed by means of L^2-projection of the domain's boundary onto a coarse NURBS basis. After that, the simplified domain can be parametrized using any computationally cheap method, for example, the Coons patch [27]. Finally, a deformation mapping from the simplified domain onto the target domain is computed using the so-called tangential incremental nonlinear elasticity (TINE) method. TINE is an efficient and robust

deformation algorithm that numerically preserves bijectivity of the deformation mapping by employing a logarithmic variation of the neo-Hookean hyperelastic material law. As I demonstrate, the proposed mesh deformation approach is able to construct high-quality parametrizations for complex domains where many existing methods either fail or become unstable and inefficient.

The second area addressed in this thesis, fluid-structure interaction (FSI), studies the interaction of a deformable structure with an internal or surrounding fluid flow. The interaction, when sufficiently strong, causes deformation of both the structure and the domain occupied by the fluid. One of the established approaches to tackle the motion of the fluid domain is based on the so-called arbitrary Lagrangian-Eulerian (ALE) mappings [11, 32]. In its essence, an ALE mapping is a way to update computational mesh in the fluid domain in response to the motion of the structure. What links FSI to the mesh deformation approach for domain parametrization is a similarity between TINE and the incremental linear elasticity (ILE) method (ILE is a state-of-the-art mesh update method used in many FSI applications [33, 34, 35, 36, 37]). A naturally arising idea then is to apply the TINE method in the FSI context. To that end, I have developed an isogeometric partitioned FSI solver and reproduced a classic 2D benchmark problem [38]. On this benchmark, I compare performance of TINE, ILE and other mesh update methods, including those based on harmonic extension.

As it turns out, TINE is as powerful as ILE when it comes to an amount of deformation the methods can handle before the computational mesh becomes invalid, or entangled. Both methods are also much more powerful that other considered methods. However, I show that the ILE method results in residual deformation of the computational mesh which accumulates over the course of simulation. I refer to this effect as accumulated distortion. As I demonstrate, accumulated distortion can significantly affect results of an FSI simulation. At the same time, the TINE method shows no signs of accumulated distortion. Consequently, it allows to perform accurate simulations over longer periods of time. However, there is a price to pay: TINE is roughly twice as computationally expensive as ILE. Although both methods require only solution of a system of linear algebraic equations of the same size per time step, it takes more time to assemble the TINE system.

1.2 Outline

This thesis is organized in five chapters. After this first introductory chapter, Chapter 2 describes basic principles and methods of isogeometric analysis. It introduces NURBS—the core technology of IGA—and gives a short overview of how NURBS can describe computational domains in an analysis-suitable way. Next, the Galerkin isogeometric method is presented using Poisson's equation as an example. Finally, all introduced concepts are illustrated on a numerical example with a special focus on the effect that different types of domain parametrization have on simulation accuracy.

Chapter 3 is devoted to the single-patch domain parametrization problem. After accurately formulating the problem, the chapter proceeds to describing the state-of-the-art solution methods. This includes the spring model, the Coons patch, different optimization-based methods and the elliptic grid generation. Next, an idea of the mesh deformation approach is presented, followed by the domain simplification technique. The bulk of the chapter deals with the TINE algorithm. To explain it in details, a short introduction into the theory of nonlinear elasticity is provided. Then, Newton's method for equations of nonlinear elasticity, its incremental variation and, finally, the TINE method are formulated. The chapter concludes with a comparison of the mesh deformation method against the state-of-the-art methods on a range of 2D and 3D examples.

Chapter 4 focuses on applying the TINE method in the context of fluid-structure interaction. First, it describes the benchmark geometry and the intended FSI simulation scenario. The next two

sections deal with the main components of the FSI system: time-dependent equations of nonlinear elasticity and the incompressible Navier-Stokes equations. Numerical methods for each component are presented, and implementation of the corresponding solvers are validated on partial benchmarks. Then, the ALE formulation of the FSI system of equations is explained, followed by the partitioned solution approach with strong coupling. After that, several state-of-the-art mesh deformation methods are formulated along with an FSI-variation of the TINE method and the Jacobian-based local stiffening. All mesh deformation methods are compared on a simplified FSI-like benchmark. At the end, the ILE and TINE methods are applied to solve the intended FSI benchmark, and the simulation results are compared against the reference values.

Finally, Chapter 5 draws a conclusion to this thesis by summarizing the results and providing an outlook on potential further research directions.

2 | Foundations of Isogeometric Analysis

The research I present in this work has been conducted in the framework of IGA. Depending on the point of view, IGA can be considered a generalization or a special case of FEM. Regardless, the main concept—first introduced in [3]—is to use the same NURBS function space both to describe computational domains and to discretize in space partial differential equations. In this chapter, I introduce B-splines and NURBS which constitute the core technology of IGA. This introduction is based mainly on [39], where detailed definitions, proofs and computer implementations can be found. Additionally, I have drawn inspiration from the following article [40], which provides a very friendly, albeit less formal, overview of NURBS from the perspective of a CAD practitioner. I then demonstrate how NURBS can describe 2D and 3D computational domains, and discuss advantages as well as disadvantages of such a representation. Next, a Galerkin version of IGA is presented—based on [2]—where NURBS play a role of a finite-dimensional function space in which an approximate solution to a given PDE is sought. After illustrating all the introduced concepts on a numerical example, I conclude this chapter by juxtaposing IGA and FEA and express my opinion regarding their relative strengths and weaknesses.

2.1 B-splines and NURBS

2.1.1 B-splines

Our journey into the world of splines begins with the definition of a non-decreasing sequence of *knots* $\xi_i \in \mathbb{R}$, $\xi_i \leqslant \xi_{i+1}$, $i = 1, \ldots, k$ which form a *knot vector* $\Xi = \{\xi_1, \ldots, \xi_k\}$. Note that the knots in Ξ do not need to have unique values. Let us say that a knot value has a *multiplicity* m if it is shared by exactly m knots in Ξ.

Given the knot vector Ξ, we can define the ith *B-spline*[1] *basis function* of degree p—denoted by $B_i^p(\xi)$—by a recursive formula of de Boor as

$$B_i^0(\xi) = \begin{cases} 1 & \text{if } \xi \in [\xi_i, \xi_{i+1}), \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

$$B_i^p(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_i^{p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1}^{p-1}(\xi). \tag{2.2}$$

Due to the knot multiplicity, the quotient $\frac{0}{0}$ may appear which is formally set to 0.

For a given knot vector Ξ and degree p, formulas (2.1–2.2) define $n = k - p - 1$ B-spline basis functions which I collectively refer to as a *B-spline basis*. There are several important properties to be mentioned.

- B-spline basis functions are polynomials of degree p between every two distinct knots; their continuity at knot ξ_i is C^{p-m}, where m is of the knot multiplicity.

[1] B in B-spline probably means **b**asis.

- B-spline basis functions are linearly independent and form a basis in the space of degree-p splines with the continuity implied by Ξ.
- B-spline basis functions are non-negative: $B_i^p(\xi) \geqslant 0 \; \forall \xi$.
- B-spline basis functions have local support: $\text{supp}(B_i^p(\xi)) = [\xi_i, \xi_{i+p+1})$. Between every two distinct knots exactly $p+1$ functions are non-zero.
- B-spline basis functions form a partition of unity within $[\xi_{p+1}, \xi_{k-p})$, that is, $\sum_{i=1}^{n} B_i^p(\xi) \equiv 1 \; \forall \xi \in [\xi_{p+1}, \xi_{k-p})$.

The primary use of B-spline basis functions is to construct smooth objects of arbitrary shape. When combined with a set of n *control points* $\mathbf{c}_i \in \mathbb{R}^d$, $d = 2$ or 3, they define a *B-spline curve* $\boldsymbol{\gamma} : [\xi_{p+1}, \xi_{k-p}) \to \mathbb{R}^d$ as

$$\boldsymbol{\gamma}(\xi) = \sum_{i=1}^{n} \mathbf{c}_i B_i^p(\xi). \tag{2.3}$$

One can interpret a B-spline curve as a smooth blending, or approximation, of given control points. Control points of a B-spline curve form a *control polygon*. From the non-negativity and the partition-of-unity properties of B-spline basis functions, it follows that the B-spline curve lies in the convex hull of its control points.

Let us study a couple of examples. Figure 2.1 depicts a quadratic B-spline basis generated with a knot vector $\{0, 1, 2, 3, 4, 4, 5, 6, 7\}$. The basis functions are C^1-continuous everywhere but at $\xi = 4$. Due to multiplicity 2 of the knot value $\xi = 4$, functions[2] B_3, B_4, and B_5 are only C^0-continuous. Figure 2.1 also shows a B-spline curve generated using this basis, as well the corresponding control points and the control polygon. Note the C^0-kink caused by the knot multiplicity. Another prominent property of the curve is its floating in space endpoints; that is, the endpoints do not coincide with any of the control points. Specifically, they do not coincide with the first and the last control points. The reason is that more than one basis function assumes a non-zero value at $\xi = 2$ and $\xi = 5$ (the beginning and the end of the parameter interval where the curve is defined). Consequently, each endpoint is a linear combination of the neighboring control points. In practice, however, one would like to directly control the position of the endpoints, and most often one would like them to coincide with the first and the last control points.

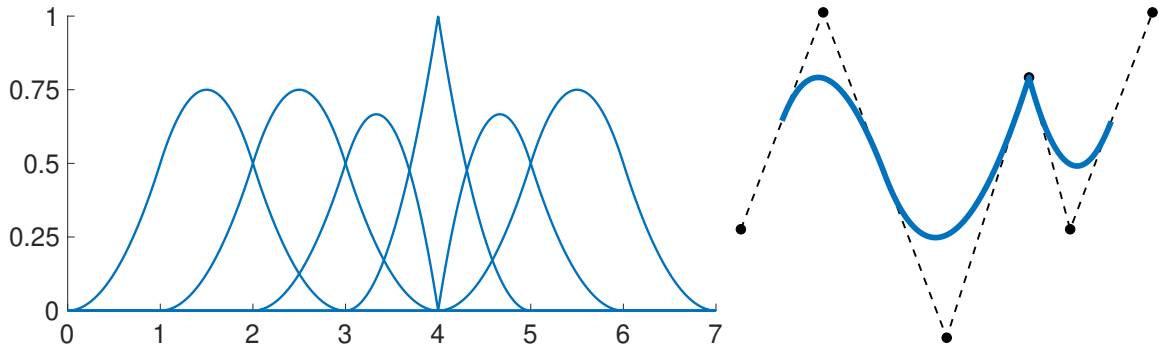

Figure 2.1. Quadratic B-spline basis (left) and a B-spline curve generated using this basis, control points and a control polygon (right).

The issue of floating in space endpoints can be solved by exploiting the knot multiplicity. Consider a knot vector $\{0, 0, 0, 0.2, 0.4, 0.6, 0.8, 1, 1, 1\}$. Figure 2.2 shows the corresponding quadratic

[2]Where it is not necessary, I omit the superscript denoting a polynomial degree of B-spline basis functions.

B-spline basis. Due to multiplicity three of the knot value $\xi = 0$, all basis functions except for the first one are zero at $\xi = 0$; the first basis function assumes a value of 1. The same happens at $\xi = 1$, where only the last basis function is non-zero. As a result, the first and the last basis functions have an exclusive control over the endpoints of the B-spline curve generated using this basis. Consequently, the endpoints of the curve coincide with the first and the last control points, see Figure 2.2. The knot vectors with multiplicity $p + 1$ of the first and the last knots are called *open knot vectors*[3], and I use only such knot vectors in this work. Note also that the considered knot vector lies in the unit interval $[0, 1]$. Although this is knot[4] necessary, for convenience I always use knot vectors from the unit interval.

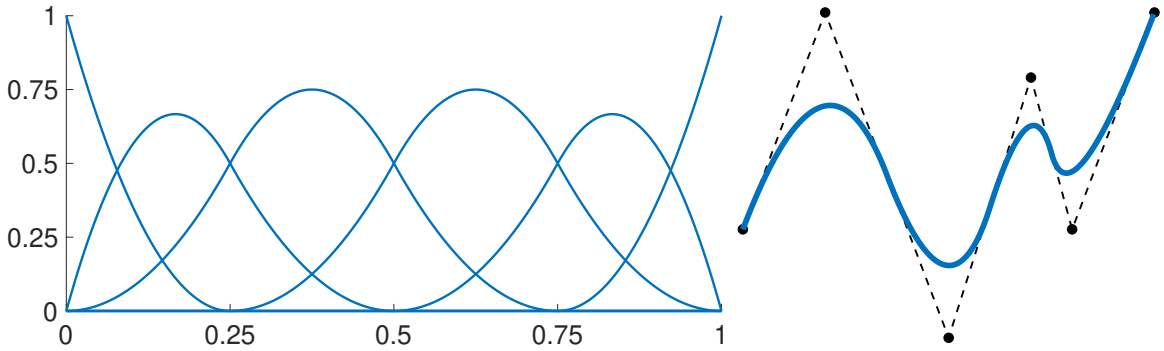

Figure 2.2. Quadratic B-spline basis with an open knot vector (left) and a B-spline curve generated using this basis, control points and a control polygon (right).

2.1.2 Refinement

One of the beautiful properties of B-splines is that one can enrich the underlying basis of a B-spline curve without changing the geometry or the parametrization of this curve. Two key algorithms exist: *knot insertion* and *degree elevation*. As the name suggests, knot insertion allows to add a (not necessarily new) knot to the knot vector of the curve, see Figure 2.3. As a result, the number of basis functions and control points increases by one. Although a new set of control points has to be constructed based on the old set, only p new control points have to be computed, whereas the rest remain the same. Therefore, knot insertion is a local operation. For a detailed description of knot insertion algorithm, I refer to [39].

The fact that knot insertion does not alter geometry of the curve motivates its two main applications. The first is *h-refinement* which involves inserting new knots between every two unique knot values. Often, a value $\xi_{new} = (\xi_i + \xi_{i+1})/2$ is inserted, decreasing the distance between every two unique knots by half. This grants more control over the shape of the curve as well as increases accuracy of the numerical methods built using the corresponding basis. Another application of knot insertion is to intentionally introduce points with C^0-continuity. This is useful for splitting a B-spline curve into two independent curves.

Another refinement algorithm, degree elevation, increases the polynomial degree of a B-spline curve by one. It does so by splitting the curve into Bézier segments using knot insertion, increasing the polynomial degree of every segment, and then removing the excessive knots. Again, I refer to [39] for the details. After the procedure, multiplicity of each knot in the knot vector is increased by one. Note that the inverse algorithms for both knot insertion and degree elevation, namely knot

[3]The term *open* means that the corresponding B-spline curve may not close on itself. On the other hand, usual knot vectors, like the one in Figure 2.1, are often used to model smoothly closed curves.

[4]Pun intended.

removal and degree reduction, can be constructed but do not preserve the geometry exactly unless the curve happens to lie in a span of the coarser basis.

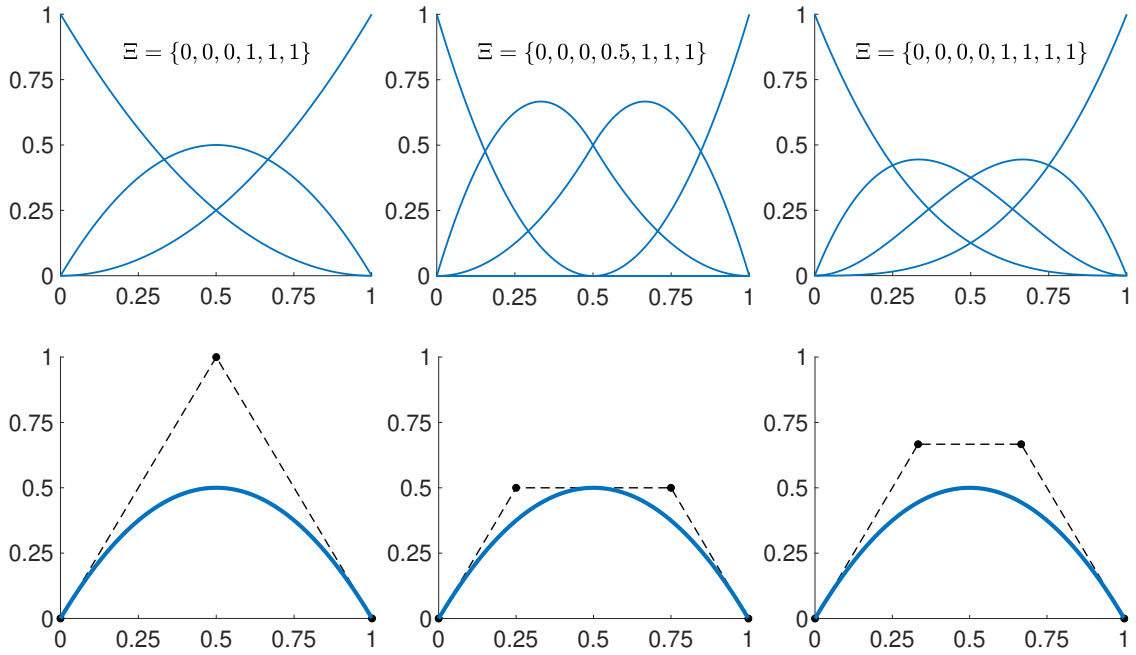

Figure 2.3. Knot insertion and degree elevation: original quadratic B-spline curve before refinement and the corresponding B-spline basis (left); the basis and the curve after inserting a knot $\xi_{new} = 0.5$ (middle); the basis and the curve after degree elevation from quadratic to cubic (right).

The order in which h-refinement (or knot insertion in particular) and degree elevation[5] are applied affects the output. In particular, degree elevation applied after knot insertion results in a larger number of points of reduced continuity than if the order is reversed, see Figure 2.4. This motivates the concept of the so-called *k-refinement*. According to it, if one wishes to apply both h-refinement and degree elevation, then degree elevation should be applied first. Note that although k-refinement results in the smoothest B-spline basis, it preserves all existing points of reduced continuity.

2.1.3 NURBS

Despite great flexibility of B-spline curves, one issue remains: they cannot represent conic sections. *NURBS*[6] is a generalization of B-splines which fixes this issue. Geometrically, a *NURBS curve* $\mathbf{C}(\xi)$ in $\mathbb{R}^d$ is defined as a projection of a B-spline curve $\hat{\mathbf{C}}(\xi)$ from $\mathbb{R}^{d+1}$: each point of the curve $\hat{\mathbf{C}}(\xi)$ is projected onto the plane $x_{d+1} = 1$ by a ray through the origin. This is accomplished by defining each curve component by

$$\big(\mathbf{C}(\xi)\big)_j = \frac{\big(\hat{\mathbf{C}}(\xi)\big)_j}{W(\xi)} = \frac{\sum_{i=1}^{n} \big(\hat{\mathbf{c}}_i\big)_j B_i(\xi)}{\sum_{i=1}^{n} w_i B_i(\xi)}, \qquad j = 1, \dots, d, \tag{2.4}$$

[5] In order to complete the picture, it is worth mentioning that degree elevation would be called p-refinement in the FEM community.

[6] **N**on-**u**niform **r**ational **B**-**s**plines. NU in NURBS refers to non-uniform knot vectors, so only R is new with respect to the previously defined B-splines.

where $\hat{\mathbf{c}}_i \in \mathbb{R}^{d+1}$ are control points of the B-spline curve, and $w_i = (\hat{\mathbf{c}}_i)_{d+1}$ are the last components of the control points which are called *weights* in the context of NURBS. Since the weights are assumed to be positive, the *weight function* $W(\xi)$ is positive as well due to the non-negativity property, and the projection is well defined. All the refinement algorithms for the NURBS curve $\mathbf{C}(\xi)$ are implemented by applying them to the corresponding B-spline $\hat{\mathbf{C}}(\xi)$.

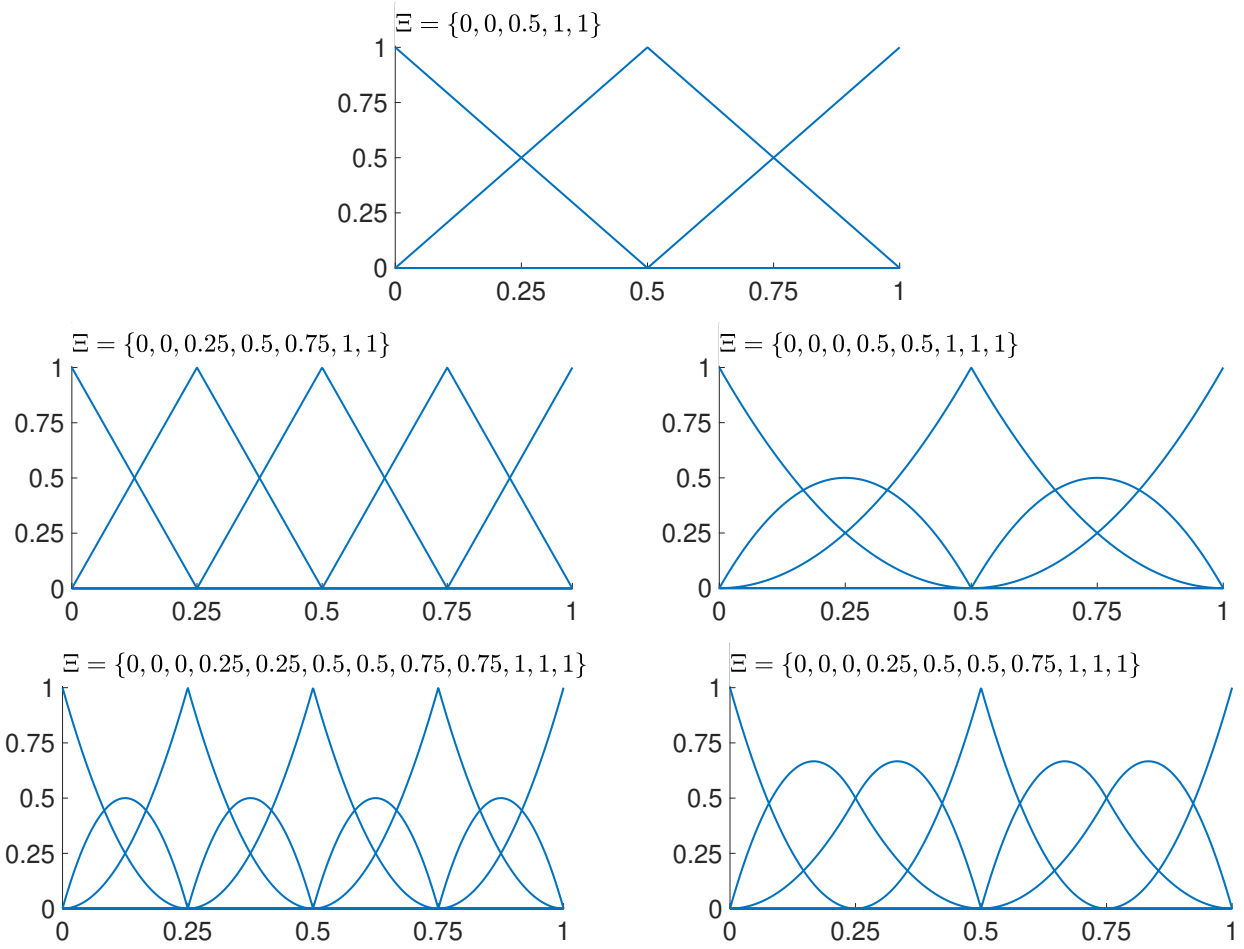

Figure 2.4. k-refinement: original linear B-spline basis (top); result of applying h-refinement followed by degree elevation (left column); degree elevation first, then h-refinement (right column).

Although the geometric definition gives us some intuition about NURBS, in practice one would like to be blissfully ignorant of the projective nature of NURBS. To that end, let us define *NURBS basis functions* $N_i(\xi)$ and *(projected) control points* $\mathbf{c}_i \in \mathbb{R}^d$ as

$$N_i(\xi) = \frac{w_i B_i(\xi)}{\sum_{i=1}^{n} w_i B_i(\xi)}, \qquad (\mathbf{c}_i)_j = (\hat{\mathbf{c}}_i)_j / w_i, \qquad j = 1, \dots, d, \tag{2.5}$$

so that the curve $\mathbf{C}(\xi)$ can be rewritten as

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} \mathbf{c}_i N_i(\xi). \tag{2.6}$$

Unlike the projective definition (2.4), this representation of a NURBS curve is simple and coincides with the representation of a B-spline curve (2.3). We can now choose control points $\mathbf{c}_i$ in the same

manner as we would do for a B-spline curve, and weights w_i are chosen independently to alter the shape of the resulting NURBS curve. Although from the algebraic point of view weights do not have an explicit geometric interpretation, the following rule of thumb applies: the larger the weight is, the stronger the resulting NURBS curve is pulled locally towards the corresponding control point. One final remark: note that if all weights have the same value, then $N_i(\xi) = B_i(\xi)$, and thus NURBS become B-splines again. Hence in what follows, I only use the term NURBS unless it is necessary to explicitly specify that we are dealing with B-splines.

Let us turn own attention back to the conic sections; the following method can be used to generate conic arcs:

- an arc is modeled by a quadratic NURBS curve with three control points;
- the first and the last control points are the endpoints of the arc; location of the inner control point defines the shape of the curve;
- weights of the first and the last control points are 1;
- a weight less than 1 for the inner control point generates an ellipse; a weight equal to 1 generates a parabola; a weight larger than 1 generates a hyperbola;
- the knot vector is $\{0, 0, 0, 1, 1, 1\}$.

The most common conic arc used in modeling is a circular arc. Since a circle is a special case of an ellipse, the previously described method can be applied with the following specifications:

- the legs of the control polygon are of the same length;
- the cord connecting the first and the last control points meets each leg at angle θ equal to half the angular extent of the desired arc;
- the weight of the inner control point equals $\cos\theta$.

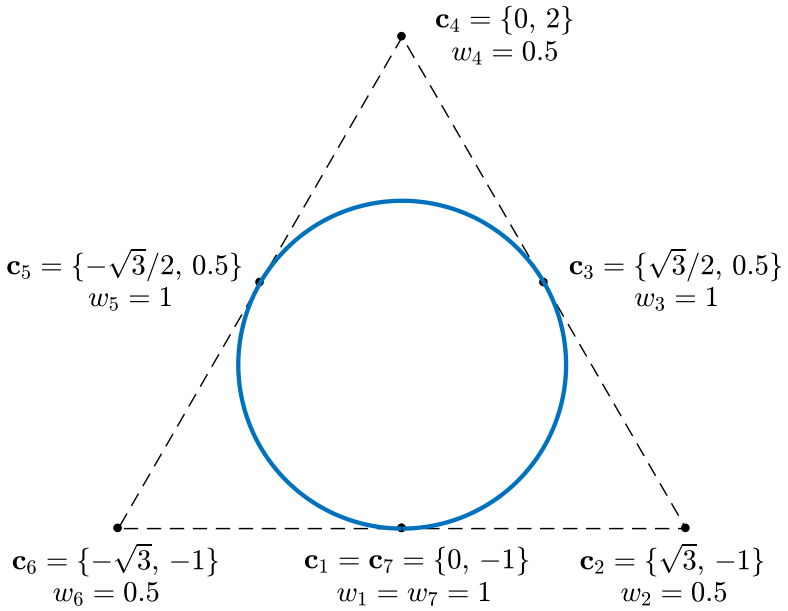

Figure 2.5. Circle modeled by three 120° circular arcs joined into one quadratic NURBS curve with a knot vector $\{0, 0, 0, 1/3, 1/3, 2/3, 2/3, 1, 1, 1\}$.

The described method can only produce circular arcs less then 180° in angular extent. To model a complete circle, one usually combines three 120° or four 90° arcs. Figure 2.5 presents an example of a radius one circle with its center at (0,0) modeled by three 120° circular arcs[7]. These three arcs are represented by a single quadratic NURBS curve with the knot vector $\{0,0,0,1/3,1/3,2/3,2/3,1,1,1\}$ and the combined control points.

2.1.4 Tensor-product NURBS

So far, we have only considered NURBS curves $\boldsymbol{\gamma}(\xi) : [0,1] \to \mathbb{R}^d$, which are univariate NURBS objects. However, since our goal is to construct computational domains using NURBS, we need to define multivariate NURBS objects as well. The most popular option is to use *tensor-product NURBS*. Given two knot vectors Ξ and H from $[0,1]$ and two corresponding sets of NURBS basis functions $N_i(\xi)$, $i = 1,\dots,n$ and $M_j(\eta)$, $j = 1,\dots,m$, a bivariate tensor-product NURBS function $\mathbf{G}(\xi,\eta) : [0,1]^2 \to \mathbb{R}^d$ is defined as

$$\mathbf{G}(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{c}_{ij} N_i(\xi) M_j(\eta), \tag{2.7}$$

where $\mathbf{c}_{ij} \in \mathbb{R}^d$ form a $n \times m$ array of control points. Image of mapping $\mathbf{G}$ is some domain $\Omega \subset \mathbb{R}^d$, and $\mathbf{G}$ can be interpreted as a *parametrization* of this domain. Bivariate tensor-product NURBS allow us to construct planar domains in $\mathbb{R}^2$ and two-dimensional hypersurfaces in $\mathbb{R}^3$, see Figure 2.6.

Note that the objects in Figure 2.6 have two co-existing grid-like structures associated to them: one formed by the control points, called a *control net*, and another formed by the isoparametric lines corresponding to unique knot values from the knot vectors Ξ and H, which I refer to as an *isogeometric mesh*. In its essence, the isogeometric mesh is very similar to computational meshes used in the standard finite element analysis. When restricted to any of the isogeometric mesh elements, mapping $\mathbf{G}$ is just a multivariate polynomial function. The crucial distinction from FEM is that $\mathbf{G}$ is smooth between the elements. Therefore, elements of the isogeometric mesh are connected smoothly, whereas in FEM, the connection is only C^0-continuous.

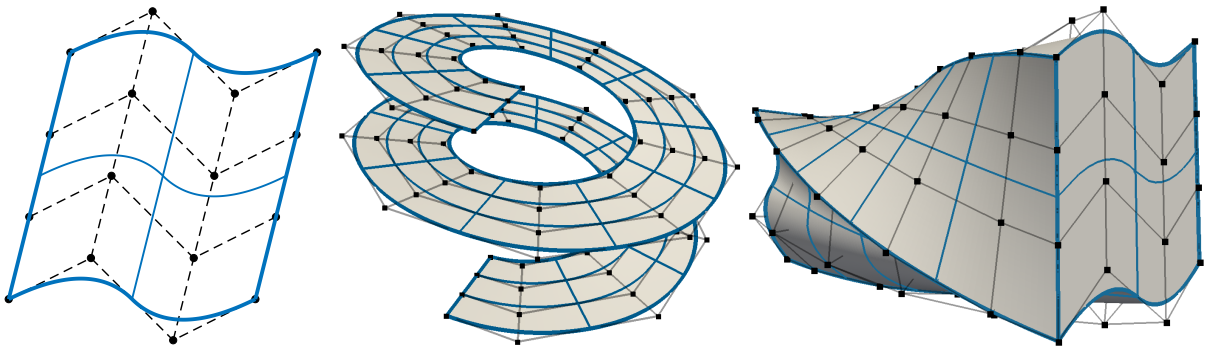

Figure 2.6. Examples of tensor-product NURBS objects: a planar domain (left), a two-dimensional surface in $\mathbb{R}^3$ (middle) and a volumetric domain. Control points and control nets are shown in black, isogeometric meshes in blue.

When working with tensor-product NURBS, it is convenient to have an ability to distinguish

[7] This is the smallest NURBS model of a circle.

between different *sides* of the domain. To that end, I use the cardinal directions: south for $\mathbf{G}(\xi, 0)$, north for $\mathbf{G}(\xi, 1)$, west for $\mathbf{G}(0, \eta)$ and east for $\mathbf{G}(1, \eta)$.

By introducing yet another set of NURBS basis functions $L_k(\zeta)$, $k = 1, \ldots, l$ with a knot vector $\mathrm{Z} \subset [0, 1]$, we can define a trivariate tensor-product NURBS function $\mathbf{G}(\xi, \eta, \zeta) : [0, 1]^3 \to \mathbb{R}^d$ as

$$\mathbf{G}(\xi, \eta, \zeta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} \mathbf{c}_{ijk} N_i(\xi) M_j(\eta) L_k(\zeta), \tag{2.8}$$

where control points $\mathbf{c}_{ijk} \in \mathbb{R}^d$ form a $n \times m \times l$ array. Trivariate tensor-product NURBS let us construct volumetric domains $[0, 1]^3 \to \mathbb{R}^3$, see Figure 2.6. In what follows, I only consider planar and volumetric domains and do not work with NURBS surfaces.

Compact tensor-product NURBS notation

In order to treat both cases of $d = 2$ and $d = 3$ simultaneously and to shorten the notation, I use a dimension-agnostic notation. I denote a point in the parametric domain $[0, 1]^d$ by $\boldsymbol{\xi}$ and a tensor-product NURBS basis function $[0, 1]^d \to \mathbb{R}$ by $N_i(\boldsymbol{\xi})$. For $d = 2$ and $d = 3$, $N_i(\boldsymbol{\xi})$ is defined as

$$N_i(\boldsymbol{\xi}) = N_i(\xi) M_j(\eta), \tag{2.9}$$
$$N_i(\boldsymbol{\xi}) = N_i(\xi) M_j(\eta) L_k(\zeta), \tag{2.10}$$

respectively, with the following renumbering:

$$i = (j - 1)n + i, \tag{2.11}$$
$$i = (k - 1)nm + (j - 1)n + i. \tag{2.12}$$

The total number of basis functions is $n = n \times m$ for $d = 2$ and $n = n \times m \times l$ for $d = 3$. With this notation, formulas (2.7) and (2.8) for tensor-product NURBS shorten to

$$\mathbf{G}(\boldsymbol{\xi}) = \sum_{i=1}^{n} \mathbf{c}_i N_i(\boldsymbol{\xi}). \tag{2.13}$$

Both the extended and the shortened notations can be used interchangeably, which, hopefully, causes no confusion.

It is also convenient to distinguish between the interior and the boundary basis functions. I call $N_i(\boldsymbol{\xi})$ an interior basis function if it is zero on the boundary of $[0, 1]^d$. I call $N_i(\boldsymbol{\xi})$ a boundary basis function if it is not an interior one. I use $\mathcal{I}$ and $\mathcal{B}$ to denote the sets of indices corresponding to the interior and the boundary basis functions, respectively. The two sets do not overlap, so (2.13) can be rewritten as

$$\mathbf{G}(\boldsymbol{\xi}) = \sum_{i=\mathcal{I}} \mathbf{c}_i N_i(\boldsymbol{\xi}) + \sum_{i=\mathcal{B}} \mathbf{c}_i N_i(\boldsymbol{\xi}). \tag{2.14}$$

Note that since open knot vectors are used to construct $\mathbf{G}$, only the first and the last basis functions in each parametric direction belong to the boundary set. In other words, the layer of boundary basis functions has a thickness of one.

2.2 Domain parametrization with NURBS

The biggest advantage of tensor-product NURBS, and at the same time their greatest weakness, is their simplicity. This simplicity stems from a natural "structuredness" inherent to tensor product and opens the door for some straightforward and efficient implementations. However, it also puts a restriction on what output tensor-product NURBS can produce. So in practice, tensor-product NURBS are augmented by additional techniques or even replaced by alternative spline technologies. In the following, I give a short overview of various approaches to geometry modeling in IGA and outline a general pipeline I stick to when working with tensor-product NURBS.

2.2.1 Multi-patch models

Due to their structure, tensor-product NURBS cannot produce topologically complex objects. If mapping $\mathbf{G}$ is a diffeomorphism[8], its image Ω is diffeomorphic to a unit square (for $d = 2$) or to a unit cube (for $d = 3$). Therefore, Ω has zero holes and four sharp corners in 2D or eight in 3D. By relaxing the bijectivity requirement on the boundary, domains with holes can be constructed, see Figure 2.7. Additionally, domains with different number of corners can be constructed if the smoothness requirement on the boundary is relaxed, see Figure 2.7 again. Still, both options are essentially tricks; the former is not universally applicable whereas the latter leads to a worse performance of numerical methods built upon it.

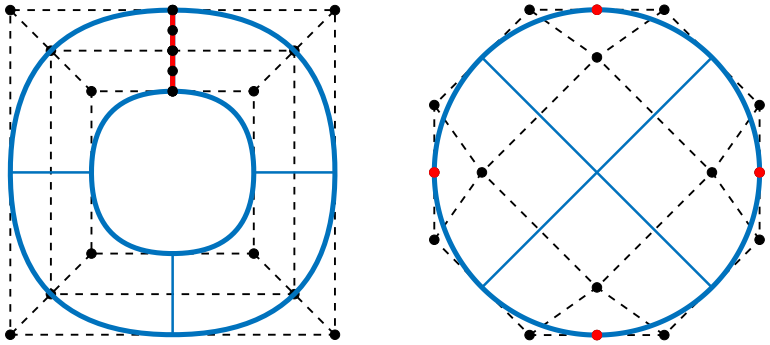

Figure 2.7. Left: an example of a domain with one hole constructed by "gluing" together the west and the east sides of the domain along the red line; the parametrization is not uniquely invertible along the gluing line. Right: an example of a domain with no corners; the inverse mapping is not differentiable at the red points.

A not-so-elegant but robust way to construct topologically non-trivial domains is to use the so-called *multi-patch* (MP) *approach*. In accordance to it, domain Ω is constructed as a union of several subdomains[9] Ω_i called *patches*,

$$\bar{\Omega} = \bigcup_{i=1}^{k} \bar{\Omega}_i. \tag{2.15}$$

Each patch is a tensor-product NURBS object. The most natural and simple in practice case is when the patches have *matching interfaces*, which is characterized by the following conditions:

- the patches do not overlap: $\Omega_i \cap \Omega_j = \emptyset \ \forall i \neq j$;

[8] A diffeomorphism is a bijective mapping such that it and its inverse are differentiable.
[9] Here, $\bar{\Omega}$ denotes a closure of the domain Ω, and $\partial\Omega$ denotes its boundary.

- if two patches Ω_i and Ω_j are adjacent, that is, $\partial\Omega_i \cap \partial\Omega_j = \Gamma_{ij} \neq 0$, then the *interface* Γ_{ij} consists of entire sides of the patches;
- parametrizations $\mathbf{G}_i$ and $\mathbf{G}_j$ of the adjacent patches Ω_i and Ω_j coincide on Γ_{ij}.

In practice, matching interfaces mean two patches share the NURBS basis and the control points at the boundary. Figure 2.8 shows an example of a multi-patch domain with matching interfaces.

One disadvantage of the matching-interface property is its non-local nature. If three or more patches in a row have matching interfaces, parametrizations in the corresponding direction for all patches have to coincide. This chain effect can place a rather inconvenient restriction on the freedom in geometric modeling. Because of that, the de facto industrial standard in CAD are MP models with *non-matching interfaces*. This can mean either that the patches coincide geometrically at the neighboring sides but are free to have different parametrizations, or that they do not coincide geometrically at all and may have small overlaps or gaps in between, see Figure 2.8. The latter is a common approach in CAD modeling which results in the so-called *non-watertight* models. From the perspective of numerical analysis, however, the non-matching interfaces are not desirable since they require special treatment to connect patches [41, 42]. In this work, I opt for easier numerical methods instead of easier geometric modeling and only use MP models with matching interfaces.

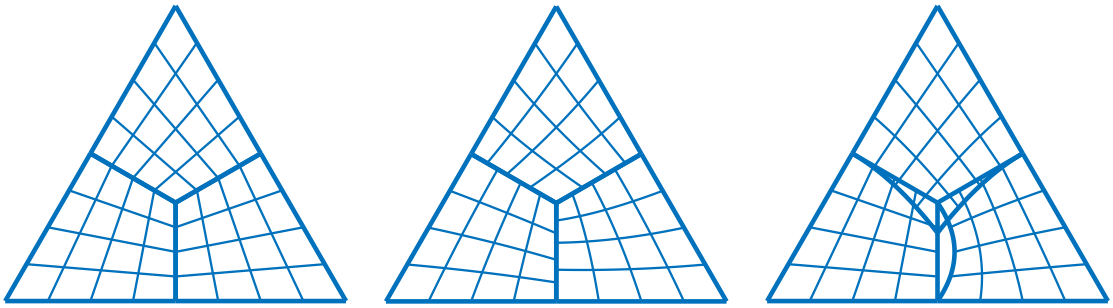

Figure 2.8. Examples of multi-patch models. Bold blue lines show the borders of each patch. Left: matching interfaces. Center: non-matching interfaces with matching geometry. Right: non-matching geometry (an exaggerated non-watertight model).

Although I exclusively use multi-patch models in this work, I would also like to mention that there exist alternative spline technologies that allow to overcome the topological restrictions of tensor-product NURBS. These include—but are not limited to—*Catmull-Clark subdivision surfaces* [43], *triangular splines* [44] and *U-splines*[10] [45]. All mentioned spline technologies have successfully been applied to in the context of IGA; however, they are not as commonly used as the multi-patch tensor-product NURBS approach.

2.2.2 Domain parametrization problem: overview

The MP approach is a powerful and convenient tool for creating geometric objects of arbitrary shape and complexity. In practice, however, one often has to solve a reverse problem: given a domain Ω, how to construct a (possibly multi-patch) tensor-product NURBS model of it? I refer to this a *domain parametrization problem*. Its solution involves a *segmentation step*, where a multi-patch structure for the domain is created, followed by the construction of a tensor-product NURBS parametrization for each patch, see Figure 2.9. In that, the domain parametrization problem is similar to reverse engineering [46], where a geometric model of an existing object is reconstructed

[10] **U**nstructured splines.

from a set of measured surface data, for example, a point cloud. The reconstructed geometric model is often a multi-patch tensor-product NURBS description of the object's surface. In the domain parametrization problem, however, only the description of the domain's boundary is available, whereas the parametrization has to be constructed for the interior of the domain as well, making it more akin to the problem of quadrilateral and hexahedral mesh generation [47].

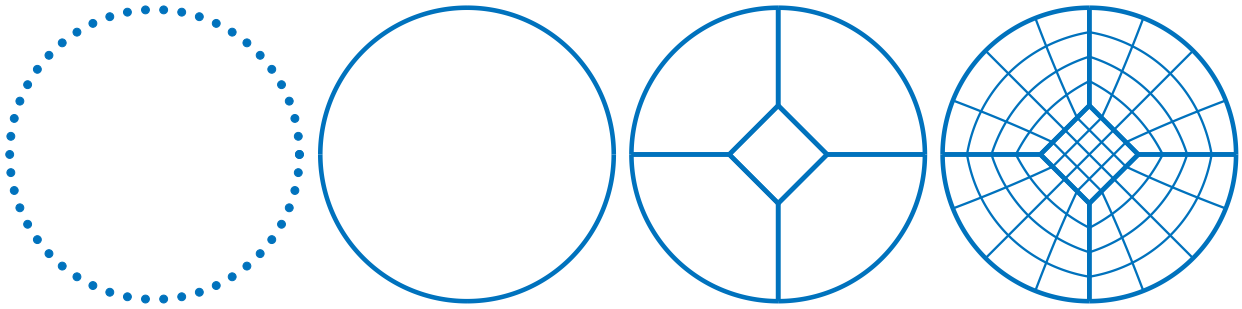

Figure 2.9. Domain parametrization pipeline. From left to right: point cloud or any other non-NURBS representation of the domain's boundary; NURBS representation of the domain's boundary; multi-patch structure of the domain's parametrization; multi-patch NURBS parametrization of the domain.

Of the two steps constituting the parametrization problem, the segmentation step is the least susceptible to a full automation. It turns out that to split a domain into a reasonably small number of well-shaped patches in a way a human would do is not a trivial task. Several fully automated methods are available, including the medial axis decomposition[11][48, 49] and the cross-field method [50]. However, these methods tend to be complex and, occasionally, unstable. A commonly accepted approach to segmentation is to employ interactive semi-automated methods which make use of some manually provided a priori knowledge about the domain. In [26], a systematic segmentation pipeline is described which implements this approach. It is worth noting that the segmentation step may also include a reverse engineering sub-step for acquiring a suitable NURBS description of the domain's boundary which facilitates or dictates the segmentation. This may be necessary if a given NURBS boundary description is not clean or watertight, or if the boundary is given in an alternative geometry format, say, a point cloud or a triangulated surface. In this work, I perform segmentation manually.

In comparison to the segmentation step, the single-patch parametrization step is straightforward and has received much more attention from the scientific community. It has a well-defined standard setting: given four boundary NURBS curves in the planar case or six boundary NURBS surfaces in the volumetric case, construct a tensor-product NURBS function which matches the boundary description. As we will see, this boils down to allocation of the interior control points in a right way. In Chapter 3, I describe of the most popular state-of-the-art solution approaches to the single-patch parametrization step as well as propose a novel approach which yields superior results for complex domains.

Alternative parametrization strategies

The described above domain parametrization pipeline aims at converting an arbitrary CAD object into an IGA-suitable multi-patch parametrization which explicitly resolves the object's boundary. At the current state of technology, this pipeline is far from being fully automated and requires

[11] Also known as skeleton decomposition.

a hefty deal of manual work (mostly at the segmentation step). For the sake of completeness, I would like to mention alternative approaches to treating complex CAD objects in IGA. One of these approaches, the *immersed boundary* method [22], overlays the domain with a Cartesian grid and resolves the boundary implicitly by integrating over the cut elements. The Cartesian grid is created as a single tensor-product NURBS patch, and one can easily treat domains with arbitrary topology using this method. Figure 2.10 presents a simplified example of this approach. Another method uses parametrizations with *trimmed patches* [19, 20]. The idea is to mimic the CAD modeling process where Boolean operations, such as union or subtraction, are used to create complex objects. Instead of constructing a new multi-patch parametrization with a boundary-fitted mesh for the object, the existing patches are used, and the implied interfaces are treated implicitly at the integration step, see Figure 2.10. Both methods have to deal with cut elements along the domain boundary, so carefully designed integration rules are required to avoid numerical instabilities.

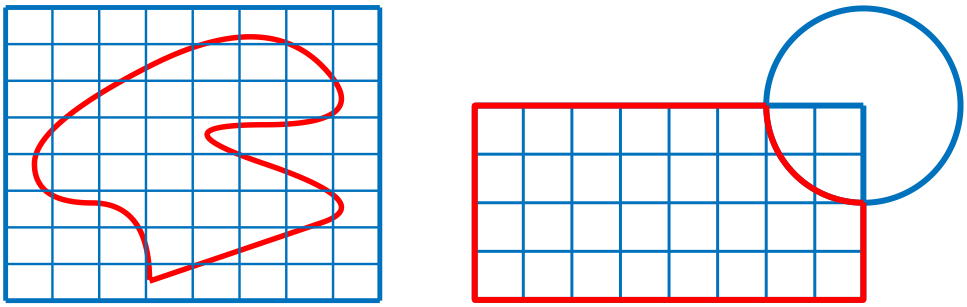

Figure 2.10. Left: immersed boundary method. Right: trimmed patch constructed by subtracting of a circle from a rectangle. Computational domains' boundaries are marked with red.

2.2.3 Local refinement

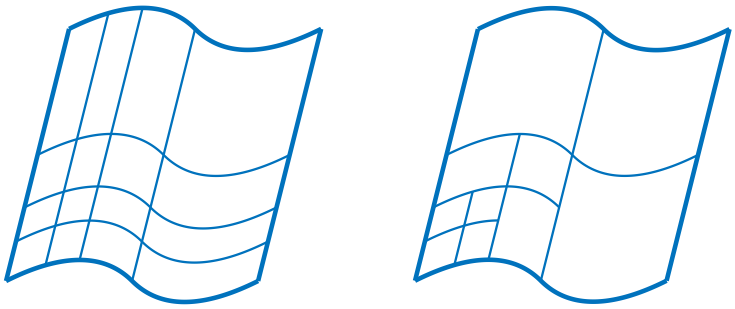

Figure 2.11. Left: propagating local refinement with tensor-product NURBS. Right: true local refinement with THB-splines.

Another problem of the tensor-product NURBS technology is that it does not support *local refinement*. The idea of local refinement is to increase the simulation resolution in a certain area by reducing the size of mesh elements in this area only. In this way, local refinement allows to improve the simulation accuracy where it is necessary without significantly increasing the total number of degrees of freedom. This idea can be naively implemented in IGA by means of knot insertion. However, if a new knot value is inserted in one parametric direction, then a whole new isoparametric line appears which splits an entire row of elements. This effect defies the idea of introducing only local changes to the mesh, see Figure 2.11. To overcome this problem, several generalizations of NURBS

were developed; among them, *THB-splines*[12] [51, 52]. A THB-basis is constructed by combining subsets of bases from different levels of refinement. The basis functions are then slightly modified (truncated) so that the resulting basis possesses such properties as the non-negativity, partition-of-unity and linear independence. An example of local refinement with THB-splines is depicted in Figure 2.11. Alternative spline technologies which support local refinement are *T-splines*[13] [53, 54], *LR B-splines*[14] [55] and already mention U-splines and triangular-based splines.

2.3 Isogeometric Galerkin method

In the previous sections, I have introduced NURBS and described various ways they can be used to model computational domains. In what follows, I demonstrate how NURBS can be applied to solve PDEs using Galerkin method. In order to keep the discussion concise, I do that on an example of Poisson's equation. However, the described procedure is applicable to any linear elliptic PDE with the properties of coercivity and continuity. After that, I present a numerical example which studies the effect that domain parametrizations have on accuracy of the numerical simulation.

2.3.1 Model problem: Poisson's equation

Let $\Omega \subset \mathbb{R}^d$ be a computational domain where Poisson's equation is to be solved:

$$-\Delta_{\mathbf{x}} u = f \text{ in } \Omega. \tag{2.16}$$

Here, function $f : \Omega \to \mathbb{R}$ is a given source term, $u : \Omega \to \mathbb{R}$ is an unknown solution and $\Delta_{\mathbf{x}}$ is the Laplace operator. The subscript $\mathbf{x}$ indicates differentiation with respect to the spacial coordinates $\mathbf{x} \in \Omega$ rather than the parametric coordinates $\boldsymbol{\xi}$. The domain's boundary $\partial\Omega$ is split into a Dirichlet part $\partial\Omega_{\mathcal{D}}$ and a Neumann part $\partial\Omega_{\mathcal{N}}$, where the corresponding boundary conditions are set:

$$u = u_{\mathcal{D}} \text{ on } \partial\Omega_{\mathcal{D}}, \tag{2.17}$$

$$\nabla_{\mathbf{x}} u \cdot \mathbf{n} = g \text{ on } \partial\Omega_{\mathcal{N}}. \tag{2.18}$$

A function $u \in C^2(\Omega) \cap C^0(\bar{\Omega})$ is called a classical solution if it satisfies the system (2.16–2.18).

Recall that a Sobolev space of order r is defined by

$$H^r(\Omega) = \{w \mid \mathcal{D}^{\boldsymbol{\alpha}} w \in L^2(\Omega), |\boldsymbol{\alpha}| \leqslant r\}, \tag{2.19}$$

where $\mathcal{D}^{\boldsymbol{\alpha}}(\cdot)$ is a differentiation operator in the sense of distributions. In order to apply the *isogeometric Galerkin method*, one has to transform the strong form (2.16–2.18) into a weak form, also known as a variational formulation. To that end, let us define the solution and the weighting spaces as

$$\mathcal{V} = \{w \in H^1(\Omega) \mid w = u_{\mathcal{D}} \text{ on } \partial\Omega_{\mathcal{D}}\}, \tag{2.20}$$

$$\mathcal{V}_0 = \{w \in H^1(\Omega) \mid w = 0 \text{ on } \partial\Omega_{\mathcal{D}}\}, \tag{2.21}$$

[12] **T**runcated **h**ierarchical **B**-splines.

[13] The term comes from the fact that this technology allows computational meshes to have T-junctions. Although T-splines are the de facto industrial standard in CAD when it comes to local refinement, they are a rather inelegant solution.

[14] **L**ocally **r**efined **B**-splines.

respectively. The function values on $\partial\Omega_{\mathcal{D}}$ are prescribed in the sense of traces. After multiplying (2.16) by a weighting function $w \in \mathcal{V}_0$, integrating over Ω and applying integration by parts to the left-hand side, we acquire the weak form of (2.16–2.18):

$$\text{find } u \in \mathcal{V} \text{ such that } \int_\Omega \nabla_{\mathbf{x}} u \nabla_{\mathbf{x}} w^T d\mathbf{x} = \int_\Omega f w d\mathbf{x} + \int_{\partial\Omega} g w d\mathbf{s} \quad \forall w \in \mathcal{V}_0. \tag{2.22}$$

Here, $\nabla_{\mathbf{x}}$ is the gradient operator which returns a row-vector containing all partial derivatives. The notation is generalized by defining a bilinear form $a(u,w) = \int_\Omega \nabla_{\mathbf{x}} u \nabla_{\mathbf{x}} w^T d\mathbf{x}$ and a linear form $l(w) = \int_\Omega f w d\mathbf{x} + \int_{\partial\Omega} g w d\mathbf{s}$. Then formulation (2.22) becomes

$$\text{find } u \in \mathcal{V} \text{ such that } \quad a(u,w) = l(w) \quad \forall w \in \mathcal{V}_0. \tag{2.23}$$

The bilinear form $a(u,w)$ is symmetric. Moreover, it is continuous, that is, there exists a constant $C > 0$ such that

$$a(u,w) \leqslant C||u||_{\mathcal{V}}||v||_{\mathcal{V}}, \tag{2.24}$$

and coercive, which means that there exists a constant $\alpha > 0$ such that

$$a(w,w) \geqslant \alpha||w||_{\mathcal{V}}. \tag{2.25}$$

According to the Lax-Milgram theorem, from the continuity and coercivity it follows that there exists a unique solution to the variational problem (2.26–**??**).

2.3.2 Single-patch case

The Galerkin method of finding an approximate solution to the infinite dimensional problem (2.23) involves replacing $\mathcal{V}$ and $\mathcal{V}_0$ by their finite dimensional subspaces $\mathcal{V}_h \subset \mathcal{V}$ and $\mathcal{V}_{0,h} \subset \mathcal{V}_0$ and solving the following finite dimensional variational problem:

$$\text{find } u_h \in \mathcal{V}_h \text{ such that } \quad a(u_h,w) = l(w) \quad \forall w \in \mathcal{V}_{0,h}, \tag{2.26}$$

where u_h is a numerical approximation of the exact solution u. Here, the subscript h refers to the discrete nature of the involved objects.

In IGA, $\mathcal{V}_h$ and $\mathcal{V}_{0,h}$ are constructed using NURBS. Let us assume for now[15], that Ω is a single-patch domain, which parametrization $\mathbf{G} : [0,1]^d \to \Omega$ is constructed using tensor-product NURBS basis functions $N_i : [0,1]^d \to \mathbb{R}$. Let $\mathbf{x}$ denote a point in Ω, and let us define functions N_i on Ω as $N_i(\mathbf{x}) = N_i(\mathbf{G}^{-1}(\mathbf{x}))$. A linear span of $N_i(\mathbf{x})$ defines a *physical NURBS space* $\mathcal{N}(\Omega) = \text{span}(N_i(\mathbf{x}),\ i = 1,\dots,n) \subset H^1(\Omega)$. We search for u_h as an *isogeometric solution* from $\mathcal{N}(\Omega)$, that is, as a linear combination

$$u_h(\mathbf{x}) = \sum_{i=1}^{n} y_i N_i(\mathbf{x}) \tag{2.27}$$

with unknown coefficients $y_i \in \mathbb{R}$. Each coefficient y_i corresponds to a basis function N_i, which in turn corresponds to a control point $\mathbf{c}_i$ of the mapping $\mathbf{G}$. Thus, the control net of $\mathbf{G}$ also visualizes the structure of degrees of freedom y_i.

[15] The multi-patch case is consider below.

Note that N_i are well-defined in Ω only if the parametrization $\mathbf{G}$ is locally invertible. This leads to the following condition on the Jacobian determinant of $\mathbf{G}$:

$$J(\mathbf{G}) = \det \nabla_{\boldsymbol{\xi}} \mathbf{G} > 0. \tag{2.28}$$

I refer to this as the *bijectivity constraint* and say that parametrizations satisfying this constraint are *valid*.

The space $\mathcal{N}(\Omega)$ can be considered a basic, or minimal, physical space associated with the parametrization $\mathbf{G}$. By applying h-refinement and degree elevation in various combinations, this space can be enriched. I denote the refined space by $\mathcal{N}_r^p$, where p is the achieved polynomial degree and r is the h-refinement level measured from the space $\mathcal{N}_0^p$. Following the k-refinement concept, degree elevation is applied before h-refinement.

The isogeometric solution u_h has to belong to $\mathcal{V}_h \subset \mathcal{V}$ so that it satisfies the Dirichlet boundary condition (2.17). Assume that $\partial\Omega_{\mathcal{D}}$ consists of one or more entire sides of the tensor-product NURBS patch that represents Ω[16]. In this case, we can isolate a subset of the NURBS basis functions which form $\partial\Omega_{\mathcal{D}}$. I denote the set of indices corresponding to these functions by $\mathcal{D}$. Correspondingly, only functions N_i, $i \in \mathcal{D}$ are responsible for resolving the prescribed boundary solution $u_{\mathcal{D}}$. Note that $u_{\mathcal{D}}$ may not belong to the space $\mathcal{N}(\partial\Omega_{\mathcal{D}}) = \operatorname{span}(N_i(\mathbf{x})|_{\partial\Omega_{\mathcal{D}}},\ i \in \mathcal{D})$, in which case the Dirichlet boundary condition (2.17) can only be satisfied approximately, that is,

$$u_{\mathcal{D}}(\mathbf{x}) \approx \sum_{i \in \mathcal{D}} y_i N_i(\mathbf{x})|_{\partial\Omega_{\mathcal{D}}}. \tag{2.29}$$

Assume that y_i, $i \in \mathcal{D}$ are known. Then we can rewrite (2.27) as

$$u_h(\mathbf{x}) = \sum_{i=1}^{n} y_i N_i(\mathbf{x}) = \sum_{i \in \mathcal{D}} y_i N_i(\mathbf{x}) + \sum_{i \in \mathcal{F}} y_i N_i(\mathbf{x}), \tag{2.30}$$

where $\mathcal{F}$ denotes a set of indices corresponding to free coefficients y_i, $i \in \mathcal{F}$ as opposed to the fixed coefficients y_i, $i \in \mathcal{D}$. We can now define the discrete function spaces $\mathcal{V}_{h,0}$ and $\mathcal{V}_h$ as

$$\mathcal{V}_{0,h} = \operatorname{span}(N_i, i \in \mathcal{F}) \ \text{ and } \ \mathcal{V}_h = \sum_{i \in \mathcal{D}} y_i N_i + \mathcal{V}_{h,0}. \tag{2.31}$$

Note that $\mathcal{V}_{0,h}$ contains functions from $\mathcal{N}(\Omega)$ for which $y_i = 0\ \forall i \in \mathcal{D}$, so $w = 0$ on $\partial\Omega_{\mathcal{D}}\ \forall w \in \mathcal{V}_{0,h}$. Similarly, $w \approx u_{\mathcal{D}}$ on $\partial\Omega_{\mathcal{D}}\ \forall w \in \mathcal{V}_h$. Both spaces contain functions that are C^{p-1}-smooth where p is the polynomial degree of the underlying parametrization $\mathbf{G}$[17].

Having defined $\mathcal{V}_h$ and $\mathcal{V}_{0,h}$, we can now construct a system of linear algebraic equations for the finite-dimensional variational problem (2.26). Since the problem, it suffices to test it with the basis of $\mathcal{V}_{0,h}$ to determine if it holds for every function from $\mathcal{V}_{0,h}$. This leads to the following system of equations:

$$a(u_h, N_i) = l(N_i), \quad i \in \mathcal{F}. \tag{2.32}$$

[16] This assumption is not too restrictive in practice.

[17] Of course, the smoothness of the tensor-product patch may vary depending on the parameter dimension if different polynomial degrees are chosen, or even within one dimension if different knot multiplicities are used. Thus, the smoothness of the generated physical NURBS spaces may vary as well. In the most cases, however, I use tensor-product NURBS which have the same polynomial degree p in every dimension and have multiplicity one for the inner knots. The resulting physical NURBS spaces are uniformly C^{p-1}-smooth.

After plugging in the expanded representation of u_h (2.30) and using the linearity and the symmetry of $a(u, w)$, we get

$$\sum_{j\in\mathcal{D}} y_j a(N_i, N_j) + \sum_{j\in\mathcal{F}} y_j a(N_i, N_j) = l(N_i), \qquad i \in \mathcal{F}. \tag{2.33}$$

Let us define the matrices $\mathbf{A} = \big(a(N_i, N_j)\big)_{i,j\in\mathcal{F}}$ and $\mathbf{A}_\mathcal{D} = \big(a(N_i, N_j)\big)_{i\in\mathcal{F},j\in\mathcal{D}}$ and the vectors $\mathbf{y} = \big(y_i\big)_{i\in\mathcal{F}}$, $\mathbf{y}_\mathcal{D} = \big(y_i\big)_{i\in\mathcal{D}}$ and $\mathbf{b} = \big(l(N_i)\big)_{i\in\mathcal{F}}$. Then the linear system (2.33) can be rewritten in a matrix form as

$$\mathbf{A}\mathbf{y} = \mathbf{b} - \mathbf{A}_\mathcal{D}\mathbf{y}_\mathcal{D}. \tag{2.34}$$

$\mathbf{A}$ is called the *stiffness matrix*, and $\mathbf{A}_\mathcal{D}$ is called the *elimination matrix* (because it eliminates Dirichlet degrees of freedom y_i, $i \in \mathcal{D}$ from the linear system). After solving matrix equation (2.34) for $\mathbf{y}$, the isogeometric solution u_h can be constructed. Since the stiffness matrix $\mathbf{A}$ inherits properties of the bilinear form $a(u, v)$, it is symmetric positive definite. Therefore, the solution vector $\mathbf{y}$ is well-defined, and so is the isogeometric solution u_h.

Dirichlet degrees of freedom

In this work, I use the elimination strategy to enforce Dirichlet boundary conditions strongly. To that end, the coefficients y_i, $i \in \mathcal{D}$, responsible for resolving the prescribed boundary solution $u_\mathcal{D}$, are determined before the rest of the coefficients y_i, $i \in \mathcal{F}$ and are treated as fixed. I compute y_i, $i \in \mathcal{D}$ by projecting $u_\mathcal{D}$ in L^2-sense onto $\mathcal{N}(\partial\Omega_\mathcal{D})$. This is achieved by solving an auxiliary matrix equation $\mathbf{M}_\mathcal{D}\mathbf{y}_\mathcal{D} = \mathbf{u}_\mathcal{D}$. Here, $\mathbf{M}_\mathcal{D}$ is a *projection mass matrix* and $\mathbf{u}_\mathcal{D}$ is a right-hand side vector. Their entries are defined as

$$(\mathbf{M}_\mathcal{D})_{ij} = \int_{\partial\Omega_\mathcal{D}} N_i N_j d\mathbf{s}, \quad i, j \in \mathcal{D} \text{ and } (\mathbf{u}_\mathcal{D})_i = \int_{\partial\Omega_\mathcal{D}} N_i u_\mathcal{D} d\mathbf{s}, \quad i \in \mathcal{D}. \tag{2.35}$$

A-priori error estimate

I recite the main result regarding the accuracy of the isogeometric solution u_h as given in [2]. Recall that the norm associated with a Sobolev space $H^r(\Omega)$ is given by

$$||u||_r^2 = \sum_{|\boldsymbol{\alpha}|\leqslant r} \big(\mathcal{D}^{\boldsymbol{\alpha}}, \mathcal{D}^{\boldsymbol{\alpha}} u\big)_\Omega. \tag{2.36}$$

Then the fundamental error estimate for elliptic problems expressed as a bound on the difference between the exact solution u and the isogeometric solution u_h takes the form

$$||u - u_h||_m \leqslant C h^\beta ||u||_r. \tag{2.37}$$

Here, h is a characteristic length scale related to the element size in the isogeometric mesh, $\beta = \min(p + 1 - m, r - m)$ where p is the polynomial order of the mapping $\mathbf{G}$, and C is a constant that does not depend on u or h. Of particular interest here is the term h^β. The order of convergence β defines the behavior of the error as we refine the mesh. For example, if we refine the mesh uniformly and reduce h to $h/2$, we can expect the error to reduce by a factor $1/2^\beta$.

Linear system formation

It is convenient to compute the entries to the stiffness matrix $\mathbf{A}$ by transforming the integrals from the spatial coordinates $\mathbf{x}$ to the parametric coordinates $\boldsymbol{\xi}$. Concretely,

$$(\mathbf{A})_{ij} = a(N_i, N_j) = \int_{\Omega} \nabla_{\mathbf{x}} N_i \nabla_{\mathbf{x}} N_j^T d\mathbf{x} = \int_{[0,1]^d} \nabla_{\boldsymbol{\xi}} N_i \nabla_{\boldsymbol{\xi}} \mathbf{G}^{-1} \big(\nabla_{\boldsymbol{\xi}} N_j \nabla_{\boldsymbol{\xi}} \mathbf{G}^{-1}\big)^T J(\mathbf{G}) d\boldsymbol{\xi}. \quad (2.38)$$

Here, we again rely on the fact that $\mathbf{G}$ is bijective in order to transform the basis function gradients with respect to $\boldsymbol{\xi}$ to the gradients with respect to $\mathbf{x}$. Note that the gradients with respect to the parametric coordinates $\boldsymbol{\xi}$ can be efficiently computed using specialized NURBS algorithms, described for example in [39]. On $[0,1]^d$, integration can be performed element-wise using the standard Gaussian quadrature. The same assembly method is applied to the entries of the loading vector $\mathbf{b}$.

Although the higher smoothness of NURBS function spaces is good in some situations, it is a disadvantage when it comes to linear system formation. This is because each NURBS basis function of degree p has a support spanning $(p+1)^d$ elements, which is a lot in comparison to 2^d elements in the case of FEM. Because of that, an element-wise assembly procedure can be extremely computationally expensive, especially for high polynomial degrees p. Concretely, a local stiffness matrix on each element has dimension $(p+1)^{2d}$, and each of its entries is computed using $(p+1)^d$ Gauss points (if standard Gaussian quadrature is used). This results in the computational cost of assembling the global stiffness matrix $\mathbf{A}$ being $O(N(p+1)^{3d})$ FLOPs, where N is a total number of degrees of freedom. In order to reduce the required computational effort, alternative assembly paradigms are necessary, like the row-wise assembly with the specially designed quadratures [56]. Another solution is to replace the Galerkin version of IGA with the collocation methods which are viable in IGA due to the high smoothness of the NURBS function spaces [57]. In this work, I do not use NURBS spaces of degree larger than three, so the computational cost of matrix assembly remains manageable.

2.3.3 Multi-patch case

Let Ω be a two-patch domain consisting of two subdomains Ω_1 and Ω_2, each being an image of its own tensor-product NURBS mapping $\mathbf{G}_k : [0,1]^d \to \Omega_k$, $k = 1, 2$. On each patch, a physical NURBS space $\mathcal{N}(\Omega_k)$ is defined in the same way as for the single-patch case. The isogeometric solution u_h is constructed patch-wise as

$$u_h(\mathbf{x}) = \begin{cases} u_h^1(\mathbf{x}) = \sum_{i=1}^{n_1} y_i^1 N_i^1(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega_1, \\ u_h^2(\mathbf{x}) = \sum_{i=1}^{n_2} y_i^2 N_i^2(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega_2, \end{cases} \quad (2.39)$$

where $N_i^k \in \mathcal{N}(\Omega_k)$ and $y_i^k \in \mathbb{R}$. Usually, the solution is required to be at least continuous at the interface Γ between the patches:

$$u_h^1 = u_h^2 \text{ on } \Gamma. \quad (2.40)$$

The continuity condition (2.40) is easy to achieve with matching interfaces. To that end, it suffices to identify the degrees of freedom $\mathbf{y}_\Gamma^k$ that correspond to the interface basis functions on both patches and enforce $\mathbf{y}_\Gamma^1 = \mathbf{y}_\Gamma^2$. This is equivalent to "gluing" together the interface basis functions from both sides of the interface, forming new inter-patch C^0-smooth basis functions as illustrated in Figure 2.12. When the linear system is assembled, matrix entries corresponding to the glued interface basis functions are summed. The approach is directly extendable for an arbitrary number of patches.

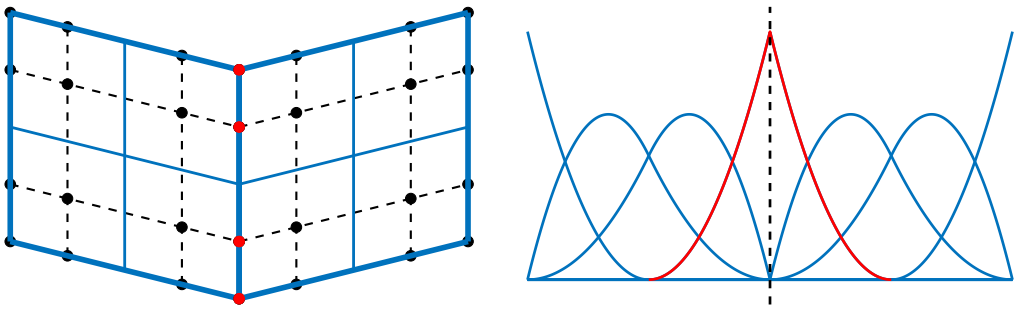

Figure 2.12. Left: two-patch domain with a matching interface. Right: NURBS bases in the cross-interface parametric direction of the two patches. Interface degrees of freedom and C^0-smooth inter-patch basis function are marked red.

It is also possible to connect the patches in a C^1-smooth fashion [58, 59]. In that case, the solution patches should satisfy the conditions

$$u_h^1 = u_h^2 \text{ on } \Gamma, \tag{2.41}$$

$$\nabla_{\mathbf{x}} u_h^1 \cdot \mathbf{n} = \nabla_{\mathbf{x}} u_h^2 \cdot \mathbf{n} \text{ on } \Gamma, \tag{2.42}$$

which result in more constraints on the interface degrees of freedom y_i^k. These constraints, however, cannot be satisfied by a similar gluing trick and involve computation of a global C^1-smooth subspace of $\mathcal{N}(\Omega_1) \cup \mathcal{N}(\Omega_2)$.

2.3.4 Numerical example: Poisson's equation on a unit disk

This section presents a numerical example of solving Poisson's equation on a disk with particular emphasis placed on the effect that the domain parametrization has on the solution accuracy. The simplicity of the chosen geometry allows a number of different parametrizations to be constructed and compared. All simulations have been performed using the open-source C++ library G+Smo [60], which implements most of geometric routines needed in IGA. A parallel implementation based on OpenMP has been used for the linear system assembly. The assembled linear systems have been solved using Pardiso [61], an efficient parallel direct linear solver.

The computational domain Ω is a radius one disk centered at the origin. Figure 2.13 shows four quadratic NURBS parametrizations of the domain Ω used for comparison. The first one is a single-patch (SP) parametrization with four degenerated corner points. Another single-patch parametrization belongs to a class of the so-called scaled boundary (SB) parametrizations [62]. Being a generalization of polar coordinates, this class of parametrizations is very suitable for domains with rotational symmetries. The SB parametrization is constructed by mapping the south side of the parametric domain to the disk center. The north side is mapped onto the boundary curve built of four 90° NURBS circular arcs. The west and the east are mapped onto one of the rays connecting the boundary curve to the disk center. Since the boundary is constructed as four curves connected in a C^0 fashion, there are four rays between the connection points and the disk center along which the parametrization is only C^0-smooth. All degrees of freedom corresponding to the disk center are unified resulting in a special central basis function and eliminating the discontinuity.

Two other parametrizations are multi-patch. The first, labeled MP1, consists of five patches: four for each patch of the boundary curve and one central hub-patch. The second multi-patch parametrization, labeled MP2, is acquired by splitting each patch of the MP1 parametrization into four, resulting in 20 patches total. The idea is to check if such an excessive number of patches has any influence on the solution accuracy.

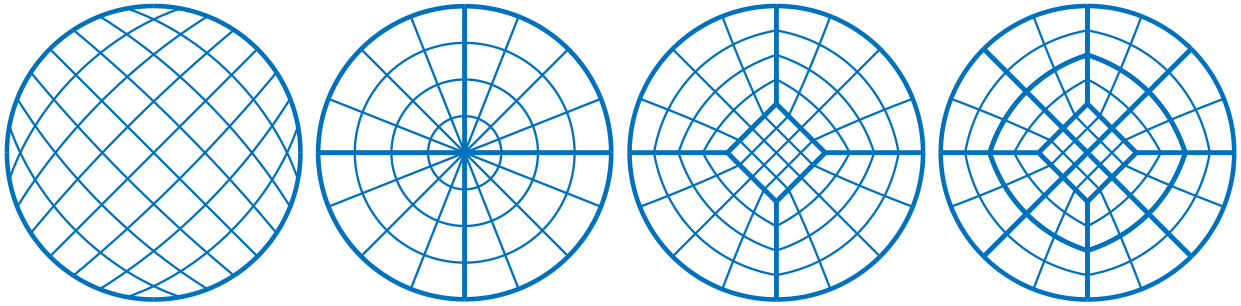

Figure 2.13. Parametrizations of the disk used in the example. From left to right: single-patch (SP), scaled boundary (SB), multi-patch with 5 patches (MP1), multi-patch with 20 patches (MP2). Every parametrization has been uniformly refined several times in order to better depict its structure.

As a reference solution for error analysis, a *manufactured solution*

$$u^* = \cos(6\pi(x^2 + y^2)) \tag{2.43}$$

is used (see Figure 2.14), which satisfies the following system:

$$-\Delta_{\mathbf{x}} u = -\Delta_{\mathbf{x}} u^* \text{ in } \Omega, \tag{2.44}$$

$$u = u^* \text{ on } \partial\Omega_{\mathcal{D}}, \tag{2.45}$$

$$\nabla_{\mathbf{x}} u \cdot \mathbf{n} = 0 \text{ on } \partial\Omega_{\mathcal{N}}. \tag{2.46}$$

Here, the Dirichlet part of the boundary $\partial\Omega_{\mathcal{D}}$ is one of the four NURBS circular arcs. Due to the symmetry of the manufactured solution u^*, any arc can be chosen. The rest of the boundary is the Neumann part $\partial\Omega_{\mathcal{N}}$. Such an unusual choice of boundary conditions should make the comparison of different parametrizations as fair as possible. If the Dirichlet boundary condition was set on the entire boundary, the corresponding boundary degrees of freedom would be excluded from the simulation due to the elimination strategy. Any parametrization with ill-shaped or degenerated elements on the boundary (for example, the SP parametrization) would benefit from it. On the contrary, the chosen setting is designed to be challenging for every parametrization.

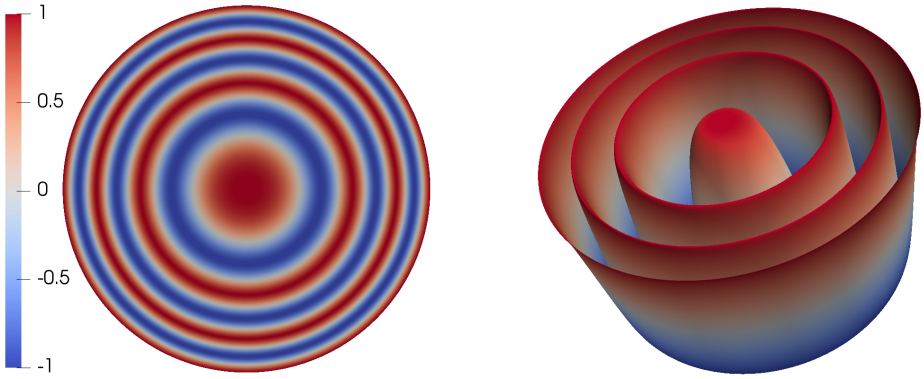

Figure 2.14. Manufactured solution u^*.

The quantities of interest are the L^2- and H^1-norms of the error $e_h = u_h - u^*$, as well as the condition number of the stiffness matrix

$$\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}. \tag{2.47}$$

Particular focus is placed on the behavior of these quantities for different parametrizations as they are uniformly h-refined. Since quadratic NURBS are used, the error estimate (2.37) predicts that $||e_h||_{L^2} = O(h^3) = O(N^{-3/2})$ and $||e_h||_{H^1} = O(h^2) = O(N^{-1})$. For the condition number, we can expect that $\kappa(A) = O(h^{-2}) = O(N)$ since Poisson's equation is a second order elliptic PDE [63].

The simulation results are presented in Figure 2.15. The observed convergence rates for the L^2- and H^1-norms coincide with the predictions. There is hardly any difference between the parametrizations in terms of the L^2-norm. With respect to the H^1-norm, the MP2 parametrization has a marginally larger error which can be explained by an excessive number of patches. Apart from that, the domain parametrization has a negligible effect on the accuracy of the solution in the presented example.

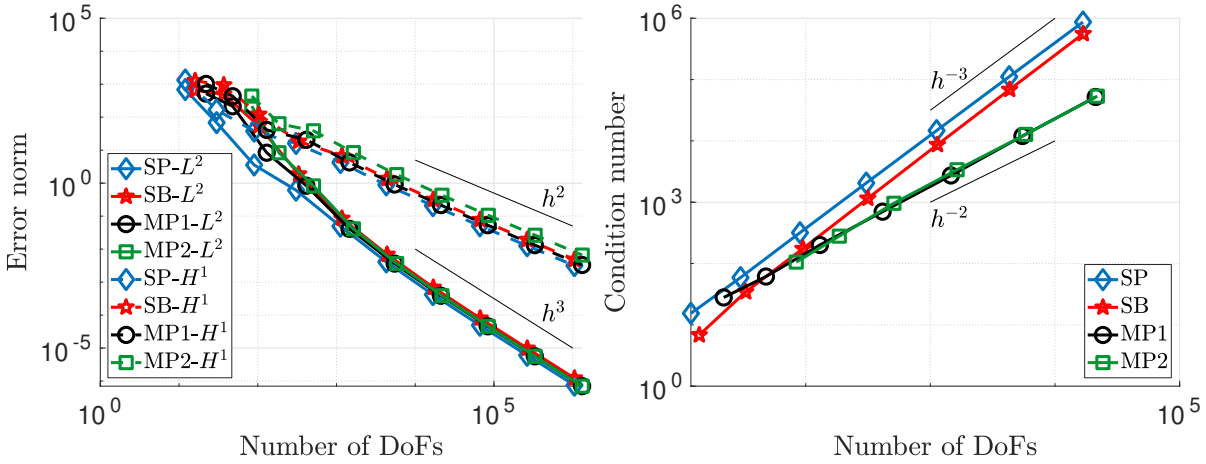

Figure 2.15. Left: L^2-norm and H^1-norm of the error for different parametrizations. Right: condition number of the stiffness matrix for different parametrizations.

The situation is different, however, when it comes to the condition number. For both multi-patch parametrizations, the observed growth rate of the condition number coincides with the prediction. On the other hand, the condition number grows roughly as h^{-3} for the the SP and SB parametrizations. The most likely reason is the degenerated elements that these parametrizations possess. For a sufficiently large number of degree of freedom N, both MP parametrizations have the lowest condition number, whereas the SP parametrization has the highest.

The results of this numerical example suggest the following conclusion. Although a domain parametrization may not have a significant effect on accuracy of the numerical simulations, it can affect the condition number of the corresponding stiffness matrix. The condition number, in turn, has an effect on the solution process. In particular, high condition numbers make iterative linear solvers inefficient unless a suitable preconditioner is used. On the other hand, the condition number does not affect the solution time if a direct linear solver is used. However, high values of condition number reduce the solution accuracy in the floating-point arithmetic. In light of that, multi-patch parametrizations with well-shaped patches should be preferred for complex domains over single-patch parametrizations.

2.4 Summary

In this chapter, I have given a brief introduction into IGA and a global overview of the-state-of-art in the area of analysis-suitable geometric modeling. In particular, I have focused on the multi-patch

tensor-product NURBS approach and discussed its most prominent advantages and disadvantages. I have also described the isogeometric Galerkin method and illustrated it on a numerical example for Poisson's equation. As a conclusion, I recapitulate the main properties of IGA. While doing it, I also juxtapose IGA with FEM in order to put each property into perspective.

- The IGA approach to describe geometry with NURBS allows to exactly represent many classes of geometrical objects, especially those coming from CAD. Piecewise linear approximations traditionally used in FEM provide a much lower degree of geometric accuracy. The higher geometric accuracy of IGA often results in better accuracy of numerical solutions solution of PDEs.
- In IGA, an exact solution of a given PDE is approximated by an element-wise polynomial function of degree p which is C^{p-1}-smooth between the elements. In FEM, the approximation is only is C^0-smooth between the elements. The higher smoothness of the numerical approximation yields more accuracy per degree of freedom and is especially beneficial in the eigenvalue problems [4].
- The higher smoothness of the NURBS function spaces makes the element-wise assembly of the stiffness matrix prohibitively expensive for high polynomial degrees. As a consequence, the traditional FEM software structure can not be efficiently reused in IGA. When developing novel IGA software, special emphasis has to be placed on alternative assembly strategies.
- The history of FEM software development is nearly fifty-year-long. In comparison, the whole concept of IGA is only fifteen years old, and IGA is still an active research field. This means that there is still a lack of robust software. Moreover, many operations that may be considered straightforward in FEM are time-consuming and require manual work.

3 | Single-patch domain parametrization

One of the most commonly encountered problems in IGA reads as follows: given a parametrization of the domain's boundary, how can one construct a consistent parametrization of its interior? This chapter deals with a core building block of this problem - construction of single-patch domain parametrizations. After defining the problem setting, I describe the start-of-the-art in the field and study a number of most used solutions in details. For the sake of simplicity, I restrict the discussion to a case of planar domains, however, all of the methods can be straightforwardly generalized to a volumetric case. I then propose an alternative solution approach to this problem based on the stationary equations of nonlinear elasticity which proves to be especially efficient for domains with complex geometry. Finally, I compare performance of the proposed approach against the existing approaches on a range of two-dimensional and three-dimensional examples.

3.1 Problem setting

Let $\Omega \subset \mathbb{R}^2$ be a simply connected domain which boundary $\partial\Omega$ is given by a union of four NURBS curves[1] $\boldsymbol{\gamma}_S(\xi)$, $\boldsymbol{\gamma}_N(\xi)$, $\boldsymbol{\gamma}_W(\eta)$ and $\boldsymbol{\gamma}_E(\eta) : [0,1] \to \mathbb{R}^2$, such that

$$\boldsymbol{\gamma}_S(0) = \boldsymbol{\gamma}_W(0), \quad \boldsymbol{\gamma}_S(1) = \boldsymbol{\gamma}_E(0), \quad \boldsymbol{\gamma}_N(0) = \boldsymbol{\gamma}_W(1), \quad \boldsymbol{\gamma}_N(1) = \boldsymbol{\gamma}_E(1). \tag{3.1}$$

Moreover, let the oppositely lying curves, namely $\boldsymbol{\gamma}_S(\xi)$ and $\boldsymbol{\gamma}_N(\xi)$ as well as $\boldsymbol{\gamma}_W(\eta)$ and $\boldsymbol{\gamma}_E(\eta)$, share the same NURBS bases. Under these assumptions, the four boundary curves imply a tensor-product NURBS parametrization[2] $\mathbf{G}(\xi,\eta) : [0,1]^2 \to \Omega$,

$$\mathbf{G}(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{c}_{ij} N_i(\xi) M_j(\eta) = \sum_{i=1}^{n} \mathbf{c}_i N_i(\boldsymbol{\xi}) = \sum_{i\in B} \mathbf{c}_i N_i(\boldsymbol{\xi}) + \sum_{i\in I} \mathbf{c}_i N_i(\boldsymbol{\xi}), \tag{3.2}$$

such that

$$\mathbf{G}(\xi,0) = \boldsymbol{\gamma}_S(\xi), \quad \mathbf{G}(\xi,1) = \boldsymbol{\gamma}_N(\xi), \quad \mathbf{G}(0,\eta) = \boldsymbol{\gamma}_W(\eta), \quad \mathbf{G}(1,\eta) = \boldsymbol{\gamma}_E(\eta). \tag{3.3}$$

The structure of $\mathbf{G}$ is largely dictated by the parametrization of the boundary curves. Concretely, the NURBS bases N_i, $i = 1,\dots,n$ and M_j, $j = 1,\dots,m$ are defined by the pairs of oppositely lying curves. The boundary control points $\mathbf{c}_i$, $i \in B$ are the control points of the boundary curves, namely the four sets $\{\mathbf{c}_{i1}\}_{i=1}^{n}$, $\{\mathbf{c}_{im}\}_{i=1}^{n}$, $\{\mathbf{c}_{1j}\}_{j=1}^{m}$ and $\{\mathbf{c}_{nj}\}_{j=i}^{m}$ come from $\boldsymbol{\gamma}_S$, $\boldsymbol{\gamma}_N$, $\boldsymbol{\gamma}_W$ and $\boldsymbol{\gamma}_E$, respectively. What is left to define are the interior control points $\mathbf{c}_i$, $i \in I$, and this is the essence of the single-patch domain parametrization problem in IGA, see Fig 3.1.

[1] First letters of the words **W**est, **E**ast, **S**outh and **N**orth are used as the indices.

[2] I remind the reader of the alternative notation for tensor-product NURBS parametrizations introduced in (2.13) and (2.14).

Additionally, in order to be suitable for IGA, $\mathbf{G}$ has to be bijective. This means that the following condition on its Jacobian determinant must hold:

$$J(\mathbf{G}) = \det \nabla_{\boldsymbol{\xi}} \mathbf{G} > 0. \tag{3.4}$$

Enforcing the bijectivity constraint (3.4) represents the main source of complexity of the single-patch domain parametrization problem.

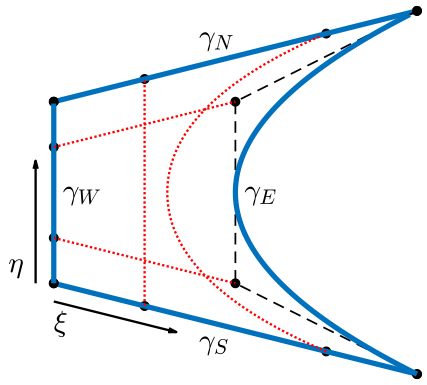

Figure 3.1. Single-patch domain parametrization problem: given four boundary NURBS curves (blue), construct a tensor-product NURBS parametrization of the domain. The known boundary control points (black) imply a grid-like structure for the unknown inner control points (red) which location has to be defined.

I would like to conclude the definition of the problem by noticing that the assumption of shared basis for the oppositely lying boundary curves is not too restrictive in practice. It can be easily satisfied by applying knot insertions and degree elevation. In the worst case, both curves can be reparametrized by fitting the given curves using new curves with the same NURBS basis. Although the reparametrization often preserves the shape of the boundary curves only up to a certain tolerance, it is justified in many applications. Moreover, the reparametrization of the boundary curves is often advisable in order to achieve relative uniformity of the parametrization, that is,

$$||\boldsymbol{\gamma}'(\xi)|| \approx \text{const}\,. \tag{3.5}$$

Property (3.5) later translates into uniformly sized elements of the resulting domain parametrization.

3.2 Overview of existing methods

Due to ubiquity of the single-patch domain parametrization problem, there exists a plethora of solution approaches which vary widely in their complexity and applicability. They can nevertheless be split into two general categories: linear and nonlinear. The former comprises methods which involve various types of interpolation or require solution of one linear system at most. Due to their simplicity, these methods are not able to take constraint (3.4) into account. The linear methods are computationally inexpensive but produce good-quality parametrizations only for rather simple domains. The latter category includes methods which involve solving nonlinear optimization problems or PDEs. These methods are more complex and can handle domains with nontrivial geometry. The nonlinear methods enforce constraint (3.4), either explicitly or implicitly. Despite their seeming superiority, the nonlinear methods often rely on the linear methods to produce a starting point for an iterative solution procedure.

In the following, I describe the most commonly used linear and nonlinear methods. Although there exist many variations of linear methods, I present only two of them which I have used extensively throughout this work: the spring model [25] and the Coons patch [27]. After that, I describe nonlinear methods: the optimization-based methods with various mesh quality measures [25, 28] and elliptic grid generation which is based on solution of the inverse Laplace equation [29].

3.2.1 Spring model

The *spring model* mimics a mechanical system obtained by replacing each edge of the NURBS control net with a linear elastic spring. The corresponding equilibrium state defines the location of the interior control points:

$$\mathbf{c}_{ij} = \frac{1}{4}(\mathbf{c}_{i-1j} + \mathbf{c}_{i+1j} + \mathbf{c}_{ij-1} + \mathbf{c}_{ij+1}), \quad i = 2, \ldots, n-1, \quad j = 2, \ldots, m-1. \tag{3.6}$$

. An alternative and sometimes neater way to write formula (3.6) is using a *mask*[3]:

$$\mathbf{c}_{ij} = \frac{1}{4} \begin{pmatrix} 0 & 1 & 0 \\ 1 & \cdot & 1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{3.7}$$

In this form, it is directly visible that the spring model is very similar to a discrete version of the Laplace PDE. Formulas (3.6) and (3.7) define a linear system which has to be solved in order to find the interior control points.

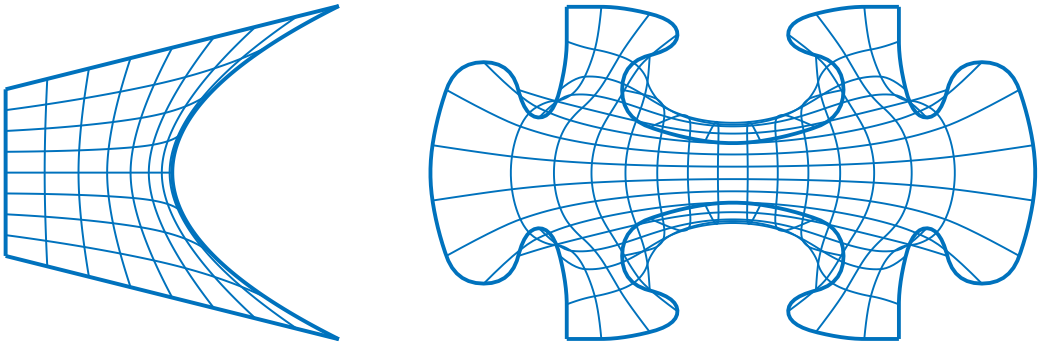

Figure 3.2. Spring model applied to a simple domain (left) and to a complex domain (right). The parametrization of the complex domain has folds - a typical sign that the bijectivity constraint (3.4) is violated.

As all linear methods, the spring model does not enforce the bijectivity constraint (3.4). So in general, one can expect that it produces bijective parametrizations only for rather simple domains, like the one in Figure 3.1. Still, as Figure 3.2 shows, the resulting parametrization is *barely* bijective since $\min J(\mathbf{G})$ is very close to zero. This is indicated by a congestion of the isoparametric lines next to the eastern side. When applied to a more complex domain, the spring model produces parametrizations with *folds* - characteristic regions where the mapping $\mathbf{G}$ is not unique invertible, see Figure 3.2.

Although it is hard to formalize a notion of simple or complex domains, I can provide the following rule of thumb: if the domain's boundary contains prominent concave regions, the domain

[3] A mask is also known as a *stencil* in the FDM lingo.

is considered complex, and the linear parametrization methods are likely to produce an invalid parametrization. Note that the parametrizations from Figure 3.2 have problems exactly next to the concave parts of the boundary.

3.2.2 Coons patch

Given the boundary curves $\boldsymbol{\gamma}_S$, $\boldsymbol{\gamma}_N$, $\boldsymbol{\gamma}_W$ and $\boldsymbol{\gamma}_E$, the bilinearly blended *Coons patch* defines the domain parametrization $\mathbf{G} : [0,1]^2 \to \Omega$ as

$$\begin{aligned}\mathbf{G}(\xi,\eta) &= (1-\xi)\boldsymbol{\gamma}_W(\eta) + \xi\boldsymbol{\gamma}_E(\eta) \\ &\quad + (1-\eta)\boldsymbol{\gamma}_S(\xi) + \eta\boldsymbol{\gamma}_N(\xi) \\ &\quad - (1-\xi \;\; \xi)\begin{pmatrix} \boldsymbol{\gamma}_S(0) & \boldsymbol{\gamma}_N(0) \\ \boldsymbol{\gamma}_S(1) & \boldsymbol{\gamma}_N(1) \end{pmatrix}\begin{pmatrix} 1-\eta \\ \eta \end{pmatrix}.\end{aligned} \tag{3.8}$$

This operation is also know as the *transfinite interpolation*, first introduced in [64]. I refer to formula (3.8) as the continuous Coons patch which operates directly on the curves[4]. Since in this work I assume that $\boldsymbol{\gamma}_S$, $\boldsymbol{\gamma}_N$, $\boldsymbol{\gamma}_W$ and $\boldsymbol{\gamma}_E$ are NURBS curves, it would be convenient to define the Coons patch as an operation acting on the control points $\mathbf{c}_{ij}$. For $i = 1,\ldots,n$ and $j = 1,\ldots,m$, the discrete version of (3.8) defines $\mathbf{c}_{ij}$ as

$$\begin{aligned}\mathbf{c}_{ij} &= (1-(i-1)/(n-1))\mathbf{c}_{1j} + (i-1)/(n-1)\mathbf{c}_{nj} \\ &\quad + (1-(j-1)/(m-1))\mathbf{c}_{i1} + (j-1)/(m-1)\mathbf{c}_{im} \\ &\quad - (1-(i-1)/(n-1) \quad (i-1)/(n-1))\begin{pmatrix} \mathbf{c}_{11} & \mathbf{c}_{1m} \\ \mathbf{c}_{n1} & \mathbf{c}_{nm} \end{pmatrix}\begin{pmatrix} 1-(j-1)/(m-1) \\ (j-1)/(m-1) \end{pmatrix}.\end{aligned} \tag{3.9}$$

I refer to formula (3.9) as the discrete Coons patch. For NURBS, formulas (3.8) and (3.9) are equivalent [27].

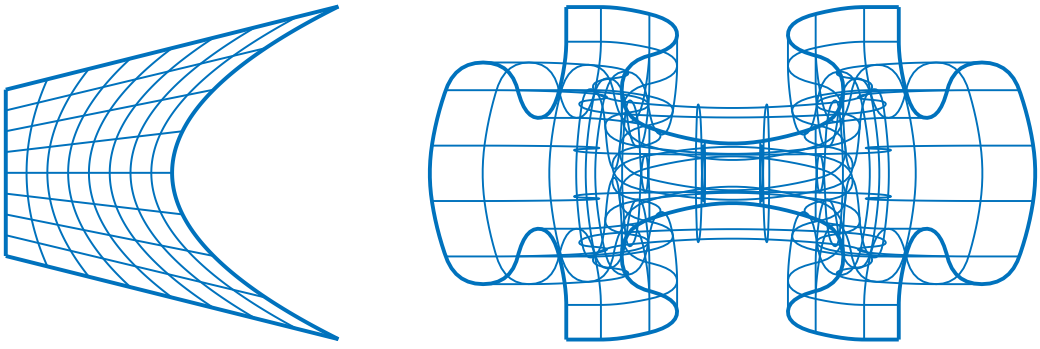

Figure 3.3. Coons patch applied to a simple domain (left) and to a complex domain (right). The method produces far better parametrization for the simple domain than the spring model. For the complex domain, the output is still unusable.

Just like the spring model, the Coons patch does not enforce the bijectivity constraint (3.4). Nevertheless, it provides a better parametrization for the simple domain from Figure 3.1 than the spring model. For the complex puzzle piece domain, it still produces a parametrization with folds, see Figure 3.3.

[4]The continuous Coons patch works for any parametric curves, not only NURBS.

Although formula (3.9) allows to directly compute the interior control points, it is instructive to observe that, according to [27], the Coons patch can be rewritten using a mask as

$$\mathbf{c}_{ij} = \frac{1}{4}\begin{pmatrix} -1 & 2 & -1 \\ 2 & \cdot & 2 \\ -1 & 2 & -1 \end{pmatrix}. \tag{3.10}$$

Of course, solving the corresponding linear system is a far more computationally expensive way to compute the interior control points than the explicit formula (3.9). Still, it offers a valuable insight that the Coons patch and the spring model are very similar methods. In fact, both are just examples of a more general method

$$\mathbf{c}_{ij} = \begin{pmatrix} \alpha & \beta & \alpha \\ \beta & \cdot & \beta \\ \alpha & \beta & \alpha \end{pmatrix}. \tag{3.11}$$

Here, a condition $4\alpha + 4\beta = 1$ has to hold so that the mask (3.11) defines an affine combination of the surrounding control points. Other linear solution methods for the parametrization problem can be constructed by varying the parameters α and β. They would all, however, require a solution of a linear system.

3.2.3 Optimization-based methods

Methods based on nonlinear optimization seek to find a domain parametrization $\mathbf{G}$ which is optimal with respect to some *quality measure*. Most used are the *geometric quality measures* which have a form

$$q(\mathbf{G}) = \int\limits_{[0,1]^2} m(\mathbf{G}) d\xi d\eta, \tag{3.12}$$

where $m(\mathbf{G})$ is a *local quality measure* depending only on $\mathbf{G}$. By contrast, the so-called *analysis-aware quality measures* [25] may also depend on a numerical solution u_h to a particular PDE that is being solved using $\mathbf{G}$. With the analysis-aware quality measures, the idea is to construct a perfect parametrization for a given problem. In this overview, however, I only consider geometric quality measures since they produce generally optimal parametrizations which should perform equally well for any encountered PDE. Moreover, all other considered categories of parametrization approaches are "analysis-unaware", which makes the comparison fair.

Interior control points $\mathbf{c}_i$ are found by solving the following optimization problem:

$$\min\ q(\mathbf{G}) \text{ subject to} \tag{3.13}$$

$$J(\mathbf{G}) \geqslant C \text{ in } [0,1]^2, \tag{3.14}$$

$$\mathbf{G} = \partial\mathbf{G} \text{ on } \partial[0,1]^2. \tag{3.15}$$

Here, $\partial\mathbf{G}$ is the parametrization of the domain boundary defined by the boundary curves $\boldsymbol{\gamma}_S$, $\boldsymbol{\gamma}_N$, $\boldsymbol{\gamma}_W$ and $\boldsymbol{\gamma}_E$ and condition (3.3). The bijectivity constraint (3.4) is explicitly enforced since most of the quality measures do not guarantee it. The constant C should ideally be larger that 0 in order to make sure that $\mathbf{G}$ is bijective. However, a concrete value of C is problem-dependent and hard to choose. In practice, C is often set to 0 which results in barely bijective parametrization. Optimization problem (3.13–3.15) is a general nonlinear non-convex problem. It is usually solved

using a black-box optimizer, for example, by providing the gradient and Hessian information to an implementation of Newton's method [65].

The most commonly used local quality measures are based on entries of the *first fundamental form* $\mathbf{g}(\mathbf{G})$, which is defined as

$$\mathbf{g}(\mathbf{G}) = \begin{pmatrix} g_{11} & g_{12} \\ g_{12} & g_{22} \end{pmatrix} = \nabla_{\boldsymbol{\xi}} \mathbf{G}^T \nabla_{\boldsymbol{\xi}} \mathbf{G}. \tag{3.16}$$

Here, I consider only three quality measures: area-orthogonality, Liao and Winslow. This list is, of course, not extensive. Additionally, a linear combination of any quality measures defines a new one. However, I would like to keep this discussion concise since the optimization-based methods for domain parametrization are not the main scope of this work.

The *area-orthogonality* (AO) local quality measure is defined as

$$m_{AO}(\mathbf{G}) = g_{11} g_{22}. \tag{3.17}$$

In my opinion, the relation between the name and the definition is not obvious. Since it is entry g_{12} that is responsible for the local skewness of the mapping, it would make more sense to minimize $m_{AO}(\mathbf{G}) = g_{12}^2$ in order to construct a parametrization with orthogonal mesh lines (which I refer to as *locally orthogonal*). However, an optimization problem (3.13–3.15) with such a quality measure does not converge. By contrast, local quality measure (3.17) results not only in a converging optimization problem but also in a locally orthogonal parametrization, see Figure 3.4. In my experience, the AO quality measure provides bijective parametrizations even without the bijectivity constraint (3.14), however, no proof is available.

The *Liao* quality measure, defined as the Frobenius norm of $\mathbf{g}(\mathbf{G})$, measures overall complexity of the parametrization:

$$m_L(\mathbf{G}) = g_{11}^2 + g_{22}^2 + 2g_{12}^2. \tag{3.18}$$

Its minimization results in simple, or "flat", parametrizations which are often prevented from becoming invalid only by constraint (3.14). If the constant C in the constraint is too low or zero, the resulting parametrization is barely bijective with characteristic regions of congested mesh lines, see Figure 3.4. On the other hand, a too high C could prevent the optimization problem from converging. In either case, I find the Liao quality measure rather unpractical.

Finally, the *Winslow* quality measure is defined as

$$m_W(\mathbf{G}) = \frac{\operatorname{tr}(\mathbf{g})}{J(\mathbf{G})} = \frac{(\sqrt{\lambda_1} - \sqrt{\lambda_2})^2}{\sqrt{\lambda_1 \lambda_2}} + 2, \tag{3.19}$$

where λ_1 and λ_2 are the eigenvalues of $\mathbf{g}$. $m_W(\mathbf{G})$ is minimal when $\lambda_1 = \lambda_2$, which means that $\mathbf{G}$ is a *conformal mapping*, that is, a composition of scaling and rotation. A unique property of the Winslow quality measure is that its unique minimizer is a bijective mapping [25]. However, the isogeometric solution acquired by solving optimization problem (3.13–3.15) using NURBS may not possess the same property if the number of degrees of freedom is too low. Figure 3.4 shows the result of minimizing the Winslow quality measure. An apparent property of the resulting parametrization is that the mesh lines tend to spread away from each other next to protruding regions of the domain boundary, and congest together next to concave regions.

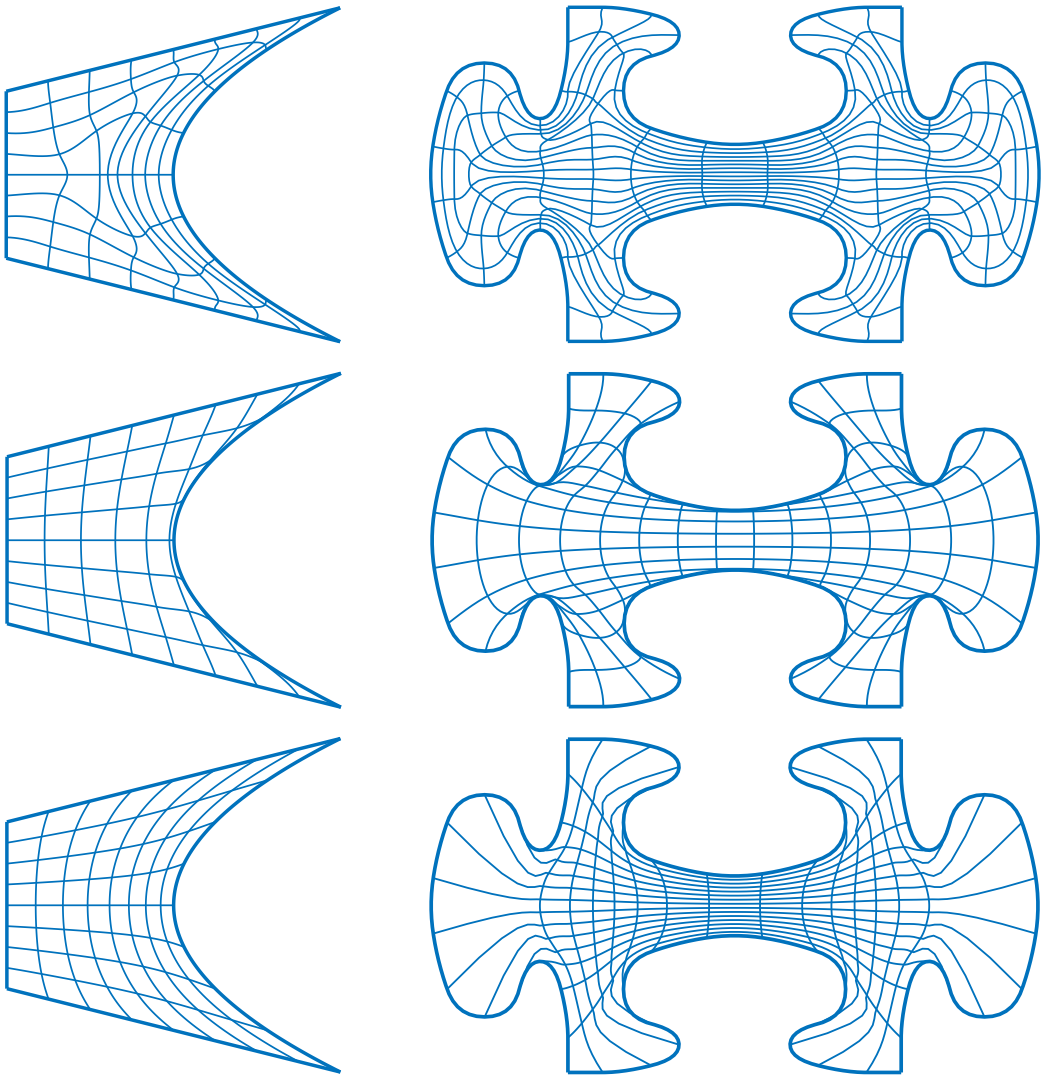

Figure 3.4. Results of applying the optimization-based methods: area-orthogonality (top), Liao (middle) and Winslow (bottom). The Liao measure yields barely bijective parametrizations. The AO and Winslow measures grant bijective parametrizations.

Disadvantages of the optimization-based approach

Overall, the optimization-based approach to the single-patch domain parametrization problem is extremely powerful due to a variety of quality measures to choose from, and because it leverages such a well-developed field as nonlinear constrained optimization. It is also one of the most widely adopted parametrization methods. Having said that, I would like to point out two significant drawbacks of this approach. First, an initial guess for the optimization loop is required. If the initial guess is chosen poorly, the optimization problem may converge only to a local minimum, or show a very low convergence rate, or even not converge at all. Moreover, the Winslow quality measure (3.19) requires a bijective initial guess—constructing which is a nontrivial task on its own.

Secondly, at least in my experience, the optimization-based methods tend to be computationally expensive. Even for simple domains, the optimization loop takes about 20–40 iterations to converge with appropriately chosen values for the absolute and relative tolerances. For complex domains,

the number of iterations can become even larger. Of course, the number of iterations depends on a particular implementation of the nonlinear optimization solver. However, the two implementations that I have tested, namely a Python implementation based on the state-of-the-art nonlinear optimization library IPOPT [66] and a Matlab implementation using the *fmincon* function, require a very similar number of iterations to converge. The reason behind such slow convergence may be a complicated nature of the non-convex bijectivity constraint (3.14). During the solution process, it was typical for the objective function $q(\mathbf{G})$ to decrease in a non-monotonous way. It is possible that by using a specialized optimization solver instead of a general one the number of iterations can be significantly reduced.

3.2.4 Elliptic grid generation

Elliptic grid generation (EGG) is an example of a PDE-based approach to the single-patch domain parametrization problem. Let $\mathbf{G}(\xi,\eta) = (x(\xi,\eta), y(\xi,\eta))^T$ be an unknown parametrization of the domain Ω. The idea behind elliptic grid generation is to characterize $\mathbf{G}$ by imposing Laplace's equation on its inverse $\mathbf{G}^{-1}(x,y) = (\xi(x,y), \eta(x,y))^T$:

$$\Delta_{\mathbf{x}}\xi = 0 \text{ in } \Omega, \tag{3.20}$$

$$\Delta_{\mathbf{x}}\eta = 0 \text{ in } \Omega, \tag{3.21}$$

$$(\xi,\eta)^T = \partial\mathbf{G}^{-1} \text{ on } \partial\Omega. \tag{3.22}$$

Here, the Dirichlet boundary condition (3.22) makes sure that $\mathbf{G}^{-1}$ maps $\partial\Omega$ onto $\partial[0,1]^2$. Note that although system (3.20–3.22) is a simple linear problem, one cannot solve it using the isogeometric Galerkin method since $\mathbf{G}$ is unknown. Because of that and because one is not usually interested in the inverse mapping $\mathbf{G}^{-1}$, problem (3.20–3.22) is inverted for x and y. The resulting system reads

$$g_{22}x_{\xi\xi} - 2g_{12}x_{\xi\eta} + g_{11}x_{\eta\eta} = 0 \text{ in } [0,1]^2, \tag{3.23}$$

$$g_{22}y_{\xi\xi} - 2g_{12}y_{\xi\eta} + g_{11}y_{\eta\eta} = 0 \text{ in } [0,1]^2, \tag{3.24}$$

$$(x,y)^T = \partial\mathbf{G} \text{ on } \partial[0,1]^2. \tag{3.25}$$

Problem (3.23–3.25) is a complex coupled system of nonlinear PDEs. On the bright side, it is set in a geometrically simple domain—the parametric unit square $[0,1]^2$. Problem (3.23–3.25) is usually solved using Newton's method, to which end it has to be linearized. I refer to [29] for details.

It can be shown that the solution of (3.23–3.25) is equal to the unique minimizer of the Winslow quality measure (3.19) and is, therefore, guaranteed to be bijective. Just like for the Winslow-based optimization, the numerical approximation obtained by solving the system using NURBS may not be bijective if the corresponding NURBS basis is not fine enough. Despite their equivalence, I recommend using elliptic grid generation instead of the Winslow-based optimization since, at least in my experience, an iterative solution method (for example, Newton's method) for problem (3.23–3.25) requires a much smaller number of iterations to converge. In the performed numerical tests, it ranged from three to four for simple domains and from eight to ten for complex domains. Thus, EGG is computationally cheaper than the optimization-based methods but is still much more expensive that the linear methods[5]. Note that, much like the optimization-based methods, EGG requires an initial guess for Newton's method. To that end, the spring model or the Coons patch can be used.

[5]In this comparison, I implicitly assume that the complexity of iterations for EGG and optimization is roughly the same for both methods. This is justified because iteration of both methods boil down to solution of linear systems which size depends mainly on the number of control points.

3.3 Parametrization by mesh deformation

All the parametrization methods introduced above seem to struggle when applied to complex domains. In this section, I propose a novel PDE-based method to the single-patch domain parametrization problem which uses equations of nonlinear elasticity. I refer to it as the mesh deformation approach. Although the proposed approach is nonlinear, it is far less computationally expensive than the optimization-based methods and is able to produce high-quality parametrizations for complex domains. After outlining the main idea of the approach, I provide a short introduction into continuum mechanics which is necessary to describe further details. Then, I formulate an efficient and robust algorithm for solving the equations of nonlinear elasticity as well as several variations of this algorithm. The content of this section is mainly based on [31].

3.3.1 Method description

The idea of the mesh deformation method is to avoid constructing a parametrization for a complex domain Ω right away. Instead, one could simplify Ω and obtain a domain Ω_0 which is, hopefully, simple enough so that it can be parametrized using an inexpensive method, for example, the Coons patch. After that, the *simplified domain* Ω_0 is deformed in such a way that its boundary coincides with the boundary of the *target domain* $\partial\Omega$. The resulting deformed configuration of Ω_0 defines a parametrization $\mathbf{G}$ of Ω, see Figure 3.5.

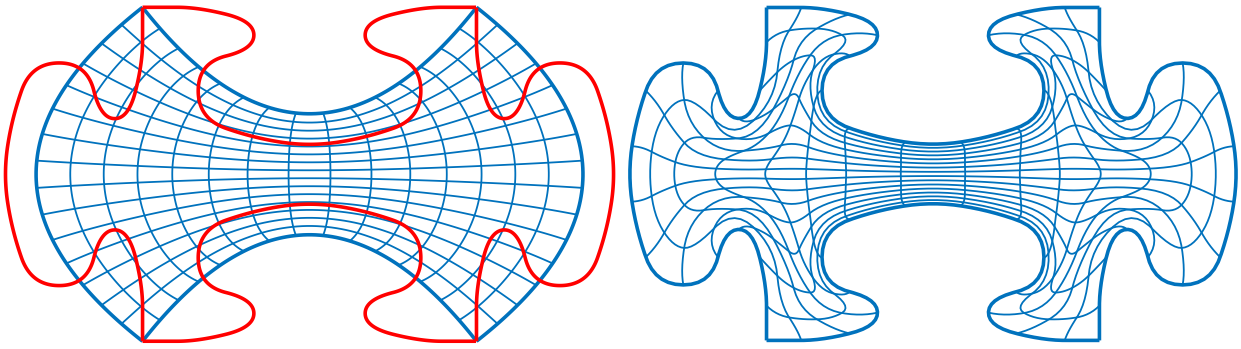

Figure 3.5. Idea of the mesh deformation method: construct a parametrization for the simplified domain (left) and deform it. The resulting deformed configuration of the simplified domain defines a parametrization for the target domain (right).

Concretely, by simplifying a given domain Ω, I understand changing control points of the corresponding boundary curves $\boldsymbol{\gamma}_S$, $\boldsymbol{\gamma}_N$, $\boldsymbol{\gamma}_W$ and $\boldsymbol{\gamma}_E$ in a certain way. This yields new boundary curves $\boldsymbol{\gamma}_S^0$, $\boldsymbol{\gamma}_N^0$, $\boldsymbol{\gamma}_W^0$ and $\boldsymbol{\gamma}_E^0$ which together define a simplified domain Ω_0. What is important is that the new curves inherit NURBS bases of the original boundary curves. Let $\mathbf{G}_0 : [0,1]^2 \to \Omega_0$ denote a parametrization of the simplified domain Ω_0. Since boundary curves of Ω_0 and Ω share the same NURBS basis, so do parametrizations $\mathbf{G}_0$ and $\mathbf{G}$:

$$\mathbf{G}_0(\boldsymbol{\xi}) = \sum_{i=1}^{n} \mathbf{c}_i^0 N_i(\boldsymbol{\xi}) \qquad \text{and} \qquad \mathbf{G}(\boldsymbol{\xi}) = \sum_{i=1}^{n} \mathbf{c}_i N_i(\boldsymbol{\xi}). \tag{3.26}$$

Here, the core assumption is that Ω_0 is so simple that construction of a bijective parametrization $\mathbf{G}_0$ is trivial. In Section 3.3.2, I describe a particular approach I suggest using for domain simplification; for now, let us assume that $\mathbf{c}_i^0$ are known.

The next step is to construct a *deformation mapping* $\mathbf{D} : \Omega_0 \to \Omega$. $\mathbf{D}$ can be arbitrary but it has to fulfill the following two conditions. First, it has to map the simplified domain's boundary $\partial\Omega_0$ onto the original domain's boundary $\partial\Omega$. In particular,

$$\mathbf{D}(\partial\mathbf{G}_0(\boldsymbol{\xi})) = \partial\mathbf{G}(\boldsymbol{\xi}) \tag{3.27}$$

has to hold. Second, $\mathbf{D}$ has to be bijective: $J(\mathbf{D}) = \det \nabla_{\mathbf{x}}\mathbf{D} > 0$. If $\mathbf{D}$ is available, we can define $\mathbf{G}$ as

$$\mathbf{G} = \mathbf{D} \circ \mathbf{G}_0. \tag{3.28}$$

Note that if both $\mathbf{G}_0$ and $\mathbf{D}$ are bijective, then $\mathbf{G}$ is bijective since $J(\mathbf{G}) = J(\mathbf{D})J(\mathbf{G}_0) > 0$.

To characterize the deformation mapping $\mathbf{D}$, I suggest using an isogeometric *displacement field* $\mathbf{u}_h : \Omega_0 \to \mathbb{R}^2$ such that for any point $\mathbf{x} \in \Omega_0$

$$\mathbf{D}(\mathbf{x}) = \mathbf{x} + \mathbf{u}_h(\mathbf{x}). \tag{3.29}$$

The displacement field $\mathbf{u}_h$ is constructed using the NUBRS basis of $\mathbf{G}_0$:

$$\mathbf{u}_h(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{d}_i N_i(\mathbf{x}). \tag{3.30}$$

The displacement control points $\mathbf{d}_i$ can be chosen arbitrarily as long as the corresponding deformation mapping is bijective and (3.27) holds. The boundary control points are predetermined by

$$\mathbf{d}_i = \mathbf{c}_i - \mathbf{c}_i^0 \;\; \forall i \in \mathcal{B}. \tag{3.31}$$

In order to determine the interior control points, I propose solving equations of nonlinear elasticity as described in Sections 3.3.4 and 3.3.5. Once the displacement field $\mathbf{u}_h$ is constructed, we can plug it (3.30) and the parametrization of the simplified domain (3.26) into the composition (3.28) to obtain the parametrization for the target domain:

$$\mathbf{G}(\boldsymbol{\xi}) = \mathbf{G}_0(\boldsymbol{\xi}) + \mathbf{u}(\mathbf{G}_0(\boldsymbol{\xi})) = \sum_{i=1}^{n} (\mathbf{c}_i^0 + \mathbf{d}_i) N_i(\boldsymbol{\xi}). \tag{3.32}$$

One could argue that the mesh deformation method just replaces a complicated problem of domain parametrization by a no less complex problem of computing a bijective deformation mapping with large displacements. Luckily, as I demonstrate in the following, the deformation mapping $\mathbf{D}$ can be computed in an efficient and robust way by applying the ideas from the continuation methods theory to the equations of nonlinear elasticity.

The proposed mesh deformation approach is not entirely new. Similar ideas are used to generate curvilinear meshes from piecewise linear triangulations in [67], however, only small deformations are considered. In [68], the mesh deformation approach is applied to construct volumetric meshes consisting of a T-spline surface layer and a core of Lagrangian elements. However, the best of my knowledge, the mesh deformation approach has not been applied to construct tensor-product NURBS parametrizations before [31].

3.3.2 Domain simplification

The choice of the simplified domain Ω_0 is a rather empirical step which directly affects the quality of the resulting parametrization $\mathbf{G}$. Ideally, Ω_0 should be simple enough−so that it is possible to construct a bijective parametrization $\mathbf{G}_0$ using a linear method such as the Coons patch−and yet geometrically close enough to Ω−so that the complexity of computing the deformation mapping $\mathbf{D}$ does not eclipse the complexity of the original parametrization problem for Ω. All possible choices of Ω_0 form a spectrum with a quadrilateral spanned by the four corners of the target domain Ω, called the *corner-quad*, on one end and Ω itself on another. In the following, I describe a basic simplification procedure which allows to generate a range of domains lying along this spectrum.

The simplification procedure involves projecting the boundary curves $\boldsymbol{\gamma}_S$, $\boldsymbol{\gamma}_N$, $\boldsymbol{\gamma}_W$ and $\boldsymbol{\gamma}_E$ onto a coarse NURBS basis[6] in the L^2-sense. Since a coarse basis can only produce objects of limited complexity, the projection is a simplified version of the original curve. After that, I re-express the simplified curve in terms of the original basis. To that end, I either apply the refinement algorithms such as knot insertion and degree elevation or project the simplified curve back onto the original basis in the L^2-sense. This step preserves the shape, either exactly or accurately enough, and makes sure that the original and simplified boundaries share the same basis. One more important aspect: since it is sometimes convenient to preserve the location of the four corners of Ω, I simplify each of the boundary curves separately and fix the location of their end points.

Concretely, let $\boldsymbol{\gamma}(\xi) : [0, 1] \to \mathbb{R}^2$ be one of the boundary curves,

$$\boldsymbol{\gamma}(\xi) = \sum_{i=1}^{n} \mathbf{c}_i N_i(\xi), \tag{3.33}$$

with n control points $\mathbf{c}_i$ and a NURBS basis consisting of n basis functions N_i of degree p. Additionally, let m NURBS basis functions of degree q form a coarse NURBS basis used for the L^2-projection. Then, I construct a *coarse simplification* $\widetilde{\boldsymbol{\gamma}}^0$ of $\boldsymbol{\gamma}$,

$$\widetilde{\boldsymbol{\gamma}}^0(\xi) = \sum_{i=1}^{m} \widetilde{\mathbf{c}}_i^0 M_i(\xi), \tag{3.34}$$

by solving the following linear system:

$$\mathbf{M}\widetilde{\mathbf{C}}^0 = \boldsymbol{\Gamma}, \tag{3.35}$$

$$\widetilde{\mathbf{c}}_1^0 = \mathbf{c}_1, \tag{3.36}$$

$$\widetilde{\mathbf{c}}_m^0 = \mathbf{c}_n. \tag{3.37}$$

Here, $\mathbf{M}$ is a $m \times m$ projection mass matrix defined for the basis functions $M_i(\xi)$ in a manner similar to (2.35). $\widetilde{\mathbf{C}}^0$ is a $m \times 2$ matrix containing unknown coordinates of the control points, and $\boldsymbol{\Gamma}$ is a $m \times 2$ which entries are defined as

$$(\boldsymbol{\Gamma})_{ij} = \int_0^1 (\boldsymbol{\gamma})_j M_i d\xi, \quad i = 1, \ldots, m, \quad j = 1, 2. \tag{3.38}$$

Boundary conditions (3.36–3.37) are included to make sure that the endpoints of $\boldsymbol{\gamma}$ and $\widetilde{\boldsymbol{\gamma}}^0$ coincide. A nice property of the L^2-projection is that it minimizes the distance between points of the original and the simplified curves corresponding to the same parametric value, namely between $\boldsymbol{\gamma}(\xi)$ and

[6] This means that a basis with relatively few elements.

$\widetilde{\boldsymbol{\gamma}}^0(\xi)$. This in turn minimizes the prescribed boundary displacement (3.31), making it easier to compute the deformation $\mathbf{D}$.

Next, the coarse simplification $\widetilde{\boldsymbol{\gamma}}^0$ is re-expressed in terms of the original NURBS basis N_i. The result is a *fine simplification* $\boldsymbol{\gamma}^0$,

$$\boldsymbol{\gamma}^0(\xi) = \sum_{i=1}^{n} \mathbf{c}_i^0 N_i(\xi), \tag{3.39}$$

with n control points $\mathbf{c}_i^0$. This step can be performed by applying knot insertion and degree elevation if the polynomial degree q of the coarse simplification is lower than the degree p of the original curve, and if the knot vector of the coarse simplification is a subset of the knot vector of the original curve. If either of this conditions does not hold, it is easier to project $\widetilde{\boldsymbol{\gamma}}^0$ in the L^2-sense onto the original basis by solving a linear system similar to (3.35–3.37). Although the projection may slightly alter the shape of $\widetilde{\boldsymbol{\gamma}}^0$ if it does not lie in the span of the original NURBS basis, it preserves the overall simplicity of the curve.

I use the southern boundary curve $\boldsymbol{\gamma}_S$ of the puzzle piece domain from above to demonstrate the performance of the described simplification procedure. The original curve is quadratic with $n = 17$ control points. In order to study the effect that a pair of parameters (q, m) has on the simplified shape, $\boldsymbol{\gamma}_S$ has been simplified using several coarse bases with varying parameters. Figure 3.6 shows the results.

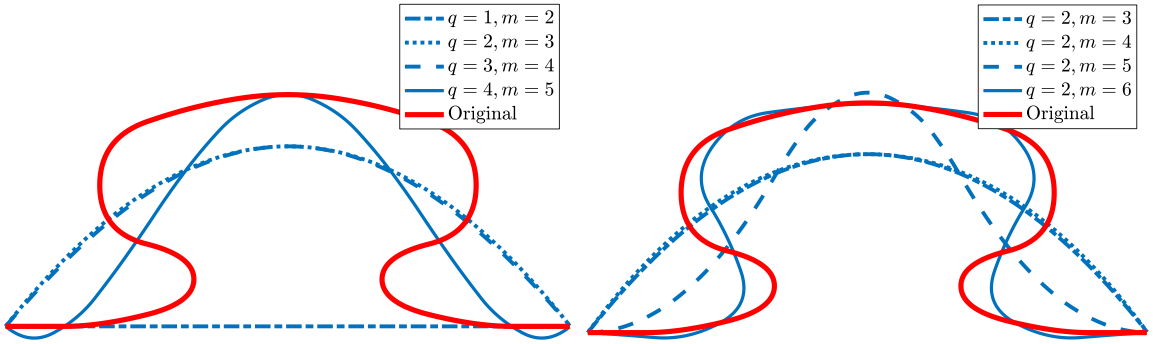

Figure 3.6. Example of applying the simplification technique to the southern boundary curve of the puzzle piece domain. The original curve has a polynomial degree $p = 2$ and $n = 17$ control points. Coarse bases with various pairs of parameters (q, m) have been used for the simplification.

Once the entire boundary of the target domain Ω is simplified, I apply the Coons patch to parametrize the obtained simplified domain Ω_0. For the Coons patch to produce a bijective parametrization, Ω_0 has to be simple enough. In the considered puzzle piece example, coarse NURBS bases with parameters $(q = 1, m = 2)$, $(q = 2, m = 3)$ and $(q = 3, m = 4)$ produce sufficiently simple geometries. Other choices result in Ω_0 being too complex for the Coons patch to succeed, see Figure 3.7. Among the three acceptable pairs of parameters, the first one, namely $(q = 1, m = 2)$, corresponds to replacing the original domain Ω with the corner-quad. If the corner-quad is convex, the Coons patch always produces a bijective parametrization $\mathbf{G}_0$. The case when the corner-quad is not convex may indicate that Ω is too complex and should not be parametrized using a singe patch. Although $(q = 1, m = 2)$ almost always produces a suitable simplified domain, there often exist better options. In particular, pairs $(q = 2, m = 3)$ and $(q = 3, m = 4)$[7] generate a domain which is still simple enough for the Coons patch to succeed, but is much closer to the

[7]Both generate very similar simplified domains, so from now one I only consider the former one.

original shape than the corner-quad produced by $(q = 1, m = 2)$. This choice seems to represent a reasonable trade-off in complexity between the initial construction of the simplified domain and the ensuing deformation step.

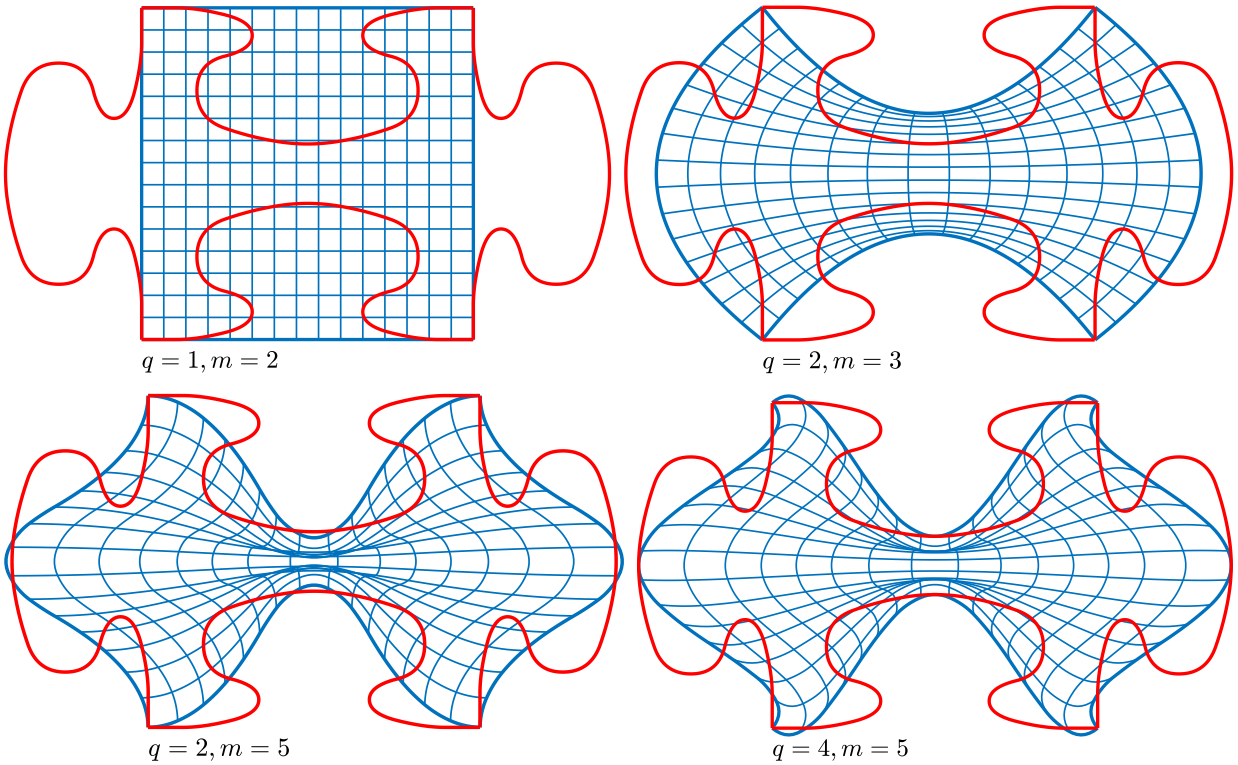

Figure 3.7. Parametrizations of various simplified domain constructed using the Coons patch. The simplified domains have been produced using coarse bases with various pairs of parameters (q, m).

Although a pair of parameters $(q = 2, m = 3)$ generates a simplified domain which can be parametrized using the Coons patch, a congestion of mesh lines in the narrow middle part of the domain indicates that the resulting parametrization $\mathbf{G}_0$ is barely bijective. Because of that, the Coons patch can sometimes be replaced by other parametrization approaches. Ideally, they should be computationally inexpensive, or at least not more expensive than the ensuing deformation step. As such, the spring model or EGG can be applied. Although the former, just like the Coons patch, does not guarantee bijectivity of the result, it can sometimes produce a better output than the Coons patch. In contrast to that, EGG always produces a bijective parametrization, so its relatively high computational cost is justified. Finally, a recursive approach may be employed where the mesh deformation method itself is used to construct a parametrization for the simplified domain[8]. To that end, the already simplified domain has to be further simplified. For the considered $(q = 2, m = 3)$ domain, I have used the corner-quad produced by the coarse basis $(q = 1, m = 2)$ as a simplified domain, and the resulting parametrization has turned out to be the best with respect to the overall uniformity of the mesh, see Figure 3.8. Note that this double-deformation procedure is not equivalent to deforming the corner-quad directly into the target domain (see Section 3.4 for examples).

[8] I am aware of the ouroborus situation caused by using the not yet introduced domain deformation step in order to implement the domain simplification step. Nevertheless, here is a good place to mention this possibility.

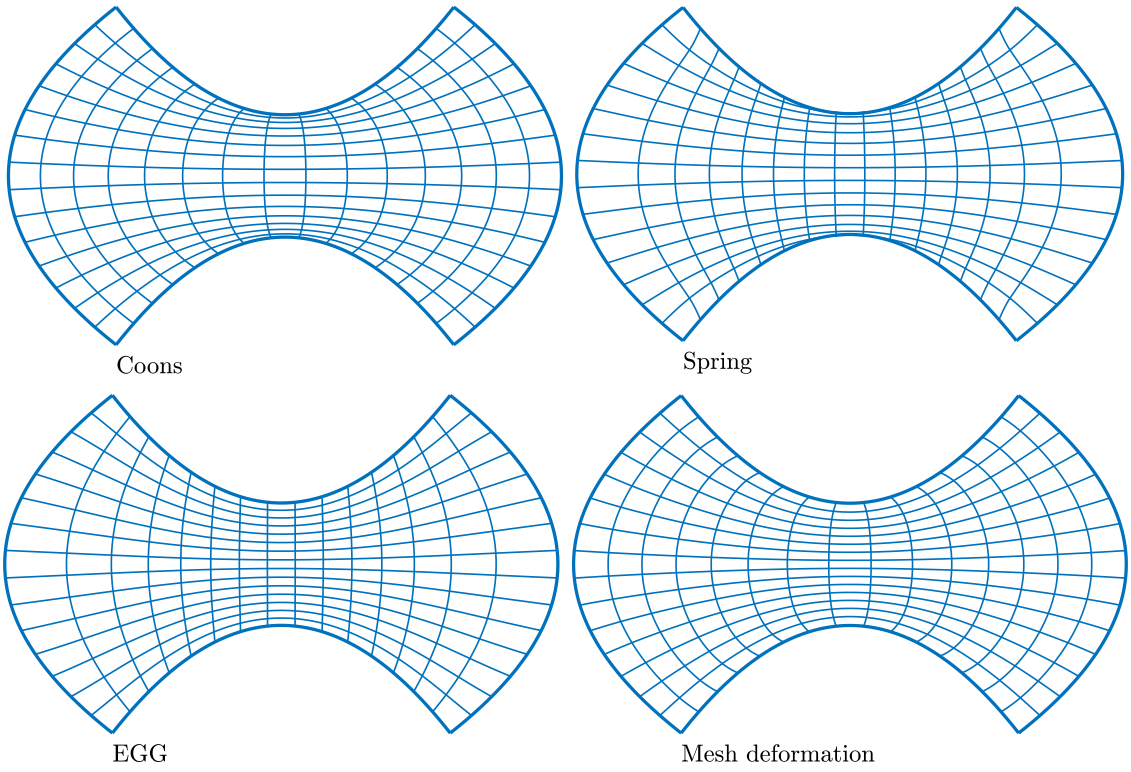

Figure 3.8. Simplified domain ($q = 2, m = 3$) for the puzzle piece example. Comparison of four different parametrization techniques: Coons patch, spring model, EGG and mesh deformation from the corner-quad.

3.3.3 Nonlinear elasticity in a nutshell

The following is a brief introduction into solid mechanics based on [30]. Here, the focus lies on the ingredients necessary for describing the deformation algorithm I use to deform the simplified domain Ω_0 into the target domain Ω. The presented theory describes the behavior of a three-dimensional elastic body. When applied to the planar case, it corresponds to the so-called *plain strain* formulation of 2D elasticity.

Let a solid body in its initial undeformed state occupy a domain $\Omega_0 \subset \mathbb{R}^3$, which is referred to as a *reference configuration*. Assume that the body undergoes a deformation $\mathbf{D}$, after which it occupies a domain $\Omega \subset \mathbb{R}^3$. Then Ω is called a *deformed configuration* of the body. In the context of solid mechanics, one refers to a point $\widehat{\mathbf{x}} \in \Omega_0$ as a *material point*. Behavior of the solid body is described by studying the motion of material points that comprise it[9]. For each material point $\widehat{\mathbf{x}}$, its position $\mathbf{x} = \mathbf{D}(\widehat{\mathbf{x}})$ in the deformed configuration Ω is expressed in terms of the differentiable displacement vector field $\mathbf{u} : \Omega_0 \to \mathbb{R}^3$ such that $\mathbf{D}(\widehat{\mathbf{x}}) = \widehat{\mathbf{x}} + \mathbf{u}(\widehat{\mathbf{x}})$. Note that $\mathbf{u}$ can describe a *rigid body motion* (a combination of translation and rotation) which in fact includes no deformation. The information about whether Ω is actually deformed or not is contained largely in the *deformation gradient* $\mathbf{F} : \Omega_0 \to \mathbb{R}^{3\times 3}$ defined as

$$\mathbf{F} = \nabla_{\widehat{\mathbf{x}}}\mathbf{D} = \mathbf{I} + \nabla_{\widehat{\mathbf{x}}}\mathbf{u}. \tag{3.40}$$

[9]This approach is called the *Lagrangian framework*, as opposed to the *Eulerian framework* which is used, for example, in fluid dynamics. Within the Eulerian framework, one does not follow the motion of material points, but instead focuses on what happens in fixed spatial points.

Here, $\mathbf{I}$ is a 3×3 identity matrix. The determinant of $\mathbf{F}-$denoted[10] by $J-$is a local measure of relative volume change between the deformed and the reference configurations. If $J > 1$, the body expends locally, whereas $J < 1$ means that it contracts locally. For a special case of an *incompressible material*, $J \equiv 1$ must hold. Since the self-penetration is excluded in order to remain consistent with the everyday physical intuition, the deformation $\mathbf{D}$ must be bijective, and the condition $J > 0$ has to hold. In what follows, two more geometric objects describing the deformation $\mathbf{D}$ are important: the *Green-Lagrange strain tensor* $\mathbf{E} : \Omega_0 \to \mathbb{R}^{3\times3}$ defined as

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}) \tag{3.41}$$

and the *right Cauchy-Green tensor* $\mathbf{C} : \Omega_0 \to \mathbb{R}^{3\times3}$ defined as

$$\mathbf{C} = \mathbf{F}^T\mathbf{F}. \tag{3.42}$$

Both tensors are symmetric.

The behavior of a solid body is defined by the internal forces arising in it in response to the deformation caused by external forces. The body is called *elastic* if it returns to the reference configuration once the external forces vanish. According to Cauchy's theorem, the internal forces can be characterized by a 3×3 symmetric tensor $\boldsymbol{\sigma}$ called the *Cauchy stress tensor*. Concretely, the force acting on a given surface within or on the boundary of the body if equal to $\boldsymbol{\sigma} \cdot \mathbf{n}$, where $\mathbf{n}$ is the surface normal. The Cauchy stress tensor describes the internal forces in the deformed configuration. It is convenient, however, to be able to express these forces with respect to the reference configuration as well. To that end, the *first Piola-Kirchhoff stress tensor* $\mathbf{P}$ is introduced, which is related to the Cauchy stress tensor $\boldsymbol{\sigma}$ by

$$\mathbf{P} = J\boldsymbol{\sigma}\mathbf{F}^{-T}. \tag{3.43}$$

Note that $\mathbf{P}$ is not symmetric. Because of that, a symmetric tensor $\mathbf{S}$, called the *second Piola-Kirchhoff stress tensor*, is defined as

$$\mathbf{S} = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T}. \tag{3.44}$$

The difference between the Piola-Kirchhoff and Cauchy stress tensors is the first example of the two ways to describe the deformation of a solid body: either in the reference or in the deformed configuration. In order to stress the difference, I use $\widehat{\mathbf{x}}$ and $\mathbf{x}$ to denote coordinates in the two configurations. Since the deformed configuration is usually not known a priori, it is convenient to study the deformation of an elastic body with respect to the reference configuration.

So far, I have not mentioned the relation between the stress tensors $\mathbf{P}$, $\mathbf{S}$ or $\boldsymbol{\sigma}$ and the displacement field $\mathbf{u}$. This relation, which describes exactly what forces arise in the body in response to the deformation, is the center point of solid mechanics and is called the *material law*. The material law is chosen according to a particular application. When modeling an elastic body, two popular choices are the *St. Venant-Kirchhoff* material law

$$\mathbf{S} = \lambda \operatorname{tr}(\mathbf{E})\mathbf{I} + 2\mu\mathbf{E} \tag{3.45}$$

and the *neo-Hookean* material law, a particular variation of which is

$$\mathbf{S} = \lambda \ln J\mathbf{C}^{-1} + \mu(\mathbf{I} - \mathbf{C}^{-1}). \tag{3.46}$$

[10]Each object introduced in this subsection depends on $\mathbf{u}$, which is a primary independent variable. I explicitly remind the reader of this dependence when it is necessary or in order to avoid confusion. For example, I write $\mathbf{F}(\mathbf{u})$. When it is possible, though, I shorten the notation and write simply $\mathbf{F}$.

Both material laws include the *Lamé parameters* λ and μ which are constitutive parameters describing physical properties of the material. They can be computed from *Young's modulus* E and *Poisson's ratio* ν of the material as

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \quad \text{and} \quad \mu = \frac{E}{2(1+\nu)}. \tag{3.47}$$

The St. Venant-Kirchhoff material law is linear in $\mathbf{E}$, whereas the neo-Hookean material law is nonlinear. What is more, due to the presence of $\ln J$, the neo-Hookean material law explicitly preserves bijectivity of the deformation since J is not allowed to be negative. I rely on this property in Subsections 3.3.4 and 3.3.5 to compute deformation mappings which can be used to construct bijective parametrizations for Ω.

Having introduced all the necessary tools, I can formulate the local balance equations of linear momentum that define the equilibrium state of an elastic body. The actual balance of forces is achieved in the deformed configuration, where the equations take form

$$\operatorname{div}_{\mathbf{x}} \boldsymbol{\sigma}(\mathbf{u}) = \mathbf{g} \text{ in } \Omega. \tag{3.48}$$

Unfortunately, the deformed configuration is not known beforehand, so the equations (3.48) have to be reformulated in the reference configuration. This is achieved using the first Piola-Kirchhoff stress tensor, and the result is

$$\operatorname{div}_{\widehat{\mathbf{x}}} \mathbf{P}(\mathbf{u}) = \mathbf{g} \text{ in } \Omega_0. \tag{3.49}$$

I refer to these equations as the *stationary equations of nonlinear elasticity*. Here, $\mathbf{g}$ is a volumetric force acting on the body. I assume that $\mathbf{g}$ is the same with respect to the reference and the deformed configurations. Most often, $\mathbf{g}$ represents the gravitation. By solving equations (3.49), the unknown displacement field $\mathbf{u}$ can be found. Equations (3.49) are nonlinear with two sources of nonlinearity: geometric, coming from the strain tensors $\mathbf{E}$ or $\mathbf{C}$, and material, if the material law is nonlinear. Both sources manifest themselves through the structure of the corresponding linearized system used in the iterative solution procedure.

Additionally, equations (3.49) have to be equipped with suitable boundary conditions. Usually, it is a combination of Dirichlet and Neumann boundary conditions:

$$\mathbf{u} = \mathbf{u}_{\mathcal{D}} \text{ on } \partial\Omega_0^{\mathcal{D}} \qquad \text{and} \qquad \mathbf{P}(\mathbf{u}) \cdot \mathbf{n} = \mathbf{f} \text{ on } \partial\Omega_0^{\mathcal{N}}. \tag{3.50}$$

Here, $\mathbf{u}_{\mathcal{D}}$ is a prescribed boundary displacement, and $\mathbf{f}$ is a prescribed surface load. A function $\mathbf{u} \in C^2(\Omega_0) \cap C^0(\bar{\Omega}_0)$ satisfying (3.49–3.50) is called a classical solution of the stationary equations of nonlinear elasticity.

Linear elasticity

Equations (3.49) can model large deformations. However, when only small deformations are considered, the equations can be simplified. Concretely, if the equations of nonlinear elasticity are linearized at $\mathbf{u} = \mathbf{0}$, then the Green-Lagrange strain tensor $\mathbf{E}$ reduces to the *linear strain tensor* $\boldsymbol{\varepsilon}$,

$$\boldsymbol{\varepsilon} = \frac{1}{2}\big(\nabla_{\widehat{\mathbf{x}}}\mathbf{u} + \nabla_{\widehat{\mathbf{x}}}\mathbf{u}^T\big), \tag{3.51}$$

both the St. Venant-Kirchhoff and the neo-Hookean material laws reduce to *Hooke's law*

$$\boldsymbol{\sigma} = \lambda \operatorname{tr}(\boldsymbol{\varepsilon})\mathbf{I} + 2\mu\boldsymbol{\varepsilon}, \tag{3.52}$$

and equations (3.49) become the *equations of linear elasticity*:

$$-\operatorname{div}_{\widehat{\mathbf{x}}}\boldsymbol{\sigma}(\mathbf{u}) = \mathbf{g} \text{ in } \Omega_0. \tag{3.53}$$

In Section 3.3.5, I describe how, quite counter-intuitively, equations (3.53) can be used to compute large bijective deformation suitable for constructing isogeometric parametrizations.

3.3.4 Newton's method for nonlinear elasticity

Equipped with the theory of nonlinear elasticity, we can tackle the domain deformation step of the mesh deformation method. In order to deform the simplified domain Ω_0 into the target domain Ω, I propose solving the stationary equations of nonlinear elasticity (3.49) with the neo-Hookean material law (3.46) and no volumetric force. Condition (3.31) on the boundary control points for the isoparametric displacement field translates into a prescribed boundary displacement displacement $\mathbf{u}_{\mathcal{D}}$. The resulting system reads

$$-\operatorname{div}_{\widehat{\mathbf{x}}}\mathbf{P}(\mathbf{u}) = \mathbf{0} \text{ in } \Omega_0, \tag{3.54}$$

$$\mathbf{u} = \mathbf{u}_{\mathcal{D}} \text{ on } \partial\Omega_0. \tag{3.55}$$

In order to solve the system (3.54–3.55) using the isogeometric Galerkin method, we need to rewrite it in a weak from. Let $\mathcal{V} = \{\mathbf{w} \in H^1(\Omega_0)^3 \mid \mathbf{w} = \mathbf{u}_{\mathcal{D}} \text{ on } \partial\Omega_0\}$ be the trial solution space and the weighting function space $\mathcal{V}_0$ be defined as $\{\mathbf{w} \in H^1(\Omega_0)^3 \mid \mathbf{w} = \mathbf{0} \text{ on } \partial\Omega_0\}$. Additionally, let : denote the Frobenius inner product. Then the weak form of the system (3.54–3.55) is

$$\text{find } \mathbf{u} \in \mathcal{V} \text{ such that } \quad R(\mathbf{u}, \mathbf{w}) = \int_{\Omega_0} \mathbf{S}(\mathbf{u}) : \delta\mathbf{E}(\mathbf{u})[\mathbf{w}] d\widehat{\mathbf{x}} = 0 \quad \forall \mathbf{w} \in \mathcal{V}_0. \tag{3.56}$$

Here, $R(\mathbf{u}, \mathbf{w})$ is called the *residual*, and $\delta\mathbf{E}(\mathbf{u})[\mathbf{w}] = \frac{1}{2}\big(\mathbf{F}(\mathbf{u})^T\nabla_{\widehat{\mathbf{x}}}\mathbf{w} + \nabla_{\widehat{\mathbf{x}}}\mathbf{w}^T\mathbf{F}(\mathbf{u})\big)$ is called the *variation of the Green-Lagrange strain tensor*.

Equation (3.56) is nonlinear in $\mathbf{u}$. One way to solve it by using *Newton's method*. For a sufficiently smooth function $f(x) : \mathbb{R} \to \mathbb{R}$, Newton's method successively computes increasingly better approximations x^s to a root of f as

$$x^{s+1} = x^s - \frac{f(x^s)}{f'(x^s)}. \tag{3.57}$$

When applying Newton's method to compute roots of a functional such as $R(\mathbf{u}, \mathbf{w})$, there exists two alternatives: R can be discretized first and then linearized, or vice versa. Here, I opt for the latter option as a more illustrative one. The Taylor expansion at $R(\mathbf{u}, \mathbf{w})$ with the *displacement increment* $\Delta\mathbf{u}$ yields

$$R(\mathbf{u} + \Delta\mathbf{u}, \mathbf{w}) = R(\mathbf{u}, \mathbf{w}) + DR(\mathbf{u}, \mathbf{w}) \cdot \Delta\mathbf{u} + o(||\Delta\mathbf{u}||), \tag{3.58}$$

where the *directional derivative* $DR(\mathbf{u}, \mathbf{w}) \cdot \Delta\mathbf{u}$ is computed as

$$DR(\mathbf{u}, \mathbf{w}) \cdot \Delta\mathbf{u} = \int_{\Omega_0} \Big(\nabla_{\widehat{\mathbf{x}}}\Delta\mathbf{u}\mathbf{S}(\mathbf{u}) : \nabla_{\widehat{\mathbf{x}}}\mathbf{w} + \mathbb{C}(\mathbf{u})\delta\mathbf{E}(\mathbf{u})[\Delta\mathbf{u}] : \delta\mathbf{E}(\mathbf{u})[\mathbf{w}]\Big) d\widehat{\mathbf{x}}. \tag{3.59}$$

Here, $\mathbb{C} = 2\frac{d\mathbf{S}}{d\mathbf{C}}$ is the forth order *elasticity tensor* whose components, in case of the neo-Hookean material law, are given by

$$C_{abcd} = \lambda(\mathbf{C}^{-1})_{ab}(\mathbf{C}^{-1})_{cd} + (\mu - \lambda \ln J)\Big((\mathbf{C}^{-1})_{ac}(\mathbf{C}^{-1})_{bd} + (\mathbf{C}^{-1})_{ad}(\mathbf{C}^{-1})_{bc}\Big). \tag{3.60}$$

For the St. Venant-Kirchhoff material law, the components of $\mathbb{C}$ are given by

$$C_{abcd} = \lambda\delta_{ab}\delta_{cd} + \mu\Big(\delta_{ac}\delta_{bd} + \delta_{ad}\delta_{bc}\Big), \tag{3.61}$$

where δ_{ij} is the Kronecker delta. Note that $DR(\mathbf{u}, \mathbf{w}) \cdot \Delta\mathbf{u}$ is a symmetric bilinear form with respect to $\Delta\mathbf{u}$ and $\mathbf{w}$.

With all the necessary tools defined, we can formulate Newton's method for problem (3.54–3.55). Let $\mathbf{u}^0 \in \mathcal{V}$ be an *initial guess*. Given an intermediate solution $\mathbf{u}^s \in \mathcal{V}$ at iteration $s \geqslant 0$, a displacement increment $\Delta\mathbf{u}^{s+1} \in \mathcal{V}_0$ is computed as a solution to the following weak problem:

$$\text{find } \Delta\mathbf{u}^{s+1} \in \mathcal{V}_0 \text{ such that } \; DR(\mathbf{u}^s, \mathbf{w}) \cdot \Delta\mathbf{u}^{s+1} = -R(\mathbf{u}^s, \mathbf{w}) \;\; \forall \mathbf{w} \in \mathcal{V}_0. \tag{3.62}$$

After that, the next intermediate solution $\mathbf{u}^{s+1} \in \mathcal{V}$ is defined as $\mathbf{u}^s + \Delta\mathbf{u}^{s+1}$. Note that all intermediate solutions satisfy the Dirichlet boundary condition (3.55), so the increment displacements have to be zero on the boundary.

The procedure continues until the convergence criteria

$$||\Delta\mathbf{u}^s||_{L^2} < \varepsilon_{abs} \;\; \text{or} \;\; \frac{||\Delta\mathbf{u}^s||_{L^2}}{||\mathbf{u}^s||_{L^2}} < \varepsilon_{rel} \tag{3.63}$$

are met. The last solution $\mathbf{u}^s$ is accepted as a solution to the original system (3.54–3.55).

Discretization and linear system assembly

Weak problem (3.62) is discretized using the isogeometric Galerkin method. Although the process is ideologically similar to the isogeometric discretization of Poisson's equation 2.3.2, I briefly outline it here, as described in [30] and [69]. The isogeometric solution $\mathbf{u}_h = \sum_{i=1}^{n} \mathbf{d}_i N_i(\mathbf{x})$ is constructed using the NURBS basis functions associated with the parametrization $\mathbf{G}_0$ of the reference domain Ω_0. Here, $\mathbf{d}_i = (d_i^1, d_i^2, d_i^3)^T$ are the displacement control points which are split into two parts: unknown control points and fixed Dirichlet control points. The corresponding indices i are collected in sets $\mathcal{F}$ and $\mathcal{D}$. Components of the unknown control points are arranged dimension-wise into a vector $\mathbf{y} \in \mathbb{R}^{3|\mathcal{F}|}$:

$$\mathbf{y} = \big(d_1^1, \ldots, d_{|\mathcal{F}|}^1, \ldots, d_1^3, \ldots, d_{|\mathcal{F}|}^3\big)^T. \tag{3.64}$$

Similarly, components of the Dirichlet control points are arranged into a vector $\mathbf{y}^{\mathcal{D}} \in \mathbb{R}^{3|\mathcal{D}|}$. It is convenient to introduce *vector-valued basis functions* $\mathbf{N}_i^k$ which uniquely correspond to each degree of freedom d_i^k, $k = 1, 2, 3$. $\mathbf{N}_i^k$ are defined as

$$\mathbf{N}_i^1 = \begin{pmatrix} N_i \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{N}_i^2 = \begin{pmatrix} 0 \\ N_i \\ 0 \end{pmatrix} \;\; \text{and} \;\; \mathbf{N}_i^3 = \begin{pmatrix} 0 \\ 0 \\ N_i \end{pmatrix}. \tag{3.65}$$

With $\mathbf{N}_i^k$, the isogeometric solution can be rewritten as $\mathbf{u}_h = \sum_{i=1}^{n} \sum_{k=1}^{3} d_i^k \mathbf{N}_i^k$. The linear span of $\mathbf{N}_i^k$, $i \in \mathcal{F}$ defines the discrete weighting function space $\mathcal{V}_{0,h} \subset \mathcal{V}_0$. The discrete trial solution space $\mathcal{V}_h$ is defined as $\mathbf{u}_{\mathcal{D}} + \mathcal{V}_{0,h}$.

First, let us discretize the weak problem (3.56):

$$\text{find } \mathbf{u}_h \in \mathcal{V}_h \text{ such that } \; R(\mathbf{u}_h, \mathbf{w}) = \int_{\Omega_0} \mathbf{S}(\mathbf{u}_h) : \delta\mathbf{E}(\mathbf{u}_h)[\mathbf{w}] d\widehat{\mathbf{x}} = 0 \;\; \forall \mathbf{w} \in \mathcal{V}_{0,h}. \tag{3.66}$$

Since the residual is linear in $\mathbf{w}$, it is enough to test this equation with every $\mathbf{N}_i^k$, $i \in \mathcal{F}$. It is convenient to test triples $\mathbf{N}_i^1$, $\mathbf{N}_i^2$ and $\mathbf{N}_i^3$ together. To that end for each scalar-valued basis function N_i, I form an auxiliary matrix $\mathbf{B}_i(\mathbf{y}, \mathbf{y}^{\mathcal{D}})$ defined[11] as

$$\mathbf{B}_i(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) = \begin{pmatrix} F_{11}N_{ix} & F_{21}N_{ix} & F_{31}N_{ix} \\ F_{12}N_{iy} & F_{22}N_{iy} & F_{32}N_{iy} \\ F_{13}N_{iz} & F_{23}N_{iz} & F_{33}N_{iz} \\ F_{11}N_{iy} + F_{12}N_{ix} & F_{21}N_{iy} + F_{22}N_{ix} & F_{31}N_{iy} + F_{32}N_{ix} \\ F_{12}N_{iz} + F_{13}N_{iy} & F_{22}N_{iz} + F_{23}N_{iy} & F_{32}N_{iz} + F_{33}N_{iy} \\ F_{11}N_{iz} + F_{13}N_{ix} & F_{21}N_{iz} + F_{23}N_{ix} & F_{31}N_{iz} + F_{33}N_{ix} \end{pmatrix}. \tag{3.67}$$

Here N_{ix}, N_{iy} and N_{iz} are components of the gradient $\nabla_{\widehat{\mathbf{x}}} N_i = (N_{ix}\ N_{iy}\ N_{iz}) = \nabla_{\boldsymbol{\xi}} N_i \nabla_{\boldsymbol{\xi}} \mathbf{G}_0^{-1}$, and F_{ij} are components of the deformation gradient $\mathbf{F}(\mathbf{y}, \mathbf{y}^{\mathcal{D}})$:

$$\mathbf{F}(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) = \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} = \mathbf{I} + \nabla_{\widehat{\mathbf{x}}} \mathbf{u}_h = \mathbf{I} + \nabla_{\boldsymbol{\xi}} \mathbf{u}_h \nabla_{\boldsymbol{\xi}} \mathbf{G}_0^{-1}. \tag{3.68}$$

Additionally, I use the *Voigt notation* which allows to write symmetric second order tensors as vectors and forth order tensors with minor symmetries as matrices. Concretely, the second Piola-Kirchhoff stress tensor $\mathbf{S}$ is a symmetric 3×3 tensor and can be rewritten as a vector with 6 elements:

$$\mathbf{S}_V = (S_{11}, S_{22}, S_{33}, S_{12}, S_{23}, S_{13})^T. \tag{3.69}$$

The elasticity tensor $\mathbb{C}$ is a $6 \times 6 \times 6 \times 6$ forth order tensor and can be rewritten as a 6×6 matrix:

$$\mathbb{C}_V = \begin{pmatrix} C_{1111} & C_{1122} & C_{1133} & C_{1112} & C_{1123} & C_{1113} \\ C_{2211} & C_{2222} & C_{2233} & C_{2212} & C_{2223} & C_{2213} \\ C_{3311} & C_{3322} & C_{3333} & C_{3312} & C_{3323} & C_{3313} \\ C_{1211} & C_{1222} & C_{1233} & C_{1212} & C_{1223} & C_{1213} \\ C_{2311} & C_{2322} & C_{2333} & C_{2312} & C_{2323} & C_{2313} \\ C_{1311} & C_{1322} & C_{1333} & C_{1312} & C_{1323} & C_{1313} \end{pmatrix} \tag{3.70}$$

Now that all the components are ready, we can proceed with the discretization. The result of testing the weak problem (3.66) with each triple $(\mathbf{N}_i^1, \mathbf{N}_i^2, \mathbf{N}_i^3)$, $i \in \mathcal{F}$ is a system of nonlinear equations:

$$\mathbf{R}_i(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) = \int_{\Omega_0} \mathbf{B}_i(\mathbf{y}, \mathbf{y}^{\mathcal{D}})^T \mathbf{S}_V(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) d\widehat{\mathbf{x}} = \mathbf{0}. \tag{3.71}$$

Here, $\mathbf{R}_i(\mathbf{y}, \mathbf{y}^{\mathcal{D}})$ is a vector with three elements.

In a similar fashion, we can obtain a discretization of the directional derivative $DR(\mathbf{u}_h, \mathbf{w}) \cdot \Delta\mathbf{u}_h$. The discrete displacement increment $\Delta\mathbf{u}_h = \sum_{i=1}^{n} \sum_{k=1}^{3} \Delta d_i^k \mathbf{N}_i^k$ belongs to the space $\mathcal{V}_{0,h}$, so the corresponding vector of Dirichlet degrees of freedom $\Delta\mathbf{y}^{\mathcal{D}}$ is zero. The result of evaluating $DR(\mathbf{u}_h, \mathbf{N}_i^k) \cdot \mathbf{N}_j^l$ for $k, l = 1, 2, 3$ is collected in a symmetric 3×3 matrix

$$\mathbf{K}_T^{ij}(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) = \int_{\Omega_0} \mathbf{B}_i(\mathbf{y}, \mathbf{y}^{\mathcal{D}})^T \mathbb{C}_V(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) \mathbf{B}_j(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) d\widehat{\mathbf{x}} + \mathbf{I} \int_{\Omega_0} \nabla_{\widehat{\mathbf{x}}} N_i \mathbf{S}(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) \nabla_{\widehat{\mathbf{x}}} N_j^T d\widehat{\mathbf{x}}. \tag{3.72}$$

[11] Whenever I explicitly write that a certain object depends on $\mathbf{y}$ and $\mathbf{y}^{\mathcal{D}}$, it means that the object depends on the isogeometric solution $\mathbf{u}_h$ corresponding to the degrees of freedom $\mathbf{y}$ and $\mathbf{y}^{\mathcal{D}}$.

Finally, all partial matrices $\mathbf{K}_T^{ij}(\mathbf{y}, \mathbf{y}^{\mathcal{D}})$ for $i, j \in \mathcal{F}$ are combined into a global matrix $\mathbf{K}_T(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) \in \mathbb{R}^{3|\mathcal{F}|\times 3|\mathcal{F}|}$, called a *tangent stiffness matrix*. Additionally, partial matrices $\mathbf{K}_T^{ij}(\mathbf{y}, \mathbf{y}^{\mathcal{D}})$ for $i \in \mathcal{F}$, $j \in \mathcal{D}$ are combined into an elimination matrix $\mathbf{K}_T^{\mathcal{D}}(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) \in \mathbb{R}^{3|\mathcal{F}|\times 3|\mathcal{D}|}$, and vectors $\mathbf{R}_i(\mathbf{y}, \mathbf{y}^{\mathcal{D}})$, $i \in \mathcal{F}$ are combined into a vector $\mathbf{R}(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) \in \mathbb{R}^{3|\mathcal{F}|}$, called a *residual vector*.

Using the tangent stiffness matrix and the residual, we can formulate a discrete version of Newton's method (3.62): in order to solve a nonlinear equation $\mathbf{R}(\mathbf{y}, \mathbf{y}^{\mathcal{D}}) = \mathbf{0}$, choose an initial guess $\mathbf{y}^0 \in \mathbb{R}^{3|\mathcal{F}|}$. Then for $s \geqslant 0$, compute an update $\Delta\mathbf{y}^{s+1} \in \mathbb{R}^{3|\mathcal{F}|}$ as a solution of the linear system

$$\mathbf{K}_T(\mathbf{y}^s, \mathbf{y}^{\mathcal{D}})\Delta\mathbf{y}^{s+1} = -\mathbf{R}(\mathbf{y}^s, \mathbf{y}^{\mathcal{D}}) - \mathbf{K}_T^{\mathcal{D}}(\mathbf{y}^s, \mathbf{y}^{\mathcal{D}})\Delta\mathbf{y}^{\mathcal{D}}. \tag{3.73}$$

Define $\mathbf{y}^{s+1} = \mathbf{y}^s + \Delta\mathbf{y}^{s+1}$. Continue until convergence. Note that both matrices $\mathbf{K}_T$ and $\mathbf{K}_T^{\mathcal{D}}$ and the residual vector $\mathbf{R}$ have to be reassembled at every iteration of Newton's method.

Initial guess for Newton's method

Newton's method is known to converge quadratically to the solution within a certain area around it. In my experience, provided a good initial guess, it requires around 5–8 iterations to converge with a relative tolerance of 10^{-6}. Of course, providing a good initial guess often is the most complex step when applying Newton's method. If the neo-Hookean material law (3.46) is used, every intermediate solution $\mathbf{u}^s$, including the initial guess $\mathbf{u}^0$, has to be bijective, that is, has to satisfy $J(\mathbf{u}^s) > 0$. Otherwise, the stress tensor $\mathbf{S}(\mathbf{u}^s)$ for the neo-Hookean material law (3.46) can not be evaluated since it contains $\ln J(\mathbf{u}^s)$, and the linear system for (3.62) can not be assembled.

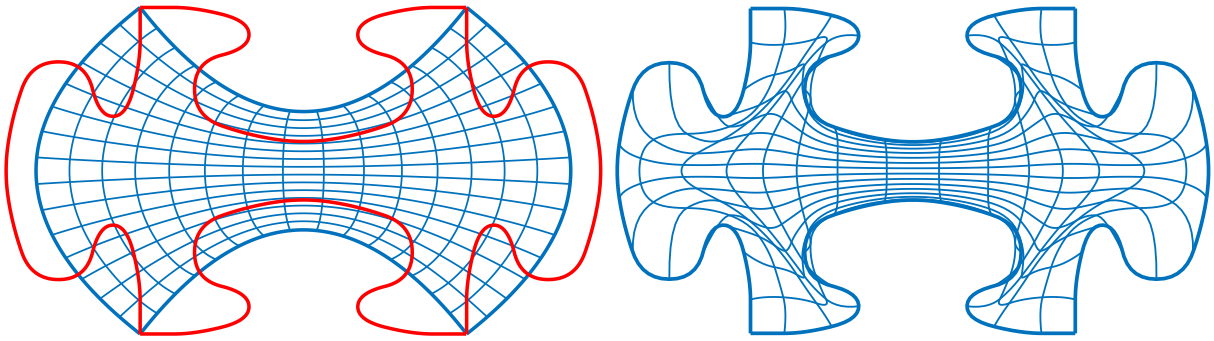

Figure 3.9. Simplified domain Ω_0 and the result of deforming it using linear elasticity (right). Characteristic mesh folds indicate that the linear elasticity system does not produce a bijective displacement field which can be used as an initial guess for Newton's method.

A popular strategy to construct an initial guess is to solve a linear elasticity problem

$$-\operatorname{div}_{\widehat{\mathbf{x}}} \boldsymbol{\sigma}(\mathbf{u}^0) = \mathbf{0} \text{ in } \Omega_0, \tag{3.74}$$

$$\mathbf{u}^0 = \mathbf{u}_{\mathcal{D}} \text{ on } \partial\Omega_0. \tag{3.75}$$

A weak form of (3.74–3.75) reads

$$\text{find } \mathbf{u}^0 \in \mathcal{V} \text{ such that } \int_{\Omega_0} \boldsymbol{\sigma}(\mathbf{u}^0) : \boldsymbol{\varepsilon}(\mathbf{w}) d\widehat{\mathbf{x}} = 0 \quad \forall \mathbf{w} \in \mathcal{V}_0. \tag{3.76}$$

In order to discretize the weak form (3.76), the same method as for the nonlinear elasticity can be conveniently applied. Concretely, the corresponding linear system can be obtained by assembling

linear system (3.73) with the St. Venant-Kirchhoff material law and the initial solution vector $\mathbf{y}^0 = \mathbf{0}$. Unfortunately, if the prescribed displacement $\mathbf{u}_\mathcal{D}$ is large, the solution of (3.74–3.74) may be not bijective like it happens for the puzzle piece example, see Figure 3.9. To overcome this problem, the incremental loading approach described in the next section can be used.

3.3.5 Incremental deformation

If the prescribed boundary displacement $\mathbf{u}_\mathcal{D}$ is too large to handle in one step, the *incremental loading* approach can be applied. It consists in replacing problem (3.54–3.55) by a sequence of problems

$$-\operatorname{div}_{\widehat{\mathbf{x}}} \mathbf{P}(\mathbf{u}_i) = \mathbf{0} \text{ in } \Omega_0, \tag{3.77}$$

$$\mathbf{u}_i = \frac{i}{N}\mathbf{u}_\mathcal{D} \text{ on } \partial\Omega_0 \tag{3.78}$$

for a range of *loading steps* $i = 1, \ldots, N$. At each step, the problem is solved using Newton's method. *Incremental solution* $\mathbf{u}_i$ at the i-th loading step provides an initial guess for Newton's method at the next step. Concretely, let $\mathcal{V}_i = \{\mathbf{w} \in H^1(\Omega_0)^3 \mid \mathbf{w} = \frac{i}{N}\mathbf{u}_\mathcal{D} \text{ on } \partial\Omega_0\}$, $i = 1, \ldots, N$ be a sequence of trial solution spaces for different loading steps. At the i-th step, iterations of Newton's method $\mathbf{u}_i^s \in \mathcal{V}_i \to \mathbf{u}_i^{s+1} \in \mathcal{V}_i$ advance the solution within $\mathcal{V}_i$. An increment $\Delta\mathbf{u}_i^{s+1} \in \mathcal{V}_0$ such that $\mathbf{u}_i^{s+1} = \mathbf{u}_i^s + \Delta\mathbf{u}_i^{s+1}$ is found as a solution to the following weak problem:

$$\text{find } \Delta\mathbf{u}_i^{s+1} \in \mathcal{V}_0 \text{ such that } \; DR(\mathbf{u}_i^s, \mathbf{w}) \cdot \Delta\mathbf{u}_i^{s+1} = -R(\mathbf{u}_i^s, \mathbf{w}) \;\; \forall \mathbf{w} \in \mathcal{V}_0. \tag{3.79}$$

Loading level i determines what vector of Dirichlet degrees of freedom $\mathbf{y}^\mathcal{D}$ is used when assembling the linear system (3.73). I call this a *type-A update*, or simply a *Newton iteration*. Type-A updates continue until the convergence criteria are met, and the last intermediate solution $\mathbf{u}_i^s$ is accepted as the incremental solution $\mathbf{u}_i$ for (3.77–3.78) at the step i.

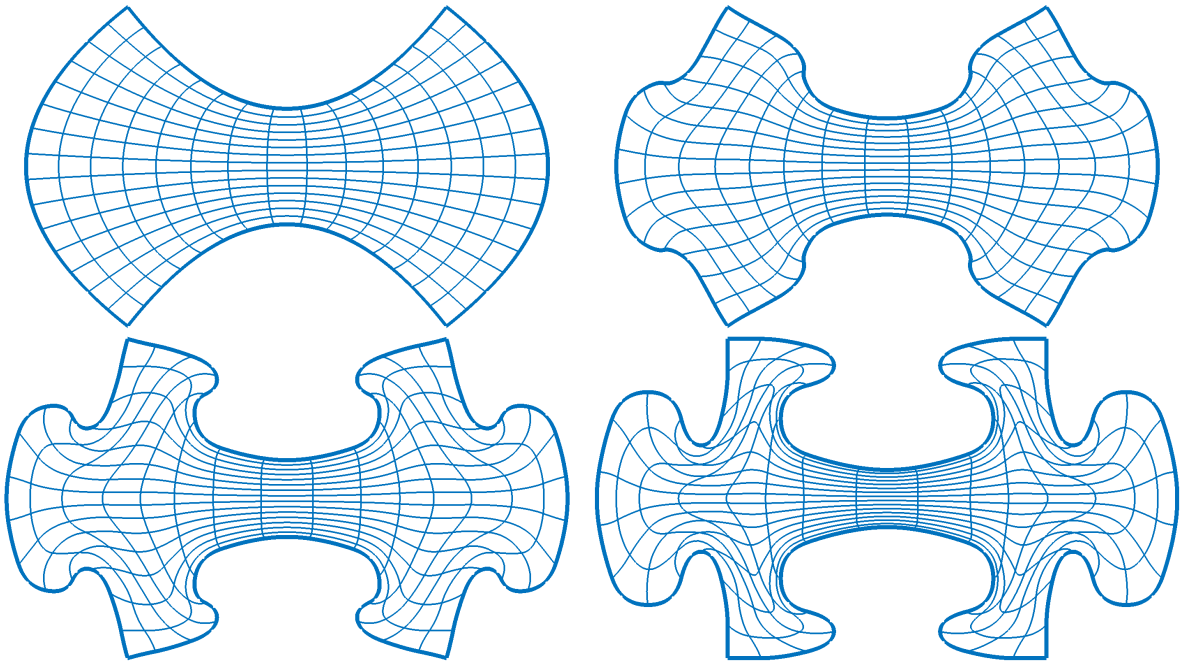

Figure 3.10. Incremental loading approach with $N = 6$ loading steps. Reference configuration (top-left) and deformed configurations corresponding to $\mathbf{u}_i$, $i = 2$ (top-right), 4 (bottom-left) and 6 (bottom-right).

Incremental solution $\mathbf{u}_i$ belongs to $\mathcal{V}_i$, whereas an initial guess $\mathbf{u}^0_{i+1}$ for Newton's method at the step $i+1$ should belong to $\mathcal{V}_{i+1}$. In order to construct it, I use a *type-B update* $\mathbf{u}_i \in \mathcal{V}_i \to \mathbf{u}^0_{i+1} \in \mathcal{V}_{i+1}$ which advances the solution from $\mathcal{V}_i$ to $\mathcal{V}_{i+1}$, that is, to the next trial solution space. I also refer to a type-B update as a *diagonal Newton step* (DNS). An increment $\Delta\mathbf{u}_{i+1} \in \mathcal{V}_1$ such that $\mathbf{u}^0_{i+1} = \mathbf{u}_i + \Delta\mathbf{u}_{i+1}$ is found as a solution to the weak problem:

$$\text{find } \Delta\mathbf{u}_{i+1} \in \mathcal{V}_1 \text{ such that } \; DR(\mathbf{u}_i, \mathbf{w}) \cdot \Delta\mathbf{u}_{i+1} = -R(\mathbf{u}_i, \mathbf{w}) \;\; \forall \mathbf{w} \in \mathcal{V}_0. \tag{3.80}$$

The difference between Newton iterations and DNS updates is that the DNS increments are not zero on the boundary. Therefore, the corresponding vector of Dirichlet degrees of freedom $\Delta\mathbf{y}^{\mathcal{D}}$ in (3.73) is not zero and is accounted for by the elimination matrix $\mathbf{K}^{\mathcal{D}}_T$. I say that the DNS increment $\Delta\mathbf{u}_i$ has a *stepsize* of $1/N$ meaning that $\Delta\mathbf{u}_i$ advances the displacement on the boundary $\partial\Omega_0$ by a $1/N$-th part of $\mathbf{u}_{\mathcal{D}}$. The incremental loading approach is initialized with an initial displacement $\mathbf{u}_0 = \mathbf{0}$, and the incremental solution $\mathbf{u}_N$ at the last step is accepted as the solution to the original system (3.54–3.55). An example of the incremental loading approach with $N = 6$ loading steps is presented in Figure 3.10. The overall procedure is illustrated schematically in Figure 3.11. I would like to point out that the incremental loading approach for nonlinear elasticity problems can be interpreted in terms of the *continuation methods* theory for general nonlinear problems [70], also known as the *numerical path following*. In particular, the type-B update is an example of applying the so-called *tangential continuation*. Alternatively, incremental loading can be seen as *pseudo-time stepping*.

TINE: tangential incremental nonlinear elasticity

I refer to the described above algorithm as *full incremental loading*. Its disadvantage is its high computational cost. If S is an average number of iterations that Newton's method requires to converge at each loading step, then the algorithm requires NS iterations to compute the solution $\mathbf{u}$, each involving solution of a linear system. This is justified in applications where the deformation history is of interest; in the considered case, however, only the final displacement $\mathbf{u} = \mathbf{u}^N$ is important. Moreover, first $N-1$ loading steps together can be considered as just an elaborate way to construct an initial guess $\mathbf{u}^0_N$ for the last loading step, that is, for the original Newton's method for the problem (3.54–3.55).

Here, I propose an alternative way to construct the initial guess $\mathbf{u}^0_N$ which is a shortcut to full incremental loading. I call it *tangential incremental nonlinear elasticity* (TINE). It consists of N diagonal Newton steps $\mathbf{u}^{\text{TINE}}_i \in \mathcal{V}_i \to \mathbf{u}^{\text{TINE}}_{i+1} \in \mathcal{V}_{i+1}$ which quickly take the solution right to the last loading level. The resulting displacement $\mathbf{u}^{\text{TINE}}_N \in \mathcal{V}_N$ is then used as an initial guess for Newton's method at the last loading step. TINE with Newton's method allows to compute the solution $\mathbf{u}$ in only $N + S$ iterations. The difference between the proposed algorithm and full incremental loading is illustrated in Figure 3.11.

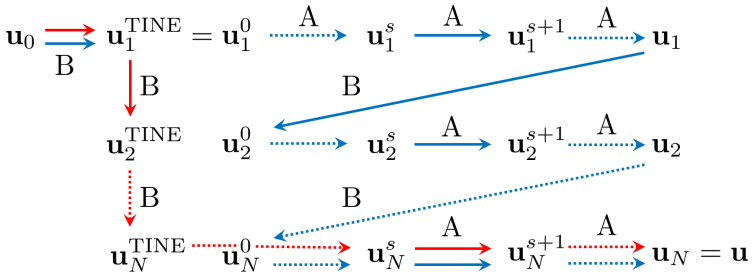

Figure 3.11. Full incremental loading (blue) and Newton's method with TINE (red).

Even more computationally efficient is to use the TINE method as a stand-alone deformation algorithm, which is justified for several reasons. First, the displacement $\mathbf{u}_N^{\text{TINE}}$ is close enough to the solution $\mathbf{u}$ so that the final application of Newton's method results only in rather small corrections. More precisely, my numerical experiments show that $\mathbf{u}_N^{\text{TINE}}$ converges to $\mathbf{u}$ quadratically as the number of incremental steps N grows, namely $||\mathbf{u} - \mathbf{u}_N^{\text{TINE}}||_{L^2} = O(N^{-2})$, see Figure 3.12. Second, solving the system (3.54–3.55) was never a goal in itself. Instead, it is just one of the ways to define a deformation mapping $\mathbf{D}$ to use in the mesh deformation method for domain parametrization. To that end, any bijective displacement field of good quality can be used. So if $\mathbf{u}_N^{\text{TINE}}$ is close to $\mathbf{u}$, that is good enough. In my experiments, it suffices to use from 5 to 10 loading steps to ensure the bijectivity of $\mathbf{u}_N^{\text{TINE}}$. At this stage, $\mathbf{u}_N^{\text{TINE}}$ is often visually indistinguishable from $\mathbf{u}$.

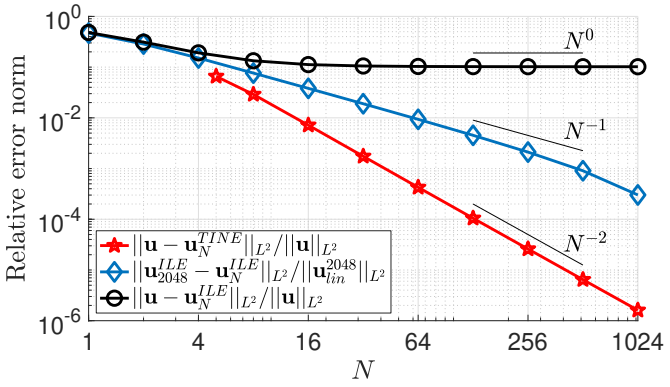

Figure 3.12. Convergence analysis for the TINE and ILE mesh deformation methods.

Bijectivity and adaptivity

Although the neo-Hookean material law (3.46) guarantees that the solution $\mathbf{u}$ of the system (3.54–3.55) is bijective, special care is required to achieve this property in practice. The directional derivative $DR(\mathbf{u}^*, \mathbf{w}) \cdot \Delta\mathbf{u}$ (3.59) can only be evaluated for a bijective displacement $\mathbf{u}^*$. However, both Newton iterations (3.79) and diagonal Newton steps (3.80) can produce an increment $\Delta\mathbf{u}$ such that $\mathbf{u}^* + \Delta\mathbf{u}$ is not bijective. The problem can be overcome by adaptively scaling $\Delta\mathbf{u}$. If $J(\mathbf{u}^*) > 0$, due to the continuity of J there exists a scaling coefficient $t \in [0, 1]$ such that

$$J(\mathbf{u}^* + t\Delta\mathbf{u}) > 0. \tag{3.81}$$

In practice, I determine t by consecutively testing values $t^k = 1/2^k$ until (3.81) is satisfied.

The implementation of adaptivity differs slightly for Newton iterations and DNS. For a Newton iteration, the increment is determined solely by the weak problem (3.79) and can be scaled freely; after that, the algorithm can proceed to the next Newton iteration. On the other hand, the step size h_i of the DNS increment $\Delta\mathbf{u}^i$ is predefined by the number of incremental steps N. Scaling the DNS increment changes the step size to th_i and−since all updates of the boundary displacement have to add up to $\mathbf{u}_\mathcal{D}$−requires changing the subsequent step sizes. One way to do it is to proceed with step sizes of $1/N$, scaling it to fulfill (3.81) if necessary. The final step size

$$h_{N^*} = 1 - \sum_{i=1}^{N^*-1} h_i \tag{3.82}$$

makes sure that all step sizes add up to 1.

Another possibility is to apply a greedy step size strategy where the algorithm starts with the first DNS step size h_1 equal to 1. If the resulting displacement is not bijective, h_1 is iteratively halved until (3.81) is satisfied. The method proceeds with the step sizes

$$h_i = 1 - \sum_{j=1}^{i-1} h_j \tag{3.83}$$

which are also iteratively halved if necessary.

Unfortunately, both described adaptive strategies suffer from two drawbacks. The first is stalling, which means that the adaptive strategies sometimes tend to produce too small step sizes by iterative halving. The second drawback is the instability caused by the logarithm in the neo-Hookean material law (3.46). If $J(\mathbf{u}^*)$ is positive but too close to zero, which often happens in an attempt to satisfy (3.81), $S(\mathbf{u}^*)$ becomes large, and the corresponding directional derivative (3.59) "pushes" the solution too strongly away from the boundary $J(\mathbf{u}) = 0$ of the feasible region. The solution is then likely to cross the boundary of the feasible region at another point, and the whole procedure repeats. This behavior is similar to the *interior-point methods* used in constrained optimization [65].

In contrast to that, the non-adaptive strategy with uniform step sizes $1/N$ seems to be very robust. If N is large enough, neither stalling nor unstable behavior occurs. In light of that, my preferred "adaptive" strategy is to double N consecutively until the bijective displacement is successfully computed.

Lastly, I would like to remark on ways to test the bijectivity condition $J(\mathbf{u}) > 0$. A solution which takes into account the NURBS nature of the isogeometric solution $\mathbf{u}_h$ is to express $J(\mathbf{u}_h)$ as a NURBS function [25]. If all coefficients in a NURBS expansion of $J(\mathbf{u}_h)$ are positive, then the displacement $\mathbf{u}_h$ is bijective. Unfortunately, this condition is only sufficient and not a necessary one. This may often lead to a false detection of the bijectivity violation. In practice, I resort to a much less elegant solution of sampling $J(\mathbf{u}_h)$ at the Gaussian quadrature points associated with $\mathbf{u}_h$.

ILE: incremental linear elasticity

I have introduced the TINE method as a simplification of the full incremental loading approach to solution of the nonlinear problem (3.54–3.55). However, TINE is in its essence very similar to another mesh deformation technique which has been successfully applied in many fluid-structure interaction (FSI) applications [11, 34, 36]. Often referred to as a mesh update strategy based on linear elasticity, it involves solving a linear elasticity problem formulated in the last known configuration of the fluid domain with the Dirichlet boundary condition prescribed by the motion of the fluid-structure interface. The resulting displacement field is used to update the computational mesh in the domain occupied by the fluid to the next time step. I call this technique *incremental linear elasticity* (ILE).

When applied to domain parametrization, ILE takes the following form. At each loading step $i = 1, \ldots, N$, a displacement increment $\Delta\mathbf{u}_i^{\mathrm{ILE}} : \Omega_{i-1} \to \mathbb{R}^3$ is computed by solving the following linear system:

$$-\operatorname{div}_{\mathbf{x}} \boldsymbol{\sigma}(\Delta\mathbf{u}_i^{\mathrm{ILE}}) = \mathbf{0} \text{ in } \Omega_{i-1}, \tag{3.84}$$

$$\Delta\mathbf{u}_i^{\mathrm{ILE}} = \frac{1}{N}\mathbf{u}_{\mathcal{D}} \text{ on } \partial\Omega_{i-1}. \tag{3.85}$$

Here, domain Ω_{i-1} is a deformed configuration of the initial simplified domain Ω_0. Once $\Delta\mathbf{u}_i^{\mathrm{ILE}}$ is found, the next deformed configuration Ω_i is defined as $\Omega_{i-1} + \Delta\mathbf{u}_{lin}^i$. At $i = 1$, the procedure

is initialized with a simplified domain Ω_0. After i steps, the displacement field $\mathbf{u}_i^{\mathrm{ILE}} : \Omega_0 \to \mathbb{R}^3$ is defined as a sum of all already computed increments:

$$\mathbf{u}_i^{\mathrm{ILE}} = \sum_{j=1}^{i} \Delta\mathbf{u}_j^{\mathrm{ILE}}. \tag{3.86}$$

Note that when assembling a system of linear algebraic equations for problem (3.84–3.85), NURBS basis functions N_j are defined for $\mathbf{x} \in \Omega_{i-1}$ as $N_j(\mathbf{x}) = N_j(\mathbf{G}_{i-1}^{-1}(\mathbf{x}))$. The parametrization $\mathbf{G}_{i-1}$ of the deformed configuration Ω_{i-1} is constructed as $\mathbf{D}(\mathbf{u}_{i-1}^{\mathrm{ILE}}) \circ \mathbf{G}_0$. The final displacement field $\mathbf{u}_N^{\mathrm{ILE}}$ is used to construct the parametrization $\mathbf{G}$ of the target domain Ω.

To the best of my knowledge, none of the published research where the ILE method is described provides an explanation for why the resulting displacement $\mathbf{u}_N^{\mathrm{ILE}}$ is bijective. Nevertheless, it is empirically confirmed by various FSI applications where the method has been successfully applied. Figure 3.13 shows the result of applying ILE to the puzzle piece example. I compare it against the baseline parametrization constructed using TINE. Note that the two parametrization are surprisingly similar. I say that this similarity is surprising because the linear elasticity model uses the Hooke's law (3.52) which is much closer related to the St.Venant-Kirchhoff law (3.45) than to the neo-Hookean law (3.46). Thus, one could expect that the result of the ILE mesh deformation resembles the solution of the nonlinear system (3.54–3.55) with the St. Venant-Kirchhoff material law depicted in Figure 3.14. However, this is not the case. In particular, the St. Venant-Kirchhoff material law results in a not bijective displacement field. Despite the similarity of the ILE and TINE results, they are not the same. Concretely, the displacement field $\mathbf{u}_N^{\mathrm{ILE}}$ does not converge to the solution $\mathbf{u}$ of the system (3.54–3.55) as the number of steps N grows. However, $\mathbf{u}_N^{\mathrm{ILE}}$ seems to converge linearly to a certain limiting bijective displacement field $\mathbf{u}^{\mathrm{ILE}}$, that is, $||\mathbf{u}^{\mathrm{ILE}} - \mathbf{u}_N^{\mathrm{ILE}}||_{L^2} \sim O(N^{-1})$, see Figure 3.12. Therefore, just like TINE, ILE can be used as a stand-alone mesh deformation technique suitable for constructing domain parametrizations.

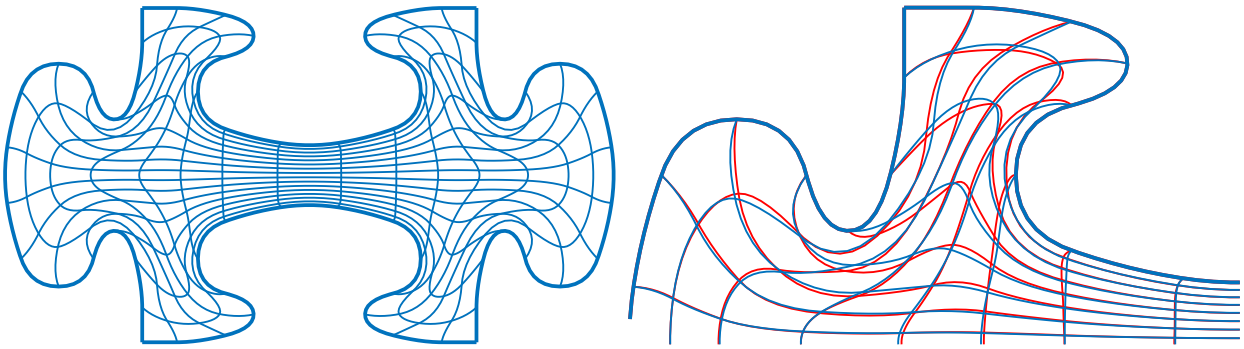

Figure 3.13. Comparison of parametrizations constructed using incremental linear elasticity (ILE, blue) and tangential incremental nonlinear elasticity (TINE, red).

To sum up, ILE is an at least empirically robust mesh deformation strategy which is very attractive due its simplicity. It goes without saying that implementation of a linear elasticity solver poses a far less formidable challenge than implementation of a nonlinear elasticity solver. In comparison to TINE, ILE requires a bit more incremental steps to compute a displacement field suitable for the domain parametrization. However, this is compensated by the fact that a linear elasticity system is faster to assemble than a linearized nonlinear elasticity system, so the total computational effort is roughly the same.

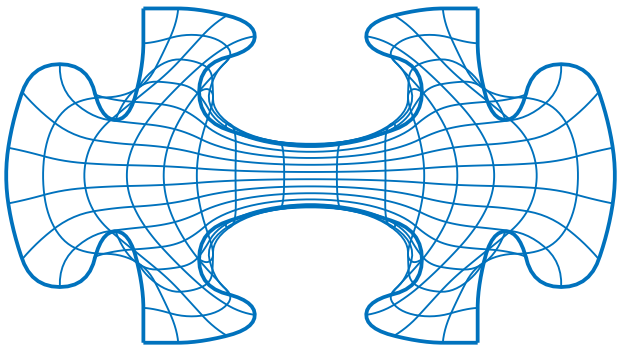

Figure 3.14. Result of solving the nonlinear system (3.54–3.55) with the St. Venant-Kirchhoff material law. The corresponding parametrization of the target domain Ω is not bijective.

Material parameters

So far, I have not focused on the material parameters. With the material laws considered in this work, the material behavior can be characterized by two constants. One popular choice is the pair consisting of Young's modulus and Poisson's ratio. Although the role of each of these parameters can be better understood by studying the material laws where they appear, I can vaguely say that Young's modulus defines the overall stiffness of the material whereas Poisson's ratio defines the resistance of the material to volumetric changes. Physically meaningful values for Poisson's ratio lie in the interval $[0, 0.5]$, where 0 corresponds to no resistance to volumetric changes, and 0.5 corresponds to an absolutely incompressible material.

In the considered problems (3.54–3.55) and (3.84–3.85), no volumetric or surface force is applied, so Young's modulus does not affect the resulting displacement field. On the other hand, Poisson's ratio proves to be of great importance. A material with high Poisson's ratio resists volumetric changes which means that $J(\mathbf{u})$ remains close to 1. This means that if the parametrization $\mathbf{G}_0$ of the simplified domain Ω_0 is bijective, then it is likely to remain bijective after the deformation $\mathbf{D}$. As a result, high values of Poisson's ratio "increase" the bijectivity of the parametrization $\mathbf{G}$ of the target domain Ω. Another benefit of high Poisson's ratio is the preservation of the parametrization uniformity. If the computational mesh corresponding to the parametrization $\mathbf{G}_0$ is uniform, it remains so also after the deformation, provided that Poisson's ratio is high enough, see Figure 3.15. In practice, I have successfully used values for Poisson's ratio between 0.45 and 0.49. Values higher than 0.49 lead to an ill-conditioned numerical system unless a special displacement-pressure formulation for (nearly) incompressible behavior is used. Although a fully incompressible material may seem advantageous, it would require Ω and Ω_0 to be of the same volume. This condition is hard to achieve in practice.

Poisson's ratio also has an effect on the minimal number of incremental steps N_{min} necessary to compute a bijective displacement field when the neo-Hookean material law is used. As Poisson's ratio approaches 0.5, N_{min} tends to grow in a hyperbolic fashion, see Figure 3.16. To compute a bijective displacement field for the puzzle piece example, the TINE mesh deformation strategy requires at least 10 steps for Poisson's ratio of 0.49. If Newton's method is used afterwards, 15 steps are required. Such behavior is not observed for the ILE technique because it does not use the logarithmic neo-Hookean material law (3.46). Just two steps are already enough to compute a bijective displacement field. However, this field may not be of high quality, meaning that it may have highly non-uniformly sized elements, see Figure 3.16. Therefore a larger number of incremental steps is advisable.

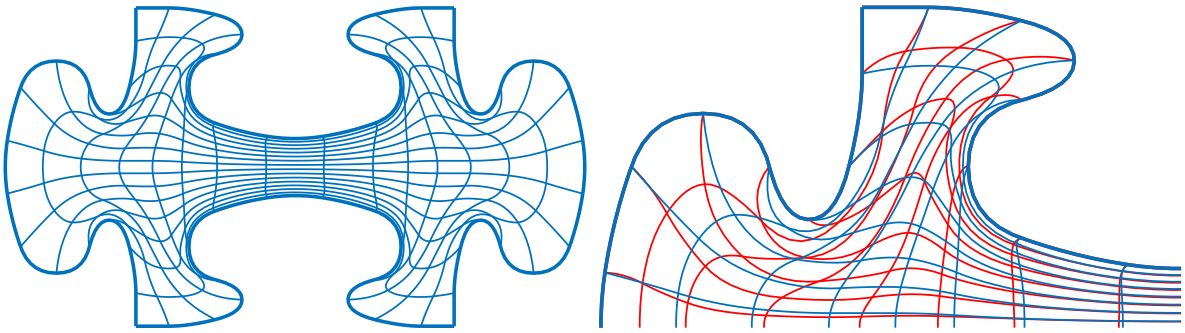

Figure 3.15. Left: result of the TINE-based mesh deformation with Poisson's ratio of 0.2. Right: comparison with the baseline parametrization (red) computed with Poisson's ratio of 0.45. Elements of the baseline parametrization are more uniformly sized.

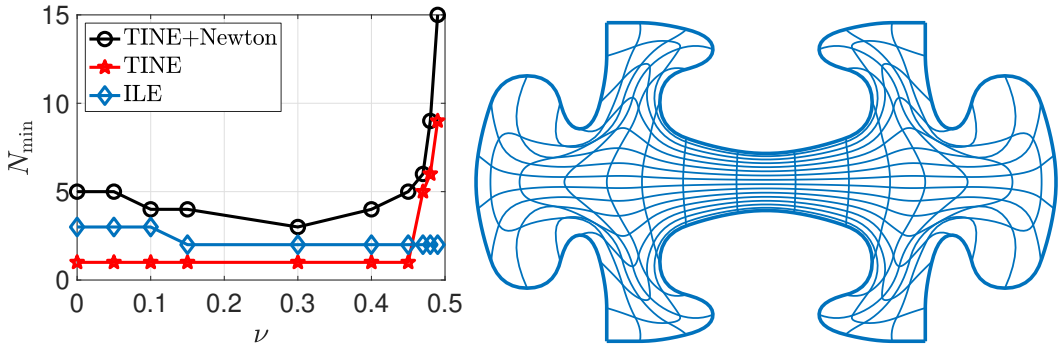

Figure 3.16. Left: dependence of the minimum number of incremental steps N_{min} on Poisson's value ν. Right: result of the ILE-based mesh deformation with $N = 2$ and $\nu = 0.49$.

Although several other combinations of material parameters can be used to describe the material behavior, I find the pair of Young's modulus and Poisson's ratio particularly suitable for this application since it allows to control the mesh deformation using only one parameter, namely Poisson's ratio, which reduces the complexity of the method. Moreover, this single parameter defines the resistance to the bijectivity violation—a crucial property for any parametrization technique.

3.4 Examples and comparison

In this section, I demonstrate the performance of the mesh deformation approach to the single-patch domain parametrization problem. First, I consider a couple of two-dimensional single-patch examples and compare the mesh deformation approach with the state-of-the-art techniques. Generally, a fair comparison is hard to conduct because it is unclear which quality measure to use for comparison. Many techniques seek to optimize a parametrization with respect to a certain quality measure, like the optimization-based methods considered in Section 3.2.3. In this case, it would be unfair to use any of these measures for comparison because the corresponding optimal parametrization would certainly be preferred. Another popular method to compare different parametrizations is to use them to solve a given PDE and to judge about the parametrization quality by the accuracy of the solution, like I do in Section 2.3.4. This method is not flawless either since the PDE can be tailored to be "easier" for some parametrization, for example by placing a particularly hard to resolve region of the solution in an area with small-sized elements.

In light of that, I have opted for the global ratio of the Jacobian determinant

$$JR(\mathbf{G}) = \frac{\max\limits_{\boldsymbol{\xi}\in[0,1]^2} J(\mathbf{G}(\boldsymbol{\xi}))}{\min\limits_{\boldsymbol{\xi}\in[0,1]^2} J(\mathbf{G}(\boldsymbol{\xi}))} \tag{3.87}$$

as a rather impartial measure which is not, at least explicitly, optimized by any of the parametrization technique. $JR(\mathbf{G})$ measures uniformity of the parametrization and reacts to violation of bijectivity.

After the comparison, I briefly consider a range of multi-patch and 3D examples to show that the mesh deformation approach is not limited to a 2D single-patch scenario. The implementation of the mesh deformation approach is based on the G+Smo library [60] and an additionally developed module with linear and nonlinear elasticity solvers.

2D male rotor

The first example is a profile of a twin screw compressor's male rotor, studied in [71]. This is a deceptively simple geometry with a clear square-shaped structure, so most of the parametrizations techniques can be expected to perform well. Each side is given by a quadratic NURBS curve with 26 control points. Apart from the mesh deformation approach, I have tested the Coons patch, the spring model, the optimization-based approach with the AO quality measure and the elliptic grid generation technique.

For the mesh deformation approach, I have constructed two simplified domains using coarse NUBRS bases with parameters $(q = 1, m = 2)$ and $(q = 3, m = 5)$. I refer to the resulting simplified domains as the linear and the cubic. The mesh deformation step has been performed using Newton's method starting from an initial guess provided by the TINE method (I refer to this as the *TINE+Newton's* method). For Poisson's ratio, I have chosen a value a value $\nu = 0.49$.

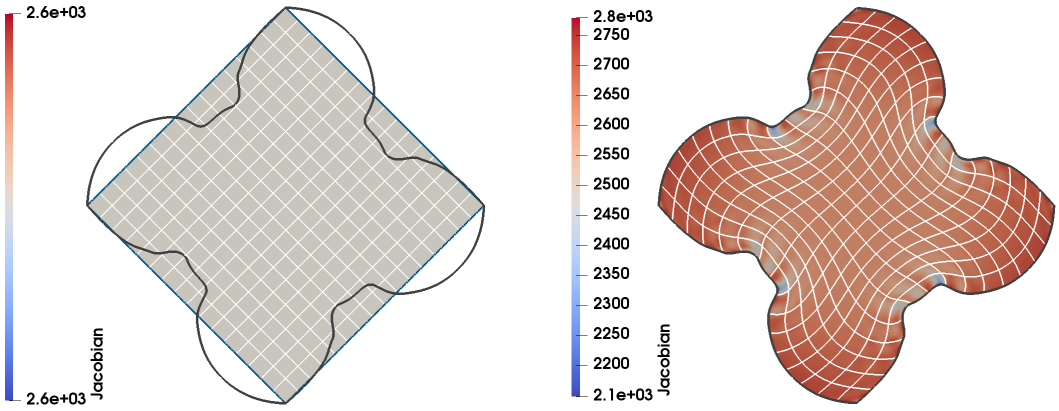

Figure 3.17. 2D male rotor: mesh deformation approach (TINE+Newton's method, N=5, ν=0.49, simplified domain $q = 1$, $m = 2$).

The results are presented in Figures 3.17–3.20. The Coons patch has failed to construct a bijective parametrization. On the other hand, the output of the spring model is a bijective but highly non-uniform parametrization with $JR(\mathbf{G}) = 58.12$. The AO-optimization and EGG perform well and produce high quality parametrizations with $JR(\mathbf{G})$ equal 3.4 and and 5.83, correspondingly. The most uniform parametrizations are produced by the mesh deformation approach with $JR(\mathbf{G}) = 1.33$ for the linear simplified domain and $JR(\mathbf{G}) = 2.46$ for the cubic one. The former required

$N = 5$ and the latter $N = 3$ loading steps. The difference can be explained by the fact that the cubic simplified domain is geometrically closer to the original. Still, the former parametrization is more uniform, probably because it has inherited this property from the simplified domain due to the high value of Poisson's ratio.

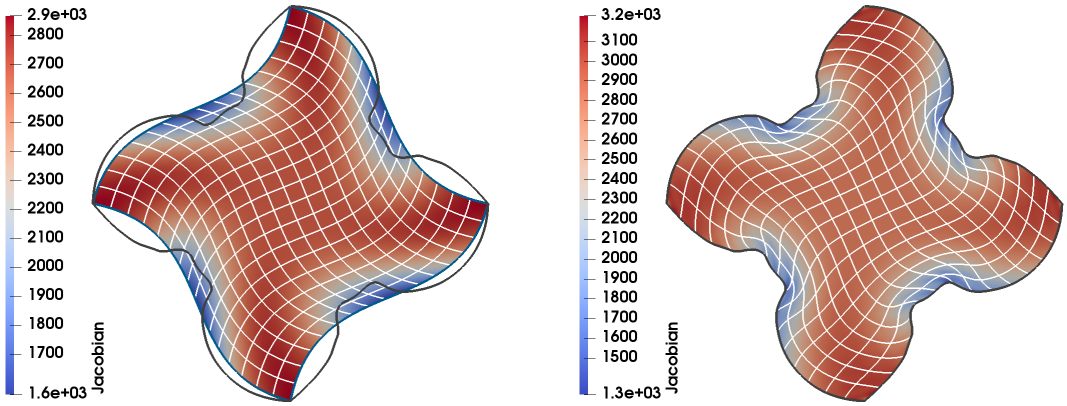

Figure 3.18. 2D male rotor: mesh deformation approach (TINE+Newton's method, N=3, ν=0.49, simplified domain $q = 3$, $m = 4$).

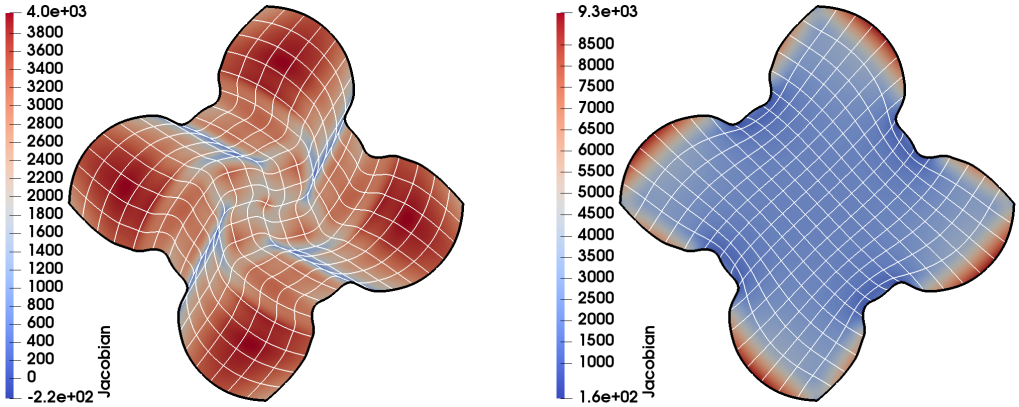

Figure 3.19. 2D male rotor: Coons patch (left) and spring model (right).

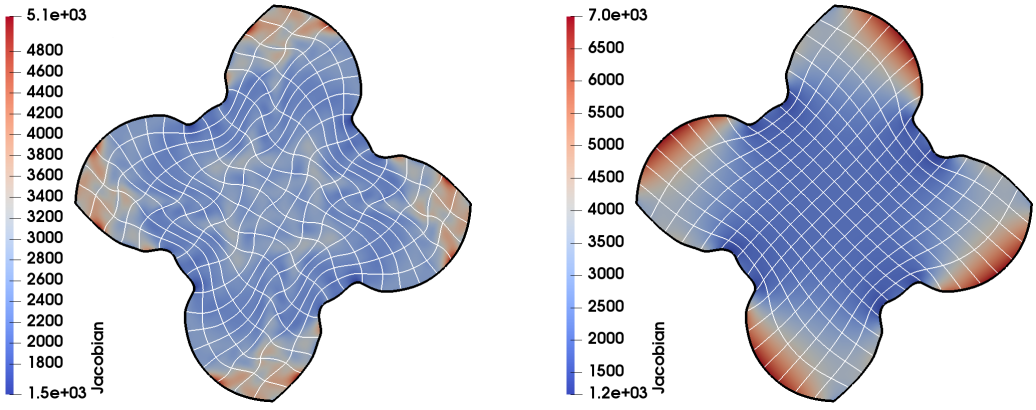

Figure 3.20. 2D male rotor: AO-optimization (left) and EGG (right).

2D puzzle piece

The second example is the already familiar to the reader puzzle piece. It is a fairly complex geometry which, arguably, lies at the very edge of the class of domains for which a single-patch parametrization should be attempted. Its boundary is given by a quadratic NURBS curve, each side having 33 control points. The mesh deformation approach is compared only against the AO-based optimization approach and the EGG technique.

Similarly to the 2D male rotor example, I have constructed two simplified domains using coarse NURBS bases with parameters $(q = 1, m = 2)$ and $(q = 2, m = 3)$. I refer to them as the linear domain and the quadratic. The mesh deformation step has been performed using the TINE+Newton's method with Poisson's ratio of $\nu = 0.49$.

The results are presented in Figures 3.21–3.23. The EGG technique has produced a barely bijective parametrization with $JR(\mathbf{G}) = 395.83$. The AO-optimization has resulted in a good-quality parametrization with $JR(\mathbf{G}) = 8.46$. The most uniform parametrizations are again produced by the mesh deformation approach. The linear simplified domain has required $N = 13$ loading steps and the resulting parametrization has the global ratio of the Jacobian determinant $JR(\mathbf{G}) = 3.86$. For the quadratic simplified domain, the numbers are $N = 8$ and $JR(\mathbf{G}) = 5.85$. Judging solely by the numbers, the parametrization produced from the linear simplified domain is superior. However, the quadratic simplified domain is much closer to the target shape and had to be deformed less. The corresponding parametrization thus appears more visually attractive.

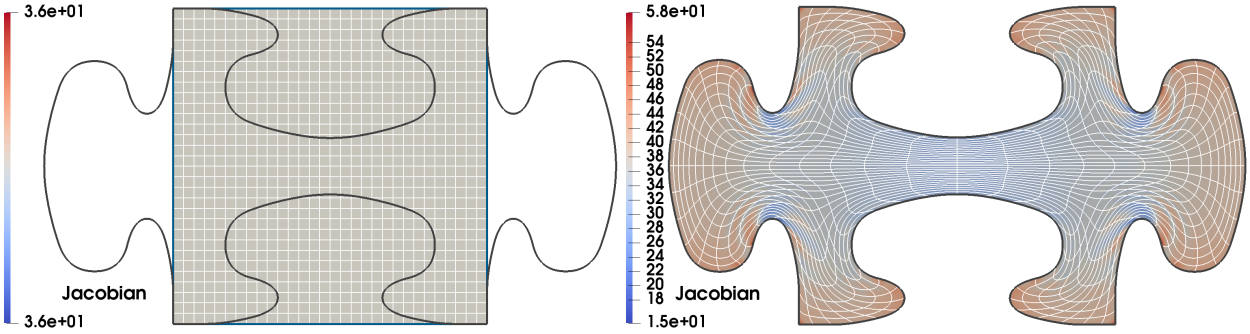

Figure 3.21. 2D puzzle piece: mesh deformation approach (TINE+Newton's method, N=13, ν=0.49, simplified domain $q = 1$, $m = 2$).

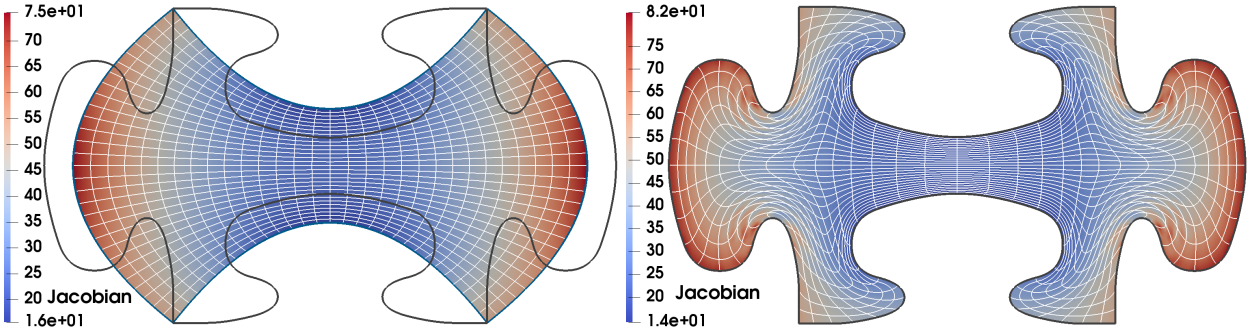

Figure 3.22. 2D puzzle piece: mesh deformation approach (TINE+Newton's method, N=10, ν=0.49, simplified domain $q = 2$, $m = 3$ parametrized using the mesh deformation approach (TINE+Newton's method, $N = 5$, $\nu = 0.49$, simplified domain $q = 1$, $m = 2$)).

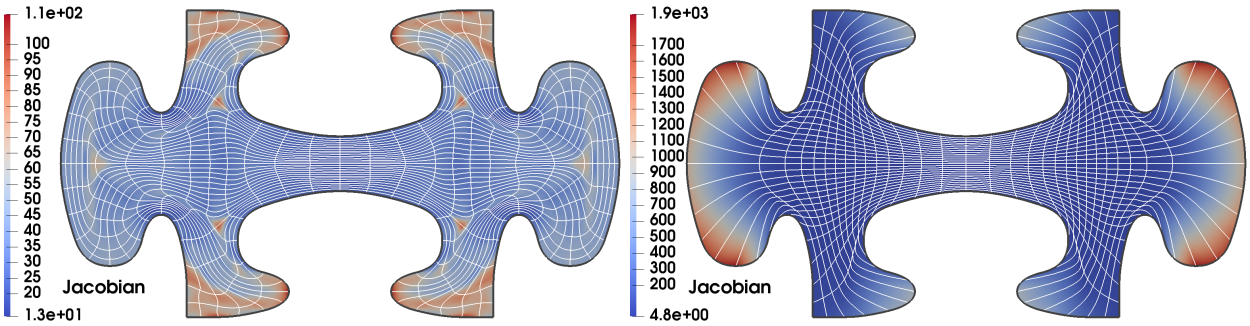

Figure 3.23. 2D puzzle piece: AO-optimization (left) and EGG (right).

2D multi-patch female rotor

The next example is a 2D profile of the twin screw compressor's female rotor, a counterpart to the male rotor presented above. Although a single-patch parametrization of this domain is possible (for example, using the mesh deformation approach), I would not recommend it because the domain's boundary has no corners. As a result, any single-patch parametrization would have four degenerated corners, see Figure 3.24. As I show in Section 2.3.4, parametrizations with degenerated elements are not desirable in numerical simulations.

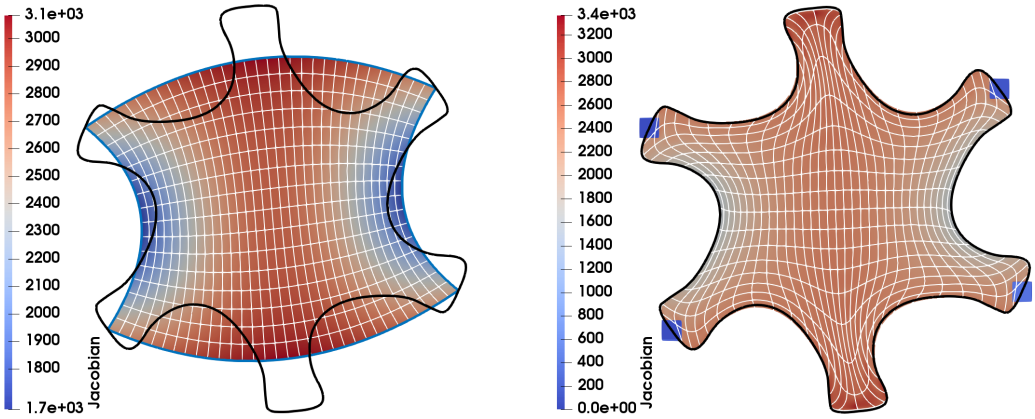

Figure 3.24. 2D female rotor: single-patch parametrization constructed using the mesh deformation approach (TINE+Newton's method, N=8, ν=0.48, simplified domain $q = 2$, $m = 3$).

A better choice would be to construct a multi-patch parametrization. In order to make use of the domain's symmetries, I have chosen a multi-patch structure with two central hub patches and six symmetric patches around them, see Figure 3.25. To apply the mesh deformation approach, I have constructed a simplified domain as a union of the corner-quads corresponding to each patch. The mesh deformation has been performed using the TINE+Newton's method with Poisson's ratio of 0.48 and $N = 5$ incremental loading steps. Note that instead of deforming each patch separately, I have deformed the entire simplified domain as a single elastic body. This has caused the inter-patch interfaces to assume more natural shapes dictated by the boundary. Such a simultaneous multi-patch parametrization technique is advantageous in comparison to those parametrization techniques that can only construct parametrizations for each patch separately, like the Coons patch or EGG.

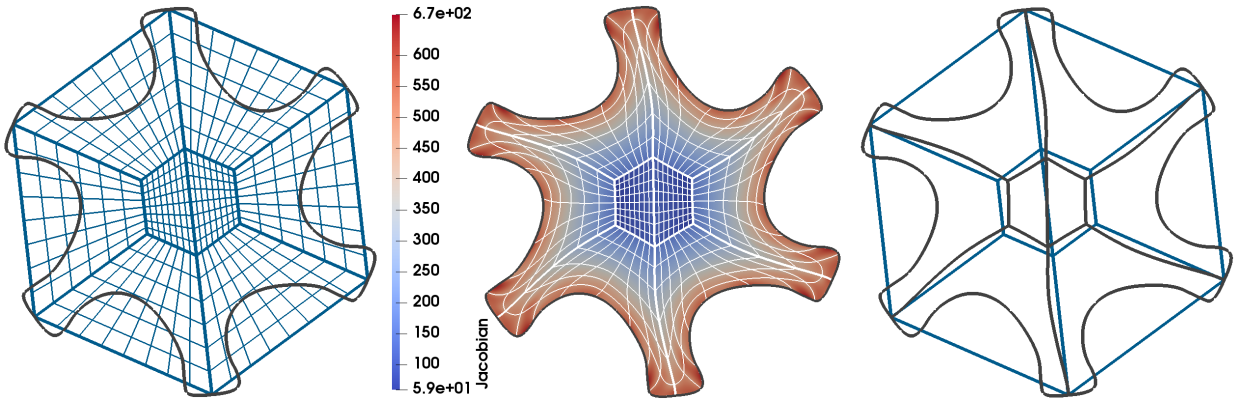

Figure 3.25. 2D female rotor: multi-patch parametrization constructed using the mesh deformation approach (TINE+Newton's method, $N = 5$, $\nu = 0.48$). Simplified domain (left), result of mesh deformation (middle), comparison of the multi-patch structure for the simplified and the resulting parametrizations (right).

3D puzzle piece

Next, I present a three-dimensional single-patch example, see Figure 3.26. With the six compatible boundary surfaces available, the mesh deformation approach can be applied in the same way as in the two-dimensional case. The simplified domain has been constructed by projection on a coarsest quadratic NURBS basis. After that, the deformation step has been performed using the TINE+Newton's method with Poisson's ratio of $\nu = 0.46$ and $N = 10$ loading steps. The cross-section presented in Figure 3.26 shows the internal structure of the resulting parametrization.

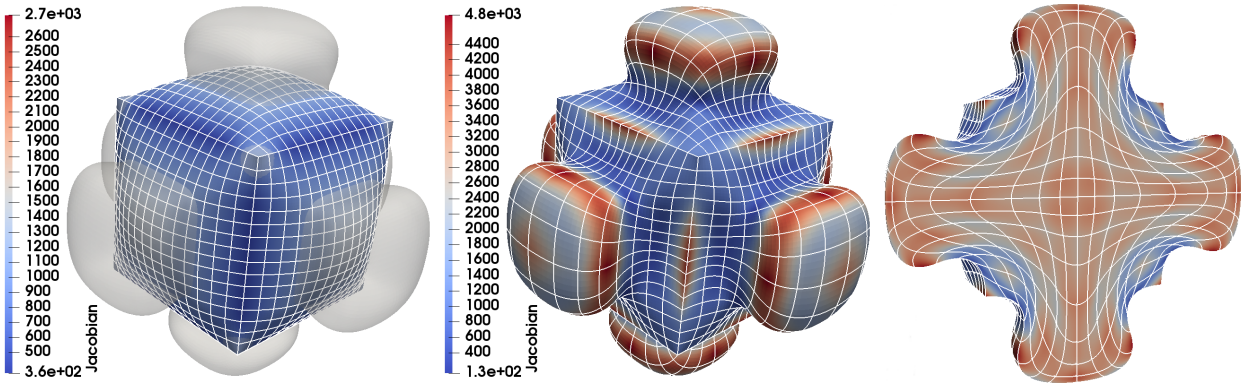

Figure 3.26. 3D puzzle piece: single-patch parametrization constructed by the mesh deformation approach (TINE+Newton's method, $N = 10$, $\nu = 0.46$, simplified domain $q = 2$, $m = 3$ in both directions). Simplified domain (left), result of mesh deformation (middle) and the domain cross-section (right).

3D male and female rotors

The 3D puzzle piece above is a rather artificial example. In practice, a complex object rarely has a convenient cuboid structure with six non-trivial boundary surfaces. A single-patch parametrization is thus inconvenient or even impossible, so a multi-patch structure for the object has to be created. The resulting patches often have simple shapes and can be parametrized using linear methods such

as the Coons patch, whereas an application of the mesh deformation approach would an obvious overkill.

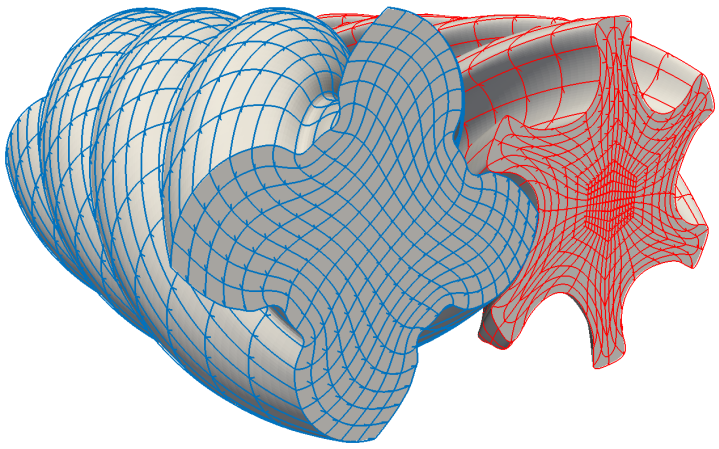

Figure 3.27. 3D male and female rotors constructed by helical extrusion of the 2D cross-sections.

Another common case with 3D objects, especially in engineering applications, is when the geometric complexity is concentrated in just two dimensions. The object is then defined by its cross-section and is extruded along a guiding line. In this case, the parametrization problem is reduced from 3D to 2D where it is reasonable to apply the mesh deformation approach. After that, the resulting control points of the cross-section are copied along the extrusion direction [72]. One example of such objects would be the already introduced rotors of the twin screw compressor, see Figure 3.27. Note that NURBS can model the helical curve exactly which is a huge advantage when dealing with objects which have to be modeled with a very high degree of accuracy.

3.5 Summary

I have dedicated this chapter to the single-patch domain parametrization problem. I have described several state-of-the-art parametrization techniques including the spring model, the Coons patch, the optimization-based methods and the elliptic grid generation (EGG) technique. The former two are linear and are only suitable for simple geometries. The latter two are nonlinear and, at least in theory, can produce bijective parametrizations for complex geometries. However, among the optimization-based methods, only the area-orthogonality quality measure provides a robust way to construct high-quality parametrizations. Unfortunately, this approach tends to be rather computationally costly. On the other hand, the EGG technique is an efficient method which always produces bijective parametrizations. The disadvantage of EGG is that the resulting parametrizations tend to have highly non-uniformly sized elements, especially if the domain's boundary contains concave regions.

After describing the state-of-the-art, I have proposed a novel domain parametrization approach based on mesh deformation. Within this approach, a parametrization for the target domain is constructed as a deformed configuration of a simplified geometry. The domain simplification is achieved by means of L^2-projection of the target domain's boundary on a coarse NURBS basis. The resulting simplified domain can be parametrized by any technique which is computationally inexpensive and grants a bijective result, for example, the Coons patch, EGG or even the mesh deformation approach applied recursively. Once the simplified domain is parametrized, it is deformed

using the stationary equations of nonlinear elasticity with the neo-Hookean material law that preserves the bijectivity of the deformation mapping. In order to solve the equations efficiently, I have proposed the tangential incremental nonlinear elasticity (TINE) algorithm. The TINE algorithm can either be used to efficiently construct high-quality initial guesses for Newton's method or as a stand-alone method to solve the equations of nonlinear elasticity. The computational cost of the mesh deformation approach based on the TINE algorithm is comparable to the EGG technique.

Apart from the simplified domain, two important parameters of the mesh deformation approach are Poisson's ratio in the material law and the number of incremental steps used for the TINE deformation algorithm. High values of Poisson's ratio help to preserve the bijectivity of the deformation mapping and to inherit the element size distribution from the simplified domain. As a disadvantage, they increase the minimal necessary number of incremental steps. The necessary number of steps is not known beforehand and has to be determined by trial and error in each particular case but rarely exceeds ten. Although adaptive algorithms for automatic determination of the incremental step size are available, they result in an unstable behavior, so a uniform step size should be preferred.

The TINE algorithm has also been compared to the incremental linear elasticity (ILE) method, which is used in many fluid-structure interaction applications as a mesh update method. When applied to the domain parametrization problem, the ILE algorithm yields bijective parametrizations similar (but not exactly equal) to the TINE-based mesh deformation approach. However, no explanation for why ILE produces bijective parametrizations is available, so further research has to be conducted. Still, the ILE-based mesh deformation approach is a robust way (at least empirically) to construct bijective parametrizations for complex domains. It is especially attractive due to its simplicity and relatively low computational cost.

Finally, I have compared the mesh deformation approach against the state-of-the-art parametrization techniques on two two-dimensional single-patch examples. In both examples, the proposed approach has produced parametrizations of the highest quality with respect to the element size uniformity. Additionally, I have demonstrated that the mesh deformation approach is applicable to multi-patch and three-dimensional problems.

4 | Fluid-structure interaction and mesh deformation

In Chapter 3, I have proposed a domain parametrization technique based on mesh deformation. To perform mesh deformation, I have developed the tangential incremental nonlinear elasticity (TINE) algorithm which allows to efficiently solve the stationary equations of nonlinear elasticity with the neo-Hookean material law. I have observed that the TINE algorithm is similar, both ideologically and in terms of the result, to the incremental linear elasticity (ILE) algorithm used as a mesh update method in various fluid-structure interaction (FSI) applications [11, 33, 34, 35, 36, 37]. Inspired and intrigued by this similarity, I have decided to apply TINE in an FSI simulation scenario as a mesh update method. My goal is to compare TINE against the ILE method in its "native environment" and to find whether the TINE method can offer any advantages in FSI simulations.

To that end, I have selected a classic FSI benchmark [38] and have set off to reproduce it. In order to implement an FSI solver, I have chosen the Arbitrary Lagrangian-Eulerian (ALE) formulation of FSI [11, 32] and the partitioned approach with strong coupling [73]. In what follows, I provide a description of the chosen benchmark. After that, I formulate standard mathematical models which define motion of an elastic structure and a viscous fluid as well as numerical algorithms that can be used to discretize them. I then focus on the fluid-structure coupling and methods for updating the computational mesh in the fluid domain. I compare several state-of-the-art mesh update methods and the TINE method in a simplified FSI-like setting and study how they react to the Jacobian-based local stiffening. Finally, I perform the simulation of the FSI benchmark, compare the results against the reference values and study the effect of different mesh update methods.

4.1 Benchmark description

The following benchmark problem is taken from [38]. It studies a flow of a viscous incompressible fluid a 2D channel as the fluid interacts with a submerged structure. The channel is a rectangle $[0, 2.2] \times [0, 0.41]$. The structure consists of a rigid disk $B_{0.05}(0.2, 0.2)$ and an elastic beam $[0.2, 0.6] \times [0.19, 0.21] \backslash B_{0.05}(0.2, 0.2)$ attached at its left end to the boundary of the disk. Figure 4.1 illustrates the setting; note that the geometry is intentionally non-symmetric.

The top and the bottom walls of the channel are impermeable for the fluid. The fluid enters the channel through the left wall with a prescribed velocity and exits through the right wall freely. The presence of the submerged structure changes the flow of the fluid and, in response, the fluid exerts a certain force on the structure. Thus, this is a *fluid-structure interaction* problem. Depending on the prescribed inflow velocity, this fluid force can result in a noticeable deformation of an elastic part of the structure. The deformation of the structure alters the shape of the flow channel, the flow itself and, as a consequence, the force exerted on the structure by the fluid. Such systems with a two-way interaction between the components are called *coupled* systems. To model such a system, one has

to model the behavior of its components subject to certain coupling conditions. In the next three sections, I first formulate models for each individual component, namely for the elastic structure and the fluid, and then describe the coupling.

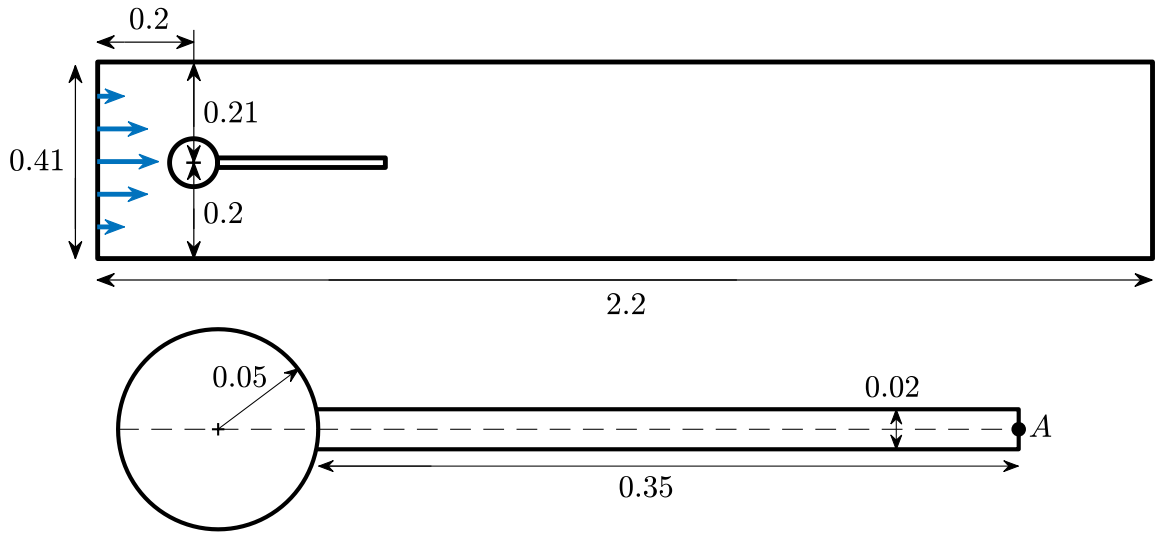

Figure 4.1. Top: initial configuration of the computational domain. Bottom: close-up on the immersed structure.

4.2 Structure modeling

In this section, I deal with the theory of motion of an elastic body. Building upon the introduction into solid mechanics given in Section 3.3.3, I formulate the time-dependent equations of nonlinear elasticity, as described in [30]. After that, I outline time integration schemes that can be used to solve these equations numerically. Finally, I validate my implementation of the time-dependent nonlinear elasticity solver on partial benchmarks.

4.2.1 Equations of motion

Let $\Omega_0 \subset \mathbb{R}^3$ be a reference configuration of an elastic body. Assume that the body undergoes a deformation over the period of time $[0, T]$ described by a displacement field $\mathbf{u}(\widehat{\mathbf{x}}, t) : \Omega_0 \times (0, T) \to \mathbb{R}^3$. Then the motion of the body in the presence of a volumetric load $\mathbf{g}(\widehat{\mathbf{x}}, t)$ is governed by the local conservation equations of linear momentum:

$$\rho \ddot{\mathbf{u}} = \operatorname{div}_{\widehat{\mathbf{x}}} \mathbf{P}(\mathbf{u}) + \rho \mathbf{g} \text{ in } \Omega_0 \times (0, T). \tag{4.1}$$

For brevity, I refer to (4.1) as the *equations of motion*. Here, $\ddot{\mathbf{u}}$ stays for the second derivative in time of $\mathbf{u}$. The *density* ρ is assumed to be constant throughout the reference configuration. Furthermore, let the body be fixed on the portion of the domain boundary and be subject to a surface load on the rest of the boundary:

$$\mathbf{u} = \mathbf{0} \text{ on } \partial\Omega_{0\mathcal{D}} \times (0, T) \quad \text{and} \quad \mathbf{P}(\mathbf{u}) \cdot \mathbf{n} = \rho \mathbf{f} \text{ on } \partial\Omega_{0\mathcal{N}} \times (0, T). \tag{4.2}$$

Finally, the equations of motion (4.1) are equipped with suitable initial conditions for the displacement $\mathbf{u}$ and *velocity* $\dot{\mathbf{u}}$:

$$\mathbf{u}(\cdot, 0) = \mathbf{u}_0 \text{ in } \Omega_0 \quad \text{and} \quad \dot{\mathbf{u}}(\cdot, 0) = \dot{\mathbf{u}}_0 \text{ in } \Omega_0. \tag{4.3}$$

A function $\mathbf{u}(\cdot,t) \in C^2(\Omega_0) \cap C^0(\bar{\Omega}_0)$ satisfying (4.1–4.3) is called a classical solution of the equations of motion.

In order to apply the isogeometric Galerkin method, I rewrite the equations of motion (4.1) in a weak-in-space form. Since the Dirichlet boundary condition (4.2) is homogeneous, both the trial solution and weighting functions spaces can be defined as $\mathcal{V}_0 = \{\mathbf{w} \in H(\Omega_0)^3 \mid \mathbf{w} = \mathbf{0} \text{ on } \partial\Omega_{0\mathcal{D}}\}$. Then the weak-in-space form of (4.1) takes the following form:

find $\mathbf{u}(\cdot,t) \in \mathcal{V}_0$ such that $\forall \mathbf{w} \in \mathcal{V}_0$

$$\rho \int_{\Omega_0} \ddot{\mathbf{u}}^T \mathbf{w} d\widehat{\mathbf{x}} + \int_{\Omega_0} \mathbf{S}(\mathbf{u}) : \delta\mathbf{E}(\mathbf{u})[\mathbf{w}] d\widehat{\mathbf{x}} = \rho \int_{\Omega_0} \mathbf{g}^T \mathbf{w} d\widehat{\mathbf{x}} + \rho \int_{\partial\Omega_{0\mathcal{N}}} \mathbf{f}^T \mathbf{w} d\widehat{\mathbf{s}}. \tag{4.4}$$

I discretize the weak problem (4.4) in space similarly to the stationary case. To that end, I use the NURBS basis functions corresponding to the parametrization $\mathbf{G}_0$ of the reference domain Ω_0. The time-dependent isogeometric solution $\mathbf{u}_h$ is defined as

$$\mathbf{u}_h(\widehat{\mathbf{x}},t) = \sum_{I=1}^{3n} d_I(t) \mathbf{N}_I(\widehat{\mathbf{x}}). \tag{4.5}$$

Here, $\mathbf{N}_I : \Omega_0 \to \mathbb{R}^3$ are the vector-valued NURBS basis functions (3.65), $d_I(t) \in \mathbb{R}$ are time-dependent degrees of freedom and I is a linear index corresponding to the dimension-wise ordering (3.64). Similarly to the stationary case, unknown degrees of freedom $d_I(t)$, $I \in \mathcal{F}$ form a vector of unknowns $\mathbf{y}(t)$. Due to the homogeneous Dirichlet boundary condition, vectors of Dirichlet degrees of freedom $\mathbf{y}^{\mathcal{D}}$ and $\Delta\mathbf{y}^{\mathcal{D}}$ are always zero, so I omit them in what follows. Linear span of basis functions $\mathbf{N}_I$, $I \in \mathcal{F}$ defines a discrete function space $\mathcal{V}_{0,h} \subset \mathcal{V}_0$. Upon replacing $\mathbf{u}$ by $\mathbf{u}_h$ in the weak form (4.4) and testing it with the basis of $\mathcal{V}_{0,h}$, we obtain the following system of ordinary differential equations of second order:

$$\mathbf{M}\ddot{\mathbf{y}} + \mathbf{R}(\mathbf{y}) = \mathbf{L}. \tag{4.6}$$

Here, $\mathbf{M}$ is a $|\mathcal{F}| \times |\mathcal{F}|$ *mass matrix*, $\mathbf{R}(\mathbf{y})$ is a residual vector from (3.73) and $\mathbf{L} \in \mathbb{R}^{|\mathcal{F}|}$ is a *loading vector*. Entries M_{IJ} and L_I of $\mathbf{M}$ and $\mathbf{L}$ are computed as

$$M_{IJ} = \rho \int_{\Omega_0} \mathbf{N}_I^T \mathbf{N}_J d\widehat{\mathbf{x}} \quad \text{and} \quad L_I = \rho \int_{\Omega_0} \mathbf{N}_I^T \mathbf{g} d\widehat{\mathbf{x}} \;\text{ for }\; I,J \in \mathcal{F}. \tag{4.7}$$

Initial conditions (4.3) for the equations of motion translate into initial conditions for the ODE (4.6). If $\mathbf{u}_0$ and $\dot{\mathbf{u}}_0$ are nontrivial, they can be projected on the discrete space $\mathcal{V}_{0,h}$ using the mass matrix $\mathbf{M}$ in order to obtain initial degrees of freedom $\mathbf{y}$ and $\mathbf{y}_0$.

4.2.2 Time integration

In order to solve numerically ODE (4.6), let us split the time period $[0,T]$ into N *time steps* $\Delta t = T/N$. The displacement vector $\mathbf{y}(t)$ has to be computed at discrete time moments $t_i = i\Delta t$, $i = 1,\dots,N$. In general, there exist two groups of methods for solving ODE (4.6), namely *explicit* and *implicit* integration schemes. Explicit schemes compute the solution at time t_{i+1} using only quantities at time t_i (and maybe before). As a result, only a system of linear algebraic equations has to be solved at every time step. Therefore, explicit methods can be very efficient, especially if the mass matrix $\mathbf{M}$ is replaced by a diagonal *lumped mass matrix*. The disadvantage of explicit

schemes is the limitation on the time step due to a stability criterion. Explicit schemes are ideal for impact problems, such as car-crash analysis, where a very short period of time needs to be resolved using a large number of small time steps.

On the contrary, implicit methods replace the time derivatives using not only quantities which are known at time t_i but also the yet unknown quantities at time t_{i+1}. Since the ODE (4.6) is nonlinear, implicit schemes require solving a system of nonlinear algebraic equation at every time step, for example using Newton's method described in Section 3.3.4. On the other hand, implicit methods can be constructed is such a way that they are unconditionally stable, thus the time step is not limited. Because of that, they are better suited for simulating phenomena happening over extended periods of time, like low-frequency structural vibrations in FSI.

In this work, I use the most popular implicit time integration scheme for elasticity problems— the *Newmark method*. For the description of the scheme, let $\mathbf{a}$ denote the vector of discrete accelerations $\ddot{\mathbf{u}}$ and $\mathbf{s}$ denote the vector of discrete velocities $\dot{\mathbf{u}}$. With this notation, ODE (4.6) at time t_i has the form

$$\mathbf{M}\mathbf{a}_i + \mathbf{R}(\mathbf{y}_i) = \mathbf{L}_i, \tag{4.8}$$

where the subscript i indicates that the relevant quantity has to be computed at time t_i. At time $t_0 = 0$, initial displacements $\mathbf{y}_0$ and velocities $\mathbf{s}_0$ are given by the initial conditions (4.3). Initial accelerations $\mathbf{a}_0$ can be computed from

$$\mathbf{a}_0 = \mathbf{M}^{-1}\big(-\mathbf{R}(\mathbf{y}_0) + \mathbf{L}_0\big). \tag{4.9}$$

The Newmark method is based on the following approximations for displacements and velocities at time t_{i+1}:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \Delta t \mathbf{s}_i + \frac{\Delta t^2}{2}\big((1-2\beta)\mathbf{a}_i + 2\beta\mathbf{a}_{i+1}\big), \tag{4.10}$$

$$\mathbf{s}_{i+1} = \mathbf{s}_i + \Delta t\big((1-\gamma)\mathbf{a}_i + \gamma\mathbf{a}_{i+1}\big). \tag{4.11}$$

The parameters $\beta \in [0, 0.5]$ and $\gamma \in [0, 1]$ are constants which determine the behavior of the method. An optimal choice is $\beta = 0.25$ and $\gamma = 0.5$, when the Newmark method becomes second-order accurate in time. However, the Newmark method was originally developed for linear problems. Special care is required when applying it to nonlinear problems.

In order to implement the Newmark method, it is convenient to re-express the accelerations $\mathbf{a}_{i+1}$ and velocities $\mathbf{s}_{i+1}$ in terms of the displacements $\mathbf{y}_{i+1}$:

$$\mathbf{a}_{i+1} = \alpha_1(\mathbf{y}_{i+1} - \mathbf{y}_i) - \alpha_2\mathbf{s}_i - \alpha_3\mathbf{a}_i, \tag{4.12}$$

$$\mathbf{s}_{i+1} = \alpha_4(\mathbf{y}_{i+1} - \mathbf{y}_i) + \alpha_5\mathbf{s}_i + \alpha_6\mathbf{a}_i. \tag{4.13}$$

Here, the auxiliary constants α_* are defined as

$$\alpha_1 = \frac{1}{\beta\Delta t^2}, \qquad \alpha_2 = \frac{1}{\beta\Delta t}, \qquad \alpha_3 = \frac{1-2\beta}{2\beta}, \tag{4.14}$$

$$\alpha_4 = \frac{\gamma}{\beta\Delta t}, \qquad \alpha_5 = 1 - \frac{\gamma}{\beta}, \qquad \alpha_6 = (1 - \frac{\gamma}{2\beta})\Delta t. \tag{4.15}$$

After inserting (4.12) into (4.8), we obtain the following system of nonlinear algebraic equations which has to be solved at every time step:

$$\mathbf{Z}(\mathbf{y}_{i+1}) = \mathbf{M}\big(\alpha_1(\mathbf{y}_{i+1} - \mathbf{y}_i) - \alpha_2\mathbf{s}_i - \alpha_3\mathbf{a}_i\big) + \mathbf{R}(\mathbf{y}_{i+1}) - \mathbf{L}_{i+1} = \mathbf{0}. \tag{4.16}$$

To that end, Newton's method is used. Provided an intermediate solution $\mathbf{y}_{i+1}^s$ for time t_{i+1}, the update $\Delta\mathbf{y}_{i+1}^{s+1}$ is computed by solving the following linear system:

$$\big(\alpha_1\mathbf{M} + \mathbf{K}_T(\mathbf{y}_{i+1}^s)\big)\Delta\mathbf{y}_{i+1}^{s+1} = -\mathbf{Z}(\mathbf{y}_{i+1}^s). \tag{4.17}$$

Here, the tangent stiffness matrix $\mathbf{K}_T(\mathbf{y}_{i+1}^s)$ is assembled in exactly the same way as in the stationary case (3.73), which allows one to reuse an already implemented solver for stationary nonlinear elasticity. The update $\Delta\mathbf{y}_{i+1}^{s+1}$ is used to define the next intermediate solution $\mathbf{y}_{i+1}^{s+1}$ as $\mathbf{y}_{i+1}^s + \Delta\mathbf{y}_{i+1}^{s+1}$. Iterations of Newton's method at each time step continue until convergence, and the last intermediate solution $\mathbf{y}_{i+1}^s$ is accepted as a solution $\mathbf{y}_{i+1}$ at time t_{i+1}. As an initial guess $\mathbf{y}_{i+1}^0$ at time t_{i+1}, the displacements vector $\mathbf{y}_i$ from the previous time step can be used. If the time step is small enough, this provides a good initial guess which allows Newton's method to converge within a few iterations. Once $\mathbf{y}_{i+1}$ is found, $\mathbf{a}_{i+1}$ and $\mathbf{s}_{i+1}$ are computed from formulas (4.12–4.13).

4.2.3 Partial benchmark CSM1: stationary deflection

I validate my implementation of the isogeometric nonlinear elasticity solver on partial benchmarks from [38]. The first benchmark, CSM1, is a stationary deflection test for the elastic beam attached to the rigid disk, see Figure 4.1. It involves solving the stationary equations of nonlinear elasticity

$$-\operatorname{div}_{\hat{\mathbf{x}}}\mathbf{P}(\mathbf{u}) = \rho\mathbf{g} \text{ in } \Omega_0 \tag{4.18}$$

with the St. Venant-Kirchhoff material law (3.45) and the following boundary conditions:

$$\mathbf{u} = \mathbf{0} \text{ on } \partial\Omega_{0\mathcal{D}} \quad \text{and} \quad \mathbf{P}(\mathbf{u})\cdot\mathbf{n} = \mathbf{0} \text{ on } \partial\Omega_{0\mathcal{N}}. \tag{4.19}$$

Here, $\partial\Omega_{0\mathcal{D}}$ is the left end of the beam (the one attached to the sphere), and $\partial\Omega_{0\mathcal{N}}$ is the rest of the beam's boundary. For the geometry, I use a quadratic two-element[1] single-patch NURBS parametrization, see Figure 4.2, which I uniformly h-refine for the analysis. The following parameters are used: density $\rho = 10^3$ kg·m^{-3}, Poisson's ratio $\nu = 0.4$, Young's modulus $E = 1.4\times10^6$ kg·m^{-1}·s^{-2} and gravitational acceleration $\mathbf{g} = (0,2)^T$ m·s^{-2}. The quantity of interest is displacement of the beam's right end midpoint denoted by A.

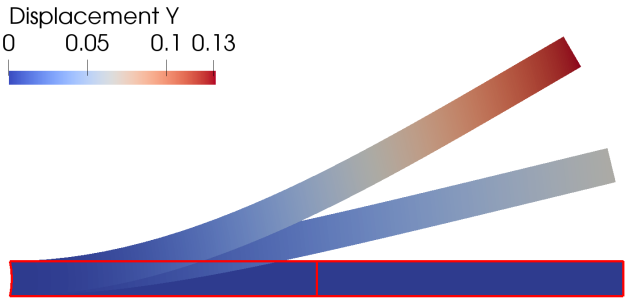

Figure 4.2. Benchmark CSM1 and CSM3: reference configuration of the beam (bottom) with the parametrization structure (red), equilibrium deflection state (middle) and maximum deflection state during elastic oscillations (top).

The simulation results are presented in Table 4.1, where I report the refinement level, the number of degrees of freedom, the horizontal and vertical displacement $u_x(A)$ and $u_y(A)$ of the point A and

[1] A single-element parametrization is also possible. However, this two-element parametrization shares the basis on the boundary with the parametrization of the surrounding fluid domain.

the computational time[2]. The results are in perfect agreement with the reference values from [38]. Figure 4.2 illustrates the deformed state of the beam.

h-lvl	#DoF	T_{comp}	$u_x(A)$ [10^{-2} m]	$u_y(A)$ [10^{-2} m]
5	4420	1.6s	-0.7172	6.602
6	17028	5.4s	-0.7183	6.608
7	66820	21s	-0.7186	6.609
8	264708	1m26s	-0.7187	6.610
Ref	435776	N/A	-0.7187	6.610

Table 4.1. Benchmark CSM1: simulation results.

4.2.4 Partial benchmark CSM3: elastic oscillations

Benchmark CSM3 is a time-dependent simulation of the elastic beam oscillating in the presence of an external volumetric force. It involves solving the equations of motion

$$\rho\ddot{\mathbf{u}} = \mathrm{div}_{\widehat{\mathbf{x}}}\,\mathbf{P}(\mathbf{u}) + \rho\mathbf{g} \text{ in } \Omega_0 \text{ in } \Omega_0 \times [0,T] \tag{4.20}$$

for $T = 10$s with the St. Venant-Kirchhoff material law (3.45), boundary conditions (4.19) and the following initial conditions:

$$\mathbf{u}(\cdot,0) = \mathbf{0} \text{ in } \Omega_0 \quad \text{and} \quad \dot{\mathbf{u}}(\cdot,0) = \mathbf{0} \text{ in } \Omega_0. \tag{4.21}$$

I use the same geometry parametrization and parameter values as for CSM1 and perform temporal discretization using the Newmark method with $\beta = 0.25$ and $\gamma = 0.5$. As before, the quantity of interest is the displacement of the point A. However, it varies with time in this simulation. Since the displacement components $u_x(A)$ and $u_y(A)$ are expected to behave periodically, they can each be described accurately enough using just three numbers: the mean and the amplitude, defined as

$$u_{mean} = \frac{1}{2}(u_{max} + u_{min}) \quad \text{and} \quad u_{amp} = \frac{1}{2}(u_{max} - u_{min}), \tag{4.22}$$

and the frequency defined as $1/\tau$, where τ is the length of the oscillation period. The last oscillation period is used to compute the values.

h-lvl	Δt [s]	#DoF	T_{comp}	$u_x(A)$ [10^{-2} m]	$u_y(A)$ [10^{-2} m]	Freq [s^{-1}]
4	0.01	1188	3m34s	-1.4537±1.4530	6.4518±6.4684	1.0989
5	0.01	4420	14m8s	-1.4630±1.4634	6.4737±6.4904	1.0989
6	0.01	17028	52m1s	-1.4645±1.4645	6.4763±6.4945	1.0989
7	0.01	66820	3h21m	-1.4645±1.4650	6.4765±6.4947	1.0989
7	0.005	66820	6h34m	-1.4303±1.4304	6.3604±6.5159	1.0995
Ref	0.005	98820	N/A	-1.4305±1.4305	6.3607±6.5160	1.0995

Table 4.2. Benchmark CSM3: simulation results.

[2]This is not CPU but real time. Of course, the real time is a poor measure of the algorithm performance since it varies highly between different machines. My intention here is to give some perspective on how much time different simulations described in this chapter take. All the simulations were performed on a Dell XPS15 (2017 edition) laptop with a 2.8GHz Intel Core i7-700HQ CPU. The implementation is based on G+Smo with Eigen as a linear algebra back-end and the Intel-MKL distribution of Pardiso [61] as a direct linear solver. The linear system assembly and solution are implemented in parallel and run on 8 hyper-threads.

The results of CSM3 are presented in Table 4.2, including the h-refinement level, the time step Δt in seconds, the number of degrees of freedom, the computational time and the displacement of the point A. The results are once again in perfect agreement with the reference values. As Figure 4.3 depicts, the behavior of $u_x(A)$ and $u_y(A)$ shows no signs of energy loss over time. Overall, my implementation of the nonlinear elasticity solver seems to be robust and accurate.

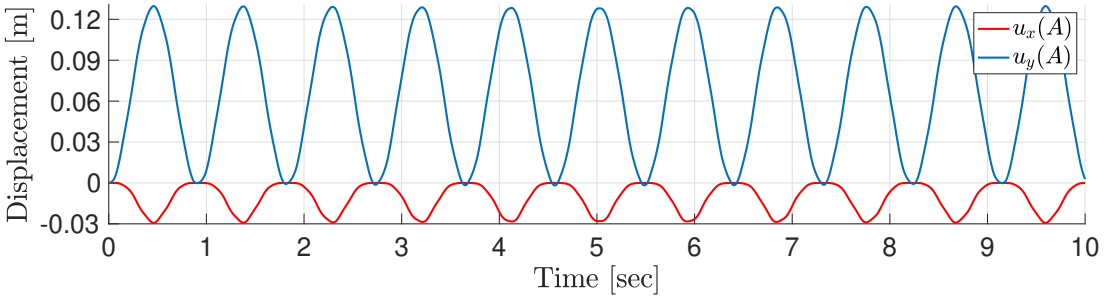

Figure 4.3. Benchmark CSM3: displacement of the point A.

4.3 Fluid modeling

This section deals with modeling of the fluid behavior. First, I give a brief introduction into the finite element method for incompressible flow problems, based mainly on [74]. I introduce the incompressible Navier-Stokes equations (INSE) and show how they can be spatially discretized using IGA. Then, I describe two time-integration schemes, one requiring a solution of a system of nonlinear algebraic equations at every time step, and another that tries to avoid doing that while still retaining a relatively high accuracy in time. Finally, I validate my implementation of the isogeometric INSE solver on partial benchmarks.

4.3.1 Incompressible Navier-Stokes equations

Let $\Omega \in \mathbb{R}^3$ be a domain occupied by a flowing fluid. The motion of the fluid over a time period $[0, T]$ is governed by the local conservation equations for linear momentum and mass:

$$\rho \dot{\mathbf{v}} + \rho \nabla_{\mathbf{x}} \mathbf{v} \cdot \mathbf{v} = \operatorname{div}_{\mathbf{x}} \boldsymbol{\sigma} + \rho \mathbf{g} \text{ in } \Omega \times [0, T], \tag{4.23}$$

$$\dot{\rho} + \operatorname{div}_{\mathbf{x}}(\rho \mathbf{v}) = 0 \text{ in } \Omega \times [0, T]. \tag{4.24}$$

Here, $\mathbf{v} : \Omega \to \mathbb{R}^3$ is a vector field describing the *velocity* of the fluid at a point of space $\mathbf{x} \in \Omega$. $\rho \in \mathbb{R}$ is the fluid density, $\boldsymbol{\sigma} : \Omega \to \mathbb{R}^{3\times 3}$ is a stress tensor describing the internal forces arising in the fluid, and $\mathbf{g} : \Omega \to \mathbb{R}^3$ denotes an external volumetric force. In this work, I assume that the fluid is an *incompressible Newtonian fluid*. This means that the fluid behavior is characterized by the incompressibility condition $\rho = const$ and the constitutive law

$$\boldsymbol{\sigma}(\mathbf{v}, p) = -p\mathbf{I} + \rho\nu(\nabla_{\mathbf{x}} \mathbf{v} + \nabla_{\mathbf{x}} \mathbf{v}^T). \tag{4.25}$$

Here, $p : \Omega \to \mathbb{R}$ is a *pressure* field and $\nu \in \mathbb{R}$ is the *kinematic*[3] *viscosity* of the fluid. After plugging the incompressibility condition and the constitutive law (4.25) into the general equations (4.23–4.23), we obtain a simplified system:

$$\rho \dot{\mathbf{v}} + \rho \nabla_{\mathbf{x}} \mathbf{v} \cdot \mathbf{v} = \rho\nu \Delta_{\mathbf{x}} \mathbf{v} - \nabla_{\mathbf{x}} p + \rho \mathbf{g} \text{ in } \Omega \times [0, T], \tag{4.26}$$

$$\operatorname{div}_{\mathbf{x}} \mathbf{v} = 0 \text{ in } \Omega \times [0, T]. \tag{4.27}$$

[3] Do not confuse with the *dynamic viscosity* $\mu = \rho\nu$ (also known as the *absolute viscosity*).

I refer to this system as the *incompressible Navier-Stokes equations* (INSE), or simply the *Navier-Stokes equations*. The system has two unknowns−the velocity $\mathbf{v}$ and the pressure p. The latter is defined only up to an additive constant. The second term on the left in (4.26) is called the *advection term*, and the first term on the right is called the *diffusion term*. Note that the system is nonlinear because of the advection term.

In comparison to the equations of nonlinear elasticity, I use the subscript $\mathbf{x}$ instead of $\widehat{\mathbf{x}}$ to underline that the spatial differentiation is performed with respect to the spatial coordinates in Ω and not with respect to the material points of the fluid. The difference becomes more prominent once we allow the fluid domain to move as, for example, in FSI simulations.

To complete the system of the Navier-Stokes equations, one has to supply it with the appropriate initial and boundary conditions. Being first-order in time, the Navier-Stokes equations require initial conditions only for the velocity:

$$\mathbf{v}(\cdot, 0) = \mathbf{v}_0 \text{ in } \Omega. \tag{4.28}$$

Note that the pressure requires no initial conditions. In fact, pressure is an algebraic variable because its time derivative does not enter the equations. From this point of view, the incompressible Navier-Stokes equations are a system of *differential-algebraic equations* (DAE). In practice, however, many time-integration schemes seem to be oblivious to this fact but still perform well.

In this work, I distinguish between three types of boundary conditions: *no-slip*, *inflow* and *do-nothing*. Accordingly, let the boundary $\partial\Omega$ be split into three parts $\partial\Omega_{ns}$, $\partial\Omega_{in}$ and $\partial\Omega_{out}$. The first two types of boundary conditions, namely the no-slip and the inflow, are defined as

$$\mathbf{v} = \mathbf{0} \text{ on } \partial\Omega_{ns} \times [0, T] \quad \text{and} \quad \mathbf{v} = \mathbf{v}_{in} \text{ on } \partial\Omega_{\mathcal{D}} \times [0, T]. \tag{4.29}$$

The no-slip condition models an impermeable boundary, whereas the inflow condition specifies the velocity at which the fluid enters the domain. Note that $\mathbf{v}_{in}$ can be time-dependent. When necessary, I refer to the both conditions combined as the Dirichlet boundary condition $\mathbf{v} = \mathbf{v}_{\mathcal{D}}$, and I use $\partial\Omega_{\mathcal{D}}$ to denote the union $\partial\Omega_{ns} \cup \partial\Omega_{in}$. Finally, the do-nothing boundary condition

$$\boldsymbol{\sigma}(\mathbf{v}, p) \cdot \mathbf{n} = \mathbf{0} \text{ on } \partial\Omega_{out} \times [0, T] \tag{4.30}$$

models a free flow of the fluid from the domain. The do-nothing condition is a natural condition for the weak formulation of the INSE and as such requires no special treatment to enforce it. Moreover, it has an added benefit of setting a zero base value for the pressure, thus solving the problem that the pressure is defined only up to an additive constant. A pair $(\mathbf{v}(\cdot, t), p(\cdot, t)) \in C^2(\Omega) \cap C^0(\bar{\Omega}) \times C^1(\Omega)$ that satisfies the equations (4.26–4.27) as well as the initial and boundary conditions (4.28–4.30) is called a classical solution of the incompressible Navier-Stokes equations.

4.3.2 Spatial discretization with IGA

In this section, I describe spatial discretization of the incompressible Navier-Stokes equations using the isogeometric Galerkin method. To that end, I first introduce the Oseen equations which are auxiliary linear equations appearing in many solution approaches to the INSE. Using the Oseen equations as an example, I discuss saddle-points problems and mixed isogeometric elements that can be used to solve them. After that, I consider the steady-state Navier-Stokes equations and discretize them using mixed isogeometric methods. In order to solve the resulting system of nonlinear algebraic equations, I formulate the Picard and Newton's linearizations of the INSE and describe the assembly of the corresponding linear systems.

Oseen equations

The *Oseen equations* are a system of linear partial differential equations which can be considered a generalization of the *advection-diffusion equation*. They have the following form:

$$-\nu\Delta_{\mathbf{x}}\mathbf{v} + \nabla_{\mathbf{x}}\mathbf{v}\cdot\mathbf{b} + c\mathbf{v} + \nabla_{\mathbf{x}}p = \mathbf{g} \text{ in } \Omega, \tag{4.31}$$

$$\operatorname{div}_{\mathbf{x}}\mathbf{v} = 0 \text{ in } \Omega, \tag{4.32}$$

where $\mathbf{b} : \Omega \to \mathbb{R}^3$ is a given divergence-free *advective velocity* and c in the *reaction term* $c\mathbf{v}$ is a non-negative scalar. Let us rewrite the equations in a weak form. To that end, let us define the velocity trial solution space $\mathcal{V}$, velocity weighting function space $\mathcal{V}_0$ and the pressure space $\mathcal{Q}$ as

$$\mathcal{V} = \{\mathbf{w} \in H^1(\Omega)^3 \mid \mathbf{w} = \mathbf{v}_{\mathcal{D}} \text{ on } \partial\Omega_{\mathcal{D}}\}, \quad \mathcal{V}_0 = \{\mathbf{w} \in H^1(\Omega)^3 \mid \mathbf{w} = \mathbf{0} \text{ on } \partial\Omega_{\mathcal{D}}\} \tag{4.33}$$

$$\text{and } \mathcal{Q} = \{q \in L^2(\Omega) \mid \int_\Omega q d\mathbf{x} = 0\}. \tag{4.34}$$

Then the weak form of the Oseen equations reads:

find $(\mathbf{v}, p) \in \mathcal{V} \times \mathcal{Q}$ such that

$$\int_\Omega \Big(\nu\nabla_{\mathbf{x}}\mathbf{v} : \nabla_{\mathbf{x}}\mathbf{w} + (\nabla_{\mathbf{x}}\mathbf{v}\cdot\mathbf{b})^T\mathbf{w} + c\mathbf{v}^T\mathbf{w} + p\operatorname{div}_{\mathbf{x}}\mathbf{w}\Big)d\mathbf{x} = \int_\Omega \mathbf{g}^T\mathbf{w}d\mathbf{x} \ \ \forall\mathbf{w} \in \mathcal{V}_0, \tag{4.35}$$

$$\int_\Omega \operatorname{div}_{\mathbf{x}}\mathbf{v} q d\mathbf{x} = 0 \ \ \forall q \in \mathcal{Q}. \tag{4.36}$$

Let us define bilinear forms $a(\mathbf{v}, \mathbf{w}) = \int_\Omega \Big(\nu\nabla_{\mathbf{x}}\mathbf{v} : \nabla_{\mathbf{x}}\mathbf{w} + (\nabla_{\mathbf{x}}\mathbf{v}\cdot\mathbf{b})^T\mathbf{w} + c\mathbf{v}^T\mathbf{w}\Big)d\mathbf{x}$ and $b(\mathbf{v}, q) = \int_\Omega \operatorname{div}_{\mathbf{x}}\mathbf{v} q d\mathbf{x}$ and a linear form $l(\mathbf{w}) = \int_\Omega \mathbf{g}^T\mathbf{w}d\mathbf{x}$. Then the variational problem (4.35–4.36) can be cast into the following abstract form:

$$\text{find } (\mathbf{v}, p) \in \mathcal{V} \times \mathcal{Q} \text{ such that } \ \ a(\mathbf{v}, \mathbf{w}) + b(\mathbf{w}, p) = l(\mathbf{w}) \ \ \forall\mathbf{w} \in \mathcal{V}, \tag{4.37}$$

$$b(\mathbf{v}, q) = 0 \ \ \forall q \in \mathcal{Q}. \tag{4.38}$$

Variational problems of the form (4.37–4.38) are called *saddle-point problems* since their solution $(\mathbf{v}, p)$ is also a solution of the constrained optimization problem

$$J(\mathbf{v}) = \frac{1}{2}a(\mathbf{v}, \mathbf{v}) - l(\mathbf{v}) \to \min, \tag{4.39}$$

$$b(\mathbf{v}, q) = 0 \ \ \forall q \in \mathcal{Q}. \tag{4.40}$$

If we introduce an associated Lagrangian

$$L(\mathbf{v}, p) = J(\mathbf{v}) + b(\mathbf{v}, p), \tag{4.41}$$

then each solution $(\mathbf{v}, p)$ of (4.39–4.40) satisfies the following property:

$$L(\mathbf{v}, q) \leqslant L(\mathbf{v}, p) \leqslant L(\mathbf{w}, p) \ \ \forall(\mathbf{w}, q) \in \mathcal{V} \times \mathcal{Q}. \tag{4.42}$$

In other words, $(\mathbf{v}, p)$ is a saddle point of the Lagrangian (4.41). In light of that, one may choose to interpret the pressure p as a Lagrange multiplier associated with the incompressibility constraint $\operatorname{div}\mathbf{v} = 0$.

The existence and uniqueness theorem for an abstract saddle-point problem (4.37–4.38) states that the problem has a unique solution if the following two conditions are satisfied. First, $a(\mathbf{v}, \mathbf{w})$ has to be coercive (2.25). Second, $b(\mathbf{v}, q)$ has to satisfy the famous *Ladyshenskaya-Babuška-Brezzi condition*, also known as the *inf-sup condition*, which requires that there exists a constant $\beta > 0$ such that

$$\inf_{q \in \mathcal{Q}, q \neq 0} \sup_{\mathbf{w} \in \mathcal{V}, \mathbf{w} \neq \mathbf{0}} \frac{b(\mathbf{w}, q)}{||\mathbf{w}||_{\mathcal{V}} ||q||_{\mathcal{Q}}} \geqslant \beta. \tag{4.43}$$

The inf-sup condition can be seen as a generalization of the coercivity condition for bilinear forms with arguments from different spaces.

For the Oseen equations, coercivity of $a(\mathbf{v}, \mathbf{w})$ follows from the divergence-free property of the advective velocity $\mathbf{b}$ and the non-negativity of the scalar c in the reaction term. In [74], it is shown that the bilinear form $b(\mathbf{v}, q)$ on the spaces $\mathcal{V}$ and $\mathcal{Q}$ satisfies the inf-sup condition. Therefore, the variational formulation (4.35–4.36) is well-posed and has a unique solution.

Another important property of (4.35–4.36) is that the bilinear form $a(\mathbf{v}, \mathbf{w})$ is not symmetric due to the advection term. This fact makes the numerical treatment of the problem more complex. Concretely, let us introduce the *Reynolds number* Re,

$$Re = \frac{vL}{\nu}, \tag{4.44}$$

where $v \in \mathbb{R}$ is the *characteristic speed* of the flow, $L \in \mathbb{R}$ is the *characteristic length* associated with the fluid geometry, and ν is the kinetic viscosity. If the viscosity ν is relatively low, the contribution of the diffusion term is dwarfed by the advection term in both the INSE and Oseen equations. In this case, the fluid flow is said to be *advection dominated*. For advection-dominated flows, the finite element methods based on the continuous Galerkin method tend to produce solutions with spurious oscillations. This effect can be mitigated by introducing additional *stabilization terms* into the weak form, see [11, 74]. One example would be the *SUPG* stabilization. It is hard to pinpoint exactly how high the Reynolds number must be before the stabilization becomes necessary. In this work, I have performed simulations with $Re \leqslant 200$ and have observed no signs of spurious oscillations. Therefore, I do not use any type of stabilization.

Mixed isogeometric elements

In order to apply the isogeometric Galerkin method to the weak problem (4.35–4.36), I use NURBS basis functions associated with the parametrization of the domain Ω to construct discrete function spaces $\mathcal{V}_{0,h} \subset \mathcal{V}_0$, $\mathcal{V}_h = \mathbf{v}_{\mathcal{D}} + \mathcal{V}_{0,h}$ and $\mathcal{Q}_h \subset \mathcal{Q}$. Since different discrete space are used for velocity and pressure, the approach falls under the category of *mixed finite elements*. The discrete spaces cannot be chosen arbitrarily because they have to satisfy the inf-sup condition (4.43), lest the Galerkin method becomes unstable.

By applying h-refinement and degree elevation in various combinations, different discrete spaces can be constructed. The mathematical analysis conducted in [75] suggests two types of stable mixed isogeometric elements $\mathcal{V}_h / \mathcal{Q}_h$. The first type are the *Taylor-Hood elements* (TH) which are well-known in the classical finite element analysis. The Taylor-Hood elements are constructed by choosing a discrete pressure space $\mathcal{N}_r^p$ (2.29) and then applying degree elevation once to obtain a discrete space $\mathcal{N}_r^{p+1}$ for velocity. I denote the TH mixed elements by $\mathcal{N}_r^{p+1} / \mathcal{N}_r^p$. The second type of stable mixed elements are the *subgrid elements* (SG) denoted by $\mathcal{N}_{r+1}^p / \mathcal{N}_r^p$. This time, the velocity space $\mathcal{N}_{r+1}^p$ is obtained from the pressure space $\mathcal{N}_r^p$ by applying h-refinement once.

Let $\mathbf{N}_I(\mathbf{x})$, $I = 1, \ldots, 3n$ denote vector-valued NURBS basis functions for velocity and $M_i(\mathbf{x})$, $i = 1, \ldots, m$ denote scalar-valued NURBS basis functions for pressure. Then the isogeometric solution $(\mathbf{v}_h, p_h) \in \mathcal{V}_h \times \mathcal{Q}_h$ can be written as

$$\mathbf{v}_h(\mathbf{x}) = \sum_{I=1}^{3n} v_I \mathbf{N}_I(\mathbf{x}) \quad \text{and} \quad p_h(\mathbf{x}) = \sum_{i=1}^{m} q_i M_i(\mathbf{x}), \tag{4.45}$$

where v_I and q_i are degrees of freedom for the velocity and pressure, respectively. As always, unknown degrees of freedom v_I, $I \in \mathcal{F}$ are collected into a velocity vector of unknowns $\mathbf{y} \in \mathbb{R}^{|\mathcal{F}|}$, Dirichlet degrees of freedom d_I, $I \in \mathcal{D}$ are collected into a Dirichlet velocity vector $\mathbf{y}^{\mathcal{D}} \in \mathbb{R}^{|\mathcal{D}|}$, and all pressure degrees of freedom are collected into a pressure vector $\mathbf{z} \in \mathbb{R}^m$. Upon testing the weak form (4.35–4.36) with the bases of $\mathcal{V}_{0,h}$ and $\mathcal{Q}_h$, we obtain the following linear system:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{L} - \mathbf{A}^{\mathcal{D}} \mathbf{y}^{\mathcal{D}} \\ \mathbf{0} \end{pmatrix}. \tag{4.46}$$

Here, $\mathbf{A}$ is a $|\mathcal{F}| \times |\mathcal{F}|$ *velocity matrix*, $\mathbf{B}$ is a $m \times |\mathcal{F}|$ *coupling matrix*, $\mathbf{A}^{\mathcal{D}}$ is a $|\mathcal{F}| \times |\mathcal{D}|$ elimination matrix, and $\mathbf{L}$ is a loading vector with $|\mathcal{F}|$ elements. Their elements are computed as

$$A_{IJ} = a(\mathbf{N}_I, \mathbf{N}_J), \qquad B_{iJ} = b(\mathbf{N}_J, M_i) \qquad \text{and} \qquad L_I = l(\mathbf{N}_I). \tag{4.47}$$

The block-matrix in (4.46), which I denote by $\mathbf{K}$ and refer to as the *saddle-point matrix*, inherits the saddle-points structure from the weak form (4.35–4.36). Since the bilinear form $a(\mathbf{v}, \mathbf{w})$ is not symmetric, neither are matrices $\mathbf{A}$ and $\mathbf{K}$. Moreover, $\mathbf{K}$ is indefinite. All these properties of the linear system (4.46) make it challenging to solve numerically. Most of general iterative linear solvers perform poorly when applied to (4.46); only iterative solvers and preconditioners designed specifically for saddle-point problems can be efficient [76]. A good alternative is to use a direct linear solver if the size of the problem permits. In this work, I have successfully applied the PardisoLU linear solver. In comparison to standard direct and iterative solvers from Eigen[4], I have achieved relative speed-ups of up to ten times.

Steady-state incompressible Navier-Stokes equations

After studying isogeometric discretization of the Oseen equations, we are ready to tackle the incompressible Navier-Stokes equations. In order to focus on the spacial discretization, it is convenient to consider first a *steady-state flow* case. If all data in the INSE equations (4.26–4.27) do not depend on time and if the Reynolds number Re is sufficiently low, the equations yield a steady-state solution. This solution can be found by solving the *steady-state Navier-Stokes equations*

$$-\rho\nu\Delta_{\mathbf{x}}\mathbf{v} + \rho\nabla_{\mathbf{x}}\mathbf{v} \cdot \mathbf{v} + \nabla_{\mathbf{x}} p = \rho\mathbf{g} \text{ in } \Omega, \tag{4.48}$$

$$\operatorname{div}_{\mathbf{x}} \mathbf{v} = 0 \text{ in } \Omega. \tag{4.49}$$

In comparison to the time-dependent Navier-Stokes equations (4.26–4.27), equations (4.48–4.49) do not include the acceleration term. As shown in [74], equations (4.48–4.49) have a unique solution, granted that Re is low enough. On the contrary, the solution becomes unstable for high values of Re and should be computed instead using the time-dependent Navier-Stokes equations.

Using the same function spaces $\mathcal{V}$, $\mathcal{V}_0$ and $\mathcal{Q}$ as for the Oseen equations, we can write the weak form of the steady-state Navier-Stokes equations (4.48–4.49) in the following way:

[4] A C++ linear algebra library which G+Smo is based on.

find $(\mathbf{v}, p) \in \mathcal{V} \times \mathcal{Q}$ such that

$$\int_\Omega \Big(\rho\nu\nabla_{\mathbf{x}}\mathbf{v} : \nabla_{\mathbf{x}}\mathbf{w} + \rho(\nabla_{\mathbf{x}}\mathbf{v}\cdot\mathbf{v})^T\mathbf{w} + p\,\mathrm{div}_{\mathbf{x}}\,\mathbf{w}\Big)d\mathbf{x} = \int_\Omega \rho\mathbf{g}^T\mathbf{w}d\mathbf{x} \quad \forall\mathbf{w}\in\mathcal{V}_0, \tag{4.50}$$

$$\int_\Omega \mathrm{div}_{\mathbf{x}}\,\mathbf{v}qd\mathbf{x} = 0 \quad \forall q\in\mathcal{Q}. \tag{4.51}$$

Let us define the *diffusion bilinear form* $d(\mathbf{v},\mathbf{w}) = \int_\Omega \rho\nu\nabla_{\mathbf{x}}\mathbf{v} : \nabla_{\mathbf{x}}\mathbf{w}d\mathbf{x}$ and the *advection trilinear form* $n(\mathbf{v},\mathbf{u},\mathbf{w}) = \int_\Omega \rho(\nabla_{\mathbf{x}}\mathbf{v}\cdot\mathbf{u})^T\mathbf{w}d\mathbf{x}$. Then we can rewrite the variational problem (4.50–4.51) in the following form:

find $(\mathbf{v}, p) \in \mathcal{V} \times \mathcal{Q}$ such that

$$d(\mathbf{v},\mathbf{w}) + n(\mathbf{v},\mathbf{v},\mathbf{w}) + b(\mathbf{w},p) = l(\mathbf{w}) \quad \forall\mathbf{w}\in\mathcal{V}_0, \tag{4.52}$$

$$b(\mathbf{v},q) = 0 \quad \forall q\in\mathcal{Q}. \tag{4.53}$$

Here, the bilinear form $b(\mathbf{w},q)$ and the linear form $l(\mathbf{w})$ are defined in the same way as for the Oseen equations. Problem (4.52–4.53) can be called a nonlinear saddle-point problem. In order to solve it, one has to linearize the nonlinear advection form $n(\mathbf{v},\mathbf{v},\mathbf{w})$. In what follows, I describe two common methods to do that: the Picard iteration and Newton's method.

Picard iteration

The *Picard iteration* is an iterative method for solving the weak formulation of the steady-state Navier-Stokes equations (4.52–4.53) that overcomes the nonlinearity in the advection term $n(\mathbf{v},\mathbf{v},\mathbf{w})$ by "freezing" the advective velocity. Concretely, let $(\mathbf{v}^s, p^s)$ be a solution at the s-th iteration. Then the next solution $(\mathbf{v}^{s+1}, p^{s+1})$ is computed by solving the following linear problem:

find $(\mathbf{v}^{s+1}, p^{s+1}) \in \mathcal{V} \times \mathcal{Q}$ such that

$$d(\mathbf{v}^{s+1},\mathbf{w}) + n(\mathbf{v}^{s+1},\mathbf{v}^s,\mathbf{w}) + b(\mathbf{w},p^{s+1}) = l(\mathbf{w}) \quad \forall\mathbf{w}\in\mathcal{V}_0, \tag{4.54}$$

$$b(\mathbf{v}^{s+1},q) = 0 \quad \forall q\in\mathcal{Q}. \tag{4.55}$$

Problem (4.54–4.55) is an Oseen-type problem with an advective velocity $\mathbf{b} = \mathbf{v}^s$ and a zero reaction term. Since the assumptions for the existence and uniqueness theorem are satisfied, the problem has a unique solution. By defining a bilinear form $a_P(\mathbf{v}^{s+1},\mathbf{w}) = d(\mathbf{v}^{s+1},\mathbf{w}) + n(\mathbf{v}^{s+1},\mathbf{v}^s,\mathbf{w})$, we can cast the problem (4.54–4.55) into an abstract saddle-point problem form (4.37–4.38). The ensuing discretization with mixed isogeometric elements and matrix assembly happen in the same way as for the Oseen equations. The Picard iteration can be initialized with $\mathbf{v}^0 = \mathbf{0}$. Note that the velocity matrix $\mathbf{A}_P$ corresponding to the discretization of a_P depends on the current solution $\mathbf{v}_s$ and has to be reassembled at every iteration. The Picard iteration continues until convergence.

From the perspective of the general theory of iterative schemes for nonlinear problems, the Picard iteration is a simple fixed-point iteration method. As such, it has a larger convergence radius than Newton's method but converges slower to the solution because it is a first-order method.

Newton's method for steady-state problems

Application of Newton's method to the weak formulation of the steady-state Navier-Stokes equations (4.52–4.53) follows the same steps as for the stationary equations of nonlinear elasticity described

in Section 3.3.4. Let us define the residual $\mathbf{R}(\mathbf{v}, p, \mathbf{w}, q)$ as

$$\mathbf{R}(\mathbf{v}, p, \mathbf{w}, q) = \begin{pmatrix} d(\mathbf{v}, \mathbf{w}) + n(\mathbf{v}, \mathbf{v}, \mathbf{w}) + b(\mathbf{w}, p) - l(\mathbf{w}) \\ b(\mathbf{v}, q) \end{pmatrix}. \tag{4.56}$$

The Taylor expansion at $\mathbf{R}(\mathbf{v}, p, \mathbf{w}, q)$ with the *velocity increment* $\Delta\mathbf{v}$ and *pressure increment* Δp yields

$$\mathbf{R}(\mathbf{v} + \Delta\mathbf{v}, p + \Delta p, \mathbf{w}, q) = \mathbf{R}(\mathbf{v}, p, \mathbf{w}, q) + D\mathbf{R}(\mathbf{v}, p, \mathbf{w}, q) \cdot \begin{pmatrix} \Delta\mathbf{v} \\ \Delta p \end{pmatrix} + \mathbf{o}(||\Delta\mathbf{v}||, ||\Delta p||), \tag{4.57}$$

where the directional derivative is computed as

$$D\mathbf{R}(\mathbf{v}, p, \mathbf{w}, q) \cdot \begin{pmatrix} \Delta\mathbf{v} \\ \Delta p \end{pmatrix} = \begin{pmatrix} d(\Delta\mathbf{v}, \mathbf{w}) + n(\Delta\mathbf{v}, \mathbf{v}, \mathbf{w}) + n(\mathbf{v}, \Delta\mathbf{v}, \mathbf{w}) + b(\mathbf{w}, \Delta p) \\ b(\Delta\mathbf{v}, p) \end{pmatrix}. \tag{4.58}$$

Then Newton's method for a nonlinear problem $\mathbf{R}(\mathbf{v}, p, \mathbf{w}, q) = \mathbf{0}$ assumes the following from: given an intermediate solution $(\mathbf{v}^s, p^s) \in \mathcal{V} \times \mathcal{Q}$, find an update $(\Delta\mathbf{v}^{s+1}, \Delta p^{s+1}) \in \mathcal{V}_0 \times \mathcal{Q}$ such that

$$\begin{aligned} d(\Delta\mathbf{v}^{s+1}, \mathbf{w}) + n(\Delta\mathbf{v}^{s+1}, \mathbf{v}^s, \mathbf{w}) + n(\mathbf{v}^s, \Delta\mathbf{v}^{s+1}, \mathbf{w}) + b(\mathbf{w}, \Delta p^{s+1}) &= l(\mathbf{w}) - \\ -d(\mathbf{v}^s, \mathbf{w}) - n(\mathbf{v}^s, \mathbf{v}^s, \mathbf{w}) - b(\mathbf{w}, p^s) \ \ \forall \mathbf{w} \in \mathcal{V}_0, & \quad (4.59) \\ b(\Delta\mathbf{v}^{i+s}, q) = -b(\mathbf{v}^s, q) \ \ \forall q \in \mathcal{Q}. & \quad (4.60) \end{aligned}$$

After that, the next solution $(\mathbf{v}^{s+1}, p^{s+1}) \in \mathcal{V} \times \mathcal{Q}$ is defined as $(\mathbf{v}^s + \Delta\mathbf{v}^{s+1}, p^s + \Delta p^{s+1})$. This *update form* of Newton's method can be simplified by moving all terms with $\mathbf{v}^s$ from the right-hand side to the left, adding $n(\mathbf{v}^s, \mathbf{v}^s, \mathbf{w})$ to each side of equation (4.59) and exploiting the linearity of all the forms. After doing that, one obtains a *next-solution form* of Newton's method: given $(\mathbf{v}^s, p^s) \in \mathcal{V} \times \mathcal{Q}$, find the next solution $(\mathbf{v}^{s+1}, p^{s+1}) \in \mathcal{V} \times \mathcal{Q}$ such that

$$\begin{aligned} d(\mathbf{v}^{s+1}, \mathbf{w}) + n(\mathbf{v}^{s+1}, \mathbf{v}^s, \mathbf{w}) + n(\mathbf{v}^s, \mathbf{v}^{s+1}, \mathbf{w}) + b(\mathbf{w}, p^{s+1}) &= l(\mathbf{w}) + \\ + n(\mathbf{v}^s, \mathbf{v}^s, \mathbf{w}) \ \ \forall \mathbf{w} \in \mathcal{V}_0, & \quad (4.61) \\ b(\mathbf{v}^{s+1}, q) = 0 \ \ \forall q \in \mathcal{Q}. & \quad (4.62) \end{aligned}$$

The next-solution form of Newton's method is an Oseen-type problem with an advective velocity $\mathbf{b} = \mathbf{v}^s$ and a tensor-valued reaction $c = \nabla\mathbf{v}^s$. Since $\nabla\mathbf{v}_s$ is not necessarily nonnegative, the existence and uniqueness theorem for the Oseen equations cannot be applied. Nevertheless, by defining a bilinear form $a_N(\mathbf{v}^{s+1}, \mathbf{w}) = d(\mathbf{v}^{s+1}, \mathbf{w}) + n(\mathbf{v}^{s+1}, \mathbf{v}^s, \mathbf{w}) + n(\mathbf{v}^s, \mathbf{v}^{s+1}, \mathbf{w})$ and a linear form $l_N(\mathbf{w}) = l(\mathbf{w}) + n(\mathbf{v}^s, \mathbf{v}^s, \mathbf{w})$, we can cast the variational problem (4.61–4.62) into an abstract saddle-point problem form (4.37–4.38) and then discretize it using mixed isogeometric elements. Note that the variational problem (4.61–4.62) yields a velocity field from $\mathcal{V}$, which means that the Dirichlet vector $\mathbf{y}^{\mathcal{D}}$ is not zero and has to be eliminated from the system. The velocity matrix $\mathbf{A}_N$ corresponding to the discretization of a_N and the loading vector $\mathbf{L}_N$ corresponding to l_N depend on the current solution $\mathbf{v}_s$ and, therefore, have to be reassembled at every iteration. Iterations of Newton's method continue until predefined convergence criteria are met.

As mentioned above, Newton's method has a smaller convergence radius, but converges to the solution faster than the Picard iteration. This fact does not antagonize the two methods but rather makes them perfectly complementary to each other. Concretely, if Newton's method struggles to converge from an initial guess $\mathbf{v}^0 = \mathbf{0}$, a few steps of the Picard iteration can be performed first. This should provide an approximate solution lying in the convergence radius of Newton's method, from where Newton's method is used to finish the solution process.

4.3.3 Time integration

Let us come back to the time-dependent Navier-Stokes equations (4.26–4.27). Using the function spaces $\mathcal{V}$, $\mathcal{V}_0$ and $\mathcal{Q}$, we can rewrite them in a weak form:

$$\begin{aligned}
&\text{find } (\mathbf{v}(\cdot,t), p(\cdot,t)) \in \mathcal{V} \times \mathcal{Q} \text{ such that} \\
&\int_\Omega \Big(\rho \dot{\mathbf{v}}^T \mathbf{w} + \rho\nu \nabla_{\mathbf{x}}\mathbf{v} : \nabla_{\mathbf{x}}\mathbf{w} + \rho(\nabla_{\mathbf{x}}\mathbf{v}\cdot\mathbf{v})^T\mathbf{w} + p\,\mathrm{div}_{\mathbf{x}}\,\mathbf{w}\Big)d\mathbf{x} = \int_\Omega \rho \mathbf{g}^T\mathbf{w} d\mathbf{x} \;\; \forall \mathbf{w} \in \mathcal{V}_0, \qquad (4.63)\\
&\int_\Omega \mathrm{div}_{\mathbf{x}}\,\mathbf{v} q d\mathbf{x} = 0 \;\; \forall q \in \mathcal{Q}. \qquad (4.64)
\end{aligned}$$

After defining an *acceleration bilinear form* $m(\dot{\mathbf{v}}, \mathbf{w}) = \int_\Omega \rho \dot{\mathbf{v}}^T \mathbf{w} d\mathbf{x}$, we can cast the variational problem (4.63–4.64) into a more concise form:

$$\begin{aligned}
&\text{find } (\mathbf{v}(\cdot,t), p(\cdot,t)) \in \mathcal{V} \times \mathcal{Q} \text{ such that} \\
&m(\dot{\mathbf{v}}, \mathbf{w}) + d(\mathbf{v}, \mathbf{w}) + n(\mathbf{v}, \mathbf{v}, \mathbf{w}) + b(\mathbf{w}, p) = l(\mathbf{w}) \;\; \forall \mathbf{w} \in \mathcal{V}_0, \qquad (4.65)\\
&b(\mathbf{v}, q) = 0 \;\; \forall q \in \mathcal{Q}. \qquad (4.66)
\end{aligned}$$

In order to solve the variational problem (4.65–4.66), I first discretize it in time, then in space. For the time discretization, I use a *one-step θ-scheme*: given the solution $(\mathbf{v}_i, p_i) \in \mathcal{V} \times \mathcal{Q}$ at time t_i, find the next solution $(\mathbf{v}_{i+1}, p_{i+1}) \in \mathcal{V} \times \mathcal{Q}$ by solving the following variational problem:

$$\begin{aligned}
&\frac{m(\mathbf{v}_{i+1}, \mathbf{w}) - m(\mathbf{v}_i, \mathbf{w})}{\Delta t} + \theta\Big(d(\mathbf{v}_{i+1}, \mathbf{w}) + n(\mathbf{v}_{i+1}, \mathbf{v}_{i+1}, \mathbf{w})\Big) + \\
&+(1-\theta)\Big(d(\mathbf{v}_i, \mathbf{w}) + n(\mathbf{v}_i, \mathbf{v}_i, \mathbf{w})\Big) + b(\mathbf{w}, p_{i+1}) = \theta l_{i+1}(\mathbf{w}) + (1-\theta) l_i(\mathbf{w}) \;\; \forall \mathbf{w} \in \mathcal{V}_0, \qquad (4.67)\\
&b(\mathbf{v}_{i+1}, q) = 0 \;\; \forall q \in \mathcal{Q}. \qquad (4.68)
\end{aligned}$$

The parameter $\theta \in [0,1]$ defines the behavior of the method. With value $\theta = 0$, (4.67–4.68) becomes the *explicit Euler scheme*; although the corresponding system (4.67–4.68) is linear, the explicit Euler scheme requires very small time steps, lest the system becomes unstable. For any other value of θ, the system (4.67–4.68) is nonlinear due to the advection term $n(\mathbf{v}_{i+1}, \mathbf{v}_{i+1}, \mathbf{w})$. Value $\theta = 1$ corresponds to the *implicit Euler scheme* which is known to be unconditionally stable, so time steps of any size can be used. A disadvantage of the implicit Euler scheme is that it is a first-order time-discretization scheme which can inaccurate and too dissipative. Often, an optimal choice is $\theta = 0.5$, which results in the second-order *Crank-Nicolson scheme*.

Note that the θ-scheme is applied only to velocity $\mathbf{v}$ in equation (4.67). In contrast to that, the bilinear form $b(\mathbf{w}, p)$ is always evaluated at time t_{i+1}, that is, it enters the equations fully implicitly. This guarantees that the solution of (4.67–4.68) is divergence-free. Additionally, observe that the pressure p_i at time t_i is not used in equations (4.67–4.68), which is a manifestation of the DAE nature of the incompressible Navier-Stokes equations. One can say that pressure has no history.

Newton's method for time-dependent problems

If $\theta \neq 0$ is chosen, one has to solve a nonlinear problem (4.67–4.68) at every time step. To that end, Newton's method can be used. For the sake of brevity, I omit the derivation and directly formulate the next-solution form of Newton's method for (4.67–4.68). First, solution $(\mathbf{v}_i, p_i) \in \mathcal{V} \times \mathcal{Q}$ at time t_i is used as an initial guess $(\mathbf{v}^0_{i+1}, p^0_{i+1}) \in \mathcal{V} \times \mathcal{Q}$ for the solution at time t_{i+1}. Then, given an

intermediate solution $(\mathbf{v}_{i+1}^s, p_{i+1}^s) \in \mathcal{V} \times \mathcal{Q}$ at time t_{i+1}, the next solution $(\mathbf{v}_{i+1}^{s+1}, p_{i+1}^{s+1}) \in \mathcal{V} \times \mathcal{Q}$ is computed by solving the linear system

$$\begin{aligned} m(\mathbf{v}_{i+1}^{s+1}, \mathbf{w}) + \Delta t \Big[\theta \Big(d(\mathbf{v}_{i+1}^{s+1}, \mathbf{w}) + n(\mathbf{v}_{i+1}^{s+1}, \mathbf{v}_{i+1}^{s}, \mathbf{w}) + n(\mathbf{v}_{i+1}^{s}, \mathbf{v}_{i+1}^{s+1}, \mathbf{w}) \Big) + \\ + b(\mathbf{w}, p_{i+1}^{s+1}) \Big] = m(\mathbf{v}_i, \mathbf{w}) + \Delta t \Big[\theta \Big(l_{i+1}(\mathbf{w}) + n(\mathbf{v}_{i+1}^{s}, \mathbf{v}_{i+1}^{s}, \mathbf{w}) \Big) + \\ + (1-\theta) \Big(l_i(\mathbf{w}) - d(\mathbf{v}_i, \mathbf{w}) - n(\mathbf{v}_i, \mathbf{v}_i, \mathbf{w}) \Big) \Big] \quad \forall \mathbf{w} \in \mathcal{V}_0, \end{aligned} \tag{4.69}$$

$$\Delta t b(\mathbf{v}_{i+1}^{s+1}, q) = 0 \quad \forall q \in \mathcal{Q}. \tag{4.70}$$

Here, equation (4.70) is multiplied by Δt for the sake of symmetry. I refer to (4.69–4.70) as the *Newton scheme*. Iterations of the Newton scheme at each time step continue until the convergence criteria are met. If the time step size Δt is small, solution $(\mathbf{v}_i, p_i) \in \mathcal{V} \times \mathcal{Q}$ at time t_i provides a good initial guess for the next moment of time, so the Newton scheme can converge fast. Still, it requires at least 3–4 iterations per time step which is quiet numerically expensive.

Weak problem (4.69–4.70) is discretized using mixed isogeometric elements. The result is the following matrix equation:

$$\begin{pmatrix} \mathbf{M} + \Delta t \theta \mathbf{A}_N(\mathbf{y}_{i+1}^s, \mathbf{y}_{i+1}^D) & \Delta t \mathbf{B}^T \\ \Delta t \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{y}_{i+1}^{s+1} \\ \mathbf{z}_{i+1}^{s+1} \end{pmatrix} = \begin{pmatrix} * \\ \mathbf{0} \end{pmatrix}, \tag{4.71}$$

where

$$\begin{aligned} * = \mathbf{M}\mathbf{y}_i + \Delta t \Big[(1-\theta) \Big(\mathbf{L}_N(\mathbf{y}_i, \mathbf{y}_i^D) - \mathbf{A}_N(\mathbf{y}_i, \mathbf{y}_i^D)\mathbf{s}_i \Big) + \theta \mathbf{L}_N(\mathbf{y}_{i+1}^s, \mathbf{y}_{i+1}^D) - \\ - \mathbf{A}_N^{\mathcal{D}}(\mathbf{y}_{i+1}^s, \mathbf{y}_{i+1}^D)\mathbf{y}_{i+1}^{\mathcal{D}} \Big] - \mathbf{M}^{\mathcal{D}}(\mathbf{y}_{i+1}^{\mathcal{D}} - \mathbf{y}_i^{\mathcal{D}}). \end{aligned} \tag{4.72}$$

Let us examine the saddle-point problem (4.71). Here, $\mathbf{M} \in \mathbb{R}^{|\mathcal{F}| \times |\mathcal{F}|}$ is a mass matrix (4.6), and $\mathbf{M}^{\mathcal{D}} \in \mathbb{R}^{|\mathcal{F}| \times |\mathcal{D}|}$ is a corresponding *mass elimination matrix*. Equations (4.71–4.72) reuse the velocity matrix $\mathbf{A}_N$ and the loading vector $\mathbf{L}_N$ from the next-solution form of Newton's method for the steady-state Navier-Stokes equations. In order to underline that $\mathbf{A}_N$ and $\mathbf{L}_N$ are assembled using a particular velocity field $\mathbf{v}_*^*$, I include the corresponding vector of unknowns $\mathbf{y}_*^*$ and the Dirichlet vector $\mathbf{y}_*^{\mathcal{D}}$ into the formulas. Finally, I account for the possibility of time-dependent Dirichlet boundary conditions. To that end, I include the last term in equation (4.72) which eliminates the *acceleration-Dirichlet degrees of freedom*. [77]. Correct treatment of time-dependent Dirichlet boundary conditions is crucial in FSI simulations.

IMEX schemes

Applying Newton's method at each time step can be rather numerically expensive. To avoid that, one can use the so-called *implicit-explicit schemes* (IMEX). IMEX schemes replace the nonlinear advection term $n(\mathbf{v}_{i+1}, \mathbf{v}_{i+1}, \mathbf{w})$ in (4.67) by a linear term $n(\mathbf{v}_{i+1}, \mathbf{v}_*, \mathbf{w})$, where the *surrogate advective velocity* $\mathbf{v}_*$ is constructed using velocity at time t_i or earlier. Then, given the solution $(\mathbf{v}_i, p_i) \in \mathcal{V} \times \mathcal{Q}$ at time t_i, the solution $(\mathbf{v}_{i+1}, p_{i+1}) \in \mathcal{V} \times \mathcal{Q}$ at time t_{i+1} is found by solving the linear system

$$\begin{aligned} m(\mathbf{v}_{i+1}, \mathbf{w}) + \Delta t \Big[\theta \Big(d(\mathbf{v}_{i+1}, \mathbf{w}) + n(\mathbf{v}_{i+1}, \mathbf{v}_*, \mathbf{w}) \Big) + b(\mathbf{w}, p_{i+1}) \Big] = m(\mathbf{v}_i, \mathbf{w}) + \\ + \Delta t \Big[\theta l_{i+1}(\mathbf{w}) + (1-\theta) \Big(l_i(\mathbf{w}) - d(\mathbf{v}_i, \mathbf{w}) - n(\mathbf{v}_i, \mathbf{v}_i, \mathbf{w}) \Big) \Big] \quad \forall \mathbf{w} \in \mathcal{V}_0, \end{aligned} \tag{4.73}$$

$$\Delta t b(\mathbf{v}_{i+1}, q) = 0 \quad \forall q \in \mathcal{Q}. \tag{4.74}$$

Here, equation (4.74) is again multiplied by Δt for symmetry. In comparison to the Newton method, IMEX schemes only require a solution of one linear system per time step. The price to pay is that IMEX schemes usually have to use a smaller time step to achieve the accuracy of Newton's method.

The simplest option for the surrogate advective velocity is $\mathbf{v}_* = \mathbf{v}_i$. In this work, however, I use a linear extrapolation of the two previous time steps:

$$\mathbf{v}_* = \mathbf{v}_i + \frac{\Delta t_{i+1}}{\Delta t_i}(\mathbf{v}_i - \mathbf{v}_{i-1}) = 2\mathbf{v}_i - \mathbf{v}_{i-1} \ \ (\text{if } \Delta t_{i+1} = \Delta t_i). \tag{4.75}$$

The system (4.73–4.74) is discretized using mixed isogeometric elements. The result is the matrix equation

$$\begin{pmatrix} \mathbf{M} + \Delta t\theta \mathbf{A}_P(\mathbf{y}_{i+1}, \mathbf{y}^D_{i+1}) & \Delta t \mathbf{B}^T \\ \Delta t \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{y}_{i+1} \\ \mathbf{z}_{i+1} \end{pmatrix} = \begin{pmatrix} * \\ \mathbf{0} \end{pmatrix}, \tag{4.76}$$

where

$$\begin{aligned} * = \mathbf{M}\mathbf{y}_i + \Delta t \Big[(1-\theta)\Big(\mathbf{L} - \mathbf{A}_P(\mathbf{y}_i, \mathbf{y}^D_i)\mathbf{s}_i\Big) + \theta \mathbf{L} - \mathbf{A}^{\mathcal{D}}_P(\mathbf{y}_{i+1}, \mathbf{y}^D_{i+1})\mathbf{y}^{\mathcal{D}}_{i+1} \Big] - \\ - \mathbf{M}^{\mathcal{D}}(\mathbf{y}^{\mathcal{D}}_{i+1} - \mathbf{y}^{\mathcal{D}}_i). \end{aligned} \tag{4.77}$$

Equation (4.76) has the same structure as (4.71). The sole exception is that the velocity matrix $\mathbf{A}_P$ from the Picard's iteration for the steady-state Navier-Stokes equations is used instead of $\mathbf{A}_N$, and the loading vector $\mathbf{L}$ (4.46) from the Oseen equations replaces $\mathbf{L}_N$.

4.3.4 Partial benchmark CFD1: stationary flow

I validate my implementation of the incompressible Navier-Stokes solver on partial benchmarks CFD1 and CFD3 from [38]. The first benchmark, CFD1, is a steady-state flow test in the flow channel Ω from Section 4.1. It involves solving the steady-state Navier-Stokes equations

$$-\rho\nu\Delta_{\mathbf{x}}\mathbf{v} + \rho\nabla_{\mathbf{x}}\mathbf{v}\cdot\mathbf{v} + \nabla_{\mathbf{x}}p = \rho\mathbf{g} \text{ in } \Omega, \tag{4.78}$$

$$\operatorname{div}_{\mathbf{x}}\mathbf{v} = 0 \text{ in } \Omega \tag{4.79}$$

with the following boundary conditions:

$$\mathbf{v} = \mathbf{0} \text{ on } \partial\Omega_{ns}, \quad \mathbf{v} = \mathbf{v}_{in} \text{ on } \partial\Omega_{in} \quad \text{and} \quad \boldsymbol{\sigma}(\mathbf{v}, p)\cdot\mathbf{n} = \mathbf{0} \text{ on } \partial\Omega_{out}. \tag{4.80}$$

Here, $\partial\Omega_{in}$ is the left wall of flow channel, $\partial\Omega_{out}$ is the right wall of flow channel, and $\partial\Omega_{ns}$ is the rest of the fluid domain boundary, including the boundary of the submerged structure. The *inflow velocity profile* $\mathbf{v}_{in}$ is specified by

$$\mathbf{v}_{in} = \begin{pmatrix} v_{\max}\big(\frac{2}{0.41}\big)^2 y(0.41-y) \\ 0 \end{pmatrix}, \tag{4.81}$$

where $v_{\max} \in \mathbb{R}$ is the *maximum inflow velocity*.

For the geometry, I use a quadratic multi-patch NURBS parametrization consisting of seven patches. The corresponding NURBS space is refined for the analysis using uniform h-refinement and degree elevation. Additionally, I apply h-refinement twice around the submerged solid in the direction perpendicular to the boundary[5]. The thickness of the refined *boundary layer* amounts to

[5]This additional refinement is necessary to better resolve sharp velocity gradients appearing in the boundary layer because of the no-slip condition. Since the gradient is directed orthogonally to the boundary, it is enough to apply refinement only in one direction. Of course, this is not a true local refinement. Still, since the additional isoparametric lines close on themselves, the non-local nature of such refinement is not a problem.

20% of the corresponding parameter direction. Figure 4.4 shows an example of the corresponding isogeometric mesh. For the simulation, the following parameters are used: density $\rho = 10^3$ kg·m^{-3}, kinematic viscosity $\nu = 10^{-3}$ m^2·s^{-1}, gravitational acceleration $\mathbf{g} = (0,0)^T$ m·s^{-2} and maximum inflow velocity $v_{\max} = 0.3$ m·s^{-1}. Such parameter values translate into the Reynolds number

$$Re = \frac{2v_{\max}/3 \cdot 0.1}{\nu} = 20. \tag{4.82}$$

Here, an average inflow velocity $2v_{\max}/3$ is a characteristic velocity, and diameter 0.01 of the rigid disc is a characteristic length. Re=20 is a relatively low value, so one can expect a steady-state flow. A quantity of interest is the net force exerted by the fluid on the submerged structure, including both the rigid disc and the elastic beam. Let us denote the boundary of the submerged solid by $\partial\Sigma_0$. Then the net fluid force can be computed as

$$(F_D, F_L)^T = \int_{\partial\Sigma_0} \boldsymbol{\sigma}(\mathbf{v}, p) \cdot \mathbf{n} d\mathbf{x}. \tag{4.83}$$

The horizontal component F_D is called the *drag*, whereas the vertical component F_L is called the *lift*.

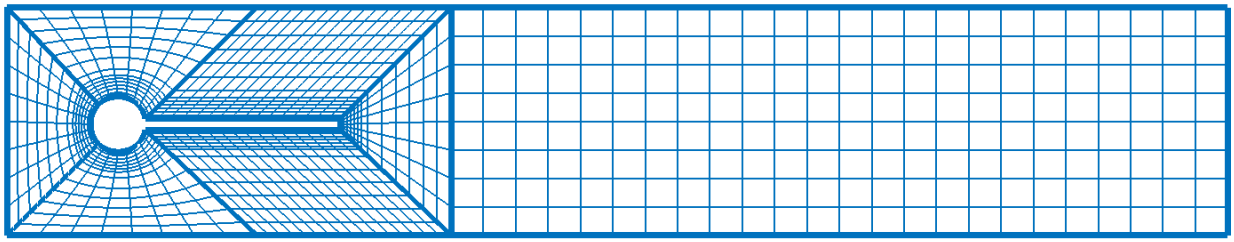

Figure 4.4. Multi-patch structure of the fluid domain and the isogeometric mesh corresponding to the third level of uniform h-refinement and the second level of boundary layer refinement.

The equations have been solved using Newton's method, and the solution fields are illustrated in Figure 4.5. The simulation results are summarized in Table 4.3, which includes the mixed element type, the number of degrees of freedom, the computational time and the resulting drag and lift. Overall, the results seem to be very close to the reference values. Unfortunately, I was not able to continue refining the analysis due to memory limitations. The results show that the subgrid elements, although resulting in the same number of degrees of freedom as the Taylor-Hood elements, tend to require about 25% more computational time. Moreover, the accuracy of the corresponding solution is marginally smaller. In light of that, the Taylor-Hood elements should be preferred over the subgrid elements if possible.

Element	#DoF	T_{comp}	F_D [N]	F_L [N]
TH $\mathcal{N}_4^3/\mathcal{N}_4^2$	36972	7.9s	14.3193	1.11969
SG $\mathcal{N}_5^2/\mathcal{N}_4^2$	36972	9.4s	14.3257	1.11983
TH $\mathcal{N}_5^3/\mathcal{N}_5^2$	145354	33s	14.3096	1.11942
SG $\mathcal{N}_6^2/\mathcal{N}_5^2$	145354	41s	14.3137	1.11949
TH $\mathcal{N}_6^3/\mathcal{N}_6^2$	590304	2m26s	14.3038	1.11927
SG $\mathcal{N}_7^2/\mathcal{N}_6^2$	590304	3m3s	14.3065	1.11930
Ref	11225600	N/A	14.2929	1.11905

Table 4.3. Benchmark CFD1: simulation results.

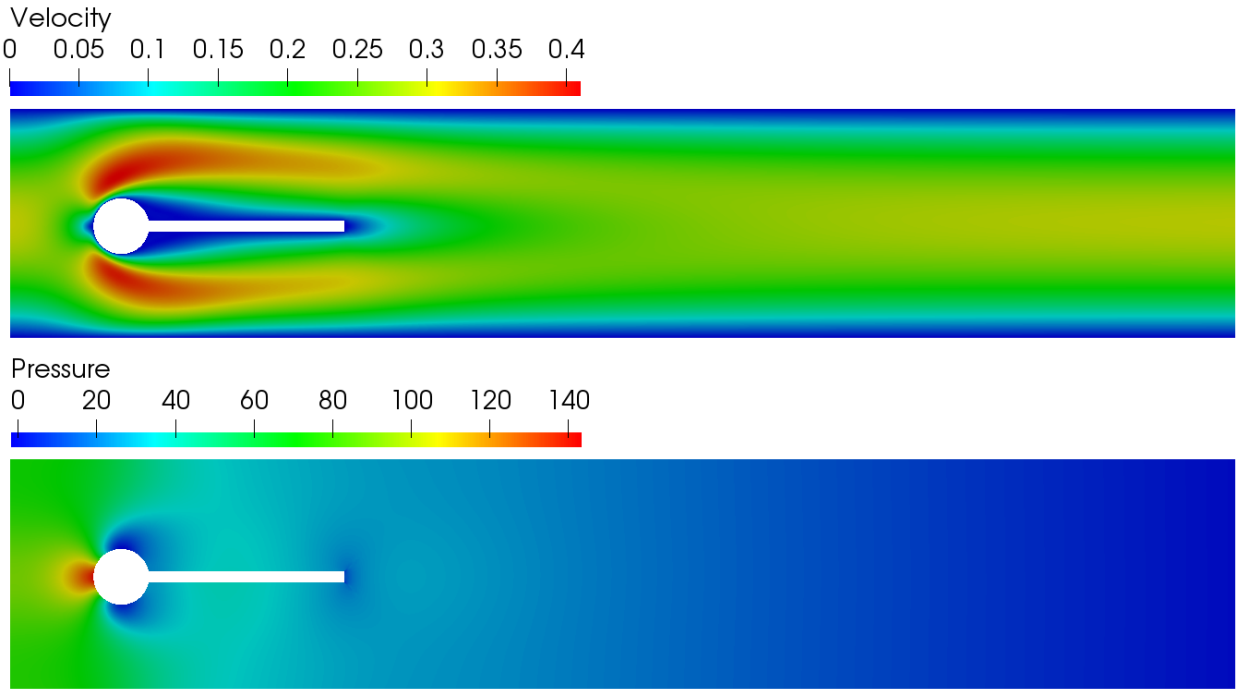

Figure 4.5. Benchmark CFD1: steady-state velocity and pressure fields.

4.3.5 Partial benchmark CFD3: vortex shedding

Benchmark CFD3 is a simulation of a time-dependent flow through the channel Ω. It involves solving the time-dependent Navier-Stokes equations

$$\rho\dot{\mathbf{v}} + \rho\nabla_{\mathbf{x}}\mathbf{v}\cdot\mathbf{v} = \rho\nu\Delta_{\mathbf{x}}\mathbf{v} - \nabla_{\mathbf{x}}p + \rho\mathbf{g} \text{ in } \Omega\times[0,T], \tag{4.84}$$

$$\operatorname{div}_{\mathbf{x}}\mathbf{v} = 0 \text{ in } \Omega\times[0,T], \tag{4.85}$$

for $T = 10$s with boundary conditions (4.80) and the initial conditions

$$\mathbf{v}(\cdot,0) = \mathbf{0} \text{ in } \Omega. \tag{4.86}$$

In order to comply with the initial conditions and to implement a smooth *warm-up phase*, the inflow velocity profile $\mathbf{v}_{in}$ (4.81) is made time-dependent:

$$\mathbf{v}_{in}(t) = \begin{cases} \mathbf{v}_{in}\frac{1-\cos(\pi t/2)}{2} & \text{if } t < 2\text{s}, \\ \mathbf{v}_{in} & \text{if } t \geqslant 2\text{s}. \end{cases} \tag{4.87}$$

I use the same geometry parametrization as in CFD1 but apply boundary layer refinement only once. Despite having shown that the Taylor-Hood elements are superior with respect to both accuracy and numerical cost, I use the subgrid elements in this simulation. This choice is motivated by the fact that, for some reason, the TH elements result in an unstable behavior when combined with direct linear solver PardisoLU. On the other hand, the SG elements combined with PardisoLU show no signs of unstable behavior which allows me to keep a relative speed-up of x5 when compared to linear solvers from the Eigen library.

For temporal discretization, I use both the Newton and IMEX schemes with $\theta = 0.5$ and time step size Δt between 0.1s and 0.0025s. I use the same parameter values for density ρ, kinematic

viscosity ν and gravitational acceleration $\mathbf{g}$ as in CFD1. For the maximum inflow velocity, I use a value $v_{\max} = 3\ \mathrm{m{\cdot}s^{-1}}$, which corresponds to $Re = 200$. With such a relatively high Reynolds number, the flow becomes unstable, and the intentional asymmetry of the geometry promotes development of the effect called *vortex shedding*. Vortex shedding is an oscillating flow occurring past a bluff body which results in oscillating drag and lift forces, see Figures 4.6 and 4.7. In this simulation, the quantities of interest are the mean, the amplitude and the frequency of both drag and lift.

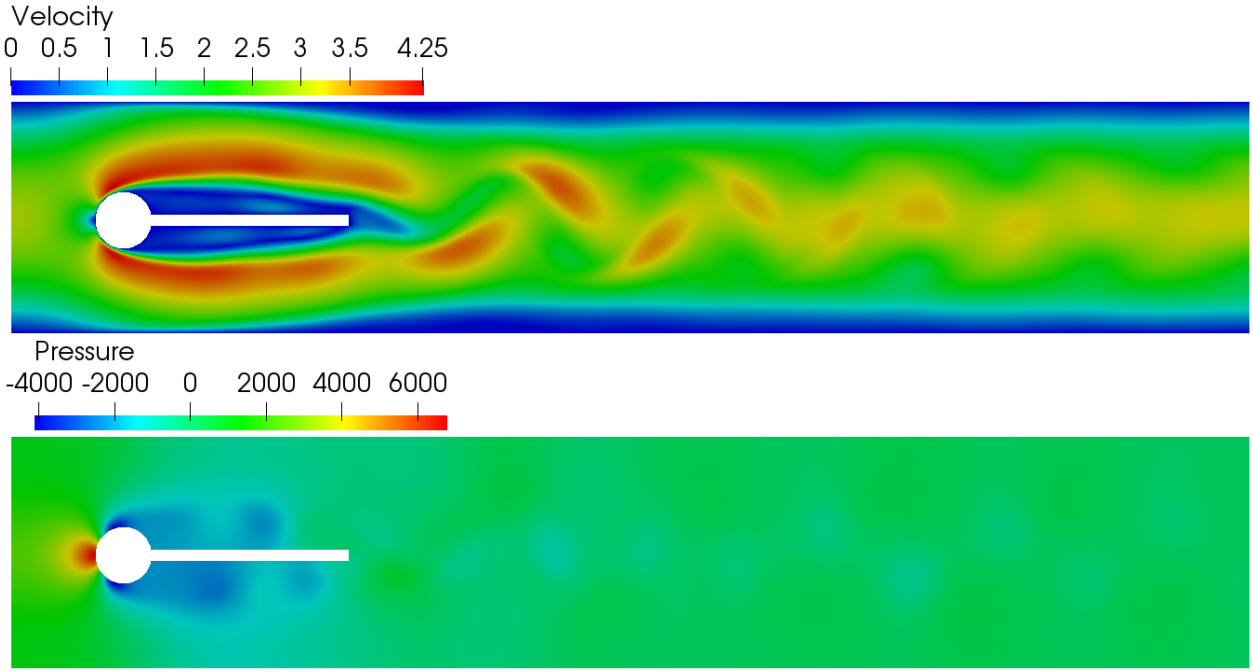

Figure 4.6. Benchmark CFD3: fully developed vortex shedding.

Element	Δt [s]	#DoF	T_{comp}	F_D [N]	F_L [N]	Freq [$\mathrm{s^{-1}}$]
SG $\mathcal{N}_4^2/\mathcal{N}_3^2$	0.01	8050	26m50s	419.24±8.1925	14.459±430.74	4.1666
SG $\mathcal{N}_4^2/\mathcal{N}_3^2$	0.005	8050	38m1s	419.25±8.3475	16.687±435.78	4.1666
SG $\mathcal{N}_5^2/\mathcal{N}_4^2$	0.01	29880	2h10m	440.06±5.7390	-8.415±436.84	4.3478
SG $\mathcal{N}_5^2/\mathcal{N}_4^2$	0.005	29880	3h5m	440.11±5.815	-9.969±439.57	4.3956
SG $\mathcal{N}_6^2/\mathcal{N}_5^2$	0.005	117346	16h58m	439.18±5.685	-12.086±438.06	4.3956
Ref	0.005	707152	N/A	439.45±5.6183	-11.893±437.81	4.3956

Table 4.4. Benchmark CFD3: simulation results for the Newton scheme.

The results of CFD3 are presented in Tables 4.4 and 4.5, including the mixed element type, the time step size, the number of degrees of freedom, the computational time and the values for drag and lift. The results show that both the Newton and IMEX schemes converge to the reference solution. However, the former produces accurate results already with the SG $\mathcal{N}_5^2/\mathcal{N}_4^2$ element and a time step of 0.01s, whereas the latter requires a much finer spatial and temporal resolution to achieve comparable accuracy. On the other hand, the Newton scheme is approximately three times more computationally expensive than the IMEX scheme. In light of that, the IMEX scheme can be more attractive than the Newton scheme for FSI simulations where time steps in the range $0.0005 - 0.0025$s are typically used.

Element	Δt [s]	#DoF	T_{comp}	F_D [N]	F_L [N]	Freq [s^{-1}]
SG $\mathcal{N}_4^2/\mathcal{N}_3^2$	0.01	8050	7m2s	428.27±5.0645	12.168±471.71	6.6666
SG $\mathcal{N}_4^2/\mathcal{N}_3^2$	0.005	8050	10m58s	420.19±9.798	27.0480±451.65	4.1666
SG $\mathcal{N}_5^2/\mathcal{N}_4^2$	0.01	29880	37m50s	448.29±5.15	49.5708±460.93	7.6923
SG $\mathcal{N}_5^2/\mathcal{N}_4^2$	0.005	29880	55m29s	442.68±3.3155	18.7685±351.07	8.6956
SG $\mathcal{N}_5^2/\mathcal{N}_4^2$	0.0025	29880	1h26m	440.29±6.2665	-10.6275±448.60	4.3956
SG $\mathcal{N}_6^2/\mathcal{N}_5^2$	0.005	117346	4h36m	442.57±3.242	25.119±359.45	8.6956
SG $\mathcal{N}_6^2/\mathcal{N}_5^2$	0.0025	117346	6h10m	439.98±6.1255	-11.701±448.09	4.3956
Ref	0.005	707152	N/A	439.45±5.6183	-11.893±437.81	4.3956

Table 4.5. Benchmark CFD3: simulation results for the IMEX scheme.

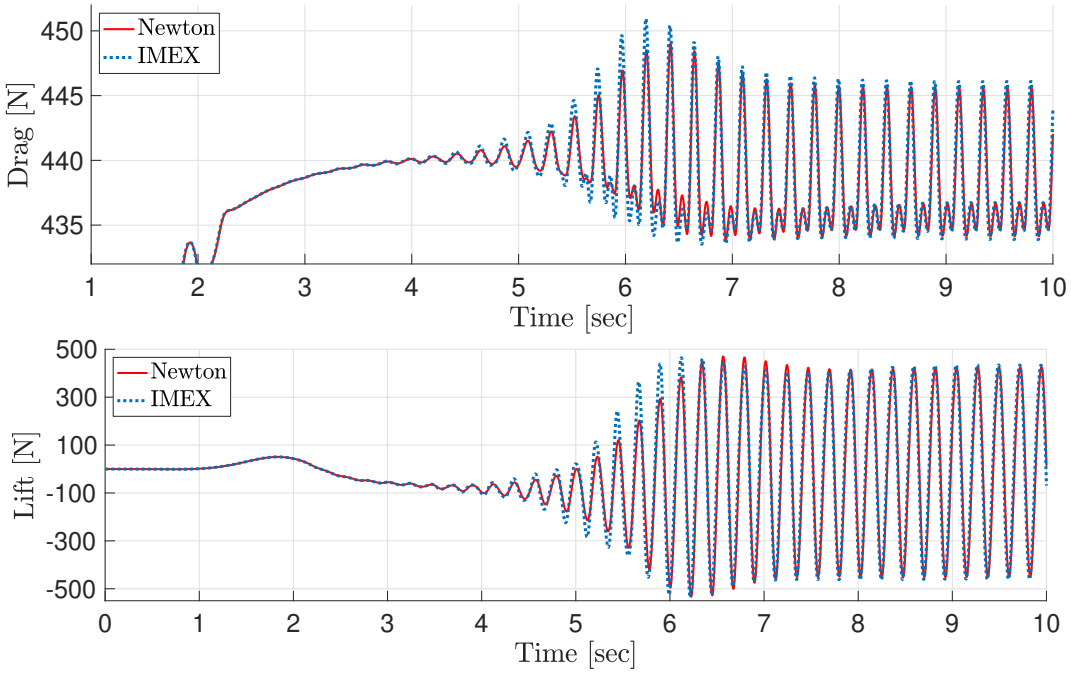

Figure 4.7. Benchmark CFD3: drag and lift over time for the Newton and IMEX schemes with the finest spatial and temporal resolution.

4.4 Fluid-structure interaction

In this section, I apply the fluid and structure models introduced in Sections 4.2 and 4.3 to the fluid-structure interaction problem. The main source of complexity in FSI is that the domain occupied by the fluid changes over time. One way to account for this effect is by defining a deformation mapping for the fluid domain, called an arbitrary Lagrangian-Eulerian (ALE) mapping [11, 32]. Here, I first introduce an ALE formulation of the FSI problem and describe the partitioned approach to its solution. Then, I discuss several state-of-the-art approaches to construct ALE mappings and propose the TINE method as an alternative. Additionally, I introduce the Jacobian-based local stiffening which can augment all of the approaches. After that, I compare different ALE approaches in a simplified FSI-like scenario. Finally, I validate my implementation of the isogeometric FSI solver on the benchmark from Section 4.1 and study the performance of different ALE approaches.

4.4.1 FSI problem in ALE formulation

Assume that the fluid domain[6] Ω^f moves in the sense that its boundary $\partial\Omega^f$ (or just a part of the boundary denoted by Γ) changes sufficiently smoothly over time. Let $\Omega^f(t)$ denote the fluid domain and $\Gamma(t)$ denote a moving part of its boundary at time t. At time $t = 0$, the initial configuration of the domain is denoted by $\Omega_0^f = \Omega^f(0)$, and the initial configuration of the deforming boundary is denoted by $\Gamma_0 = \Gamma(0)$. Assume further, that the motion of the domain's boundary is given in terms of the displacement field $\mathbf{u}^f(t) : \partial\Omega_0^f \to \mathbb{R}^3$ such that

$$\Gamma(t) = \Gamma_0 + \mathbf{u}^f(t) \quad \text{and} \quad \mathbf{u}^f(t) = \mathbf{0} \text{ on } \partial\Omega_0^f \setminus \Gamma_0. \tag{4.88}$$

Displacement field $\mathbf{u}^f$ defines a mapping $\mathbf{D}^f(t) : \partial\Omega_0^f \to \partial\Omega^f(t)$ which can be extended to Ω_0^f to describe the motion of the domain's interior. This extension should be at least invertible; ideally, it should be a diffeomorphism. In the context of fluid dynamics, this extension is called an *arbitrary Lagrangian-Eulerian*[7] (ALE) *mapping*. Since an extension of a particular boundary displacement is not unique, an ALE mapping can be defined arbitrarily. Several methods exist to construct ALE mappings, which I discuss in Section 4.4.3. Because an ALE mapping can be interpreted as a deformation of the computational mesh[8], some of these methods we have already encountered in Chapter 3, for example, the ILE and TINE methods.

Assume that an ALE mapping $\mathbf{D}^f$ is given or can be constructed. Similarly to the elasticity theory in Section 3.3.3, its gradient $\mathbf{F}^f$ and the Jacobian determinant J^f are defined by[9]

$$\mathbf{F}^f = \nabla_{\widehat{\mathbf{x}}}\mathbf{D}^f = \mathbf{I} + \nabla_{\widehat{\mathbf{x}}}\mathbf{u}^f \quad \text{and} \quad J^f = \det \mathbf{F}^f. \tag{4.89}$$

Then the motion of the fluid in the moving domain $\Omega^f(t)$ over the period of time $[0, T]$ can be described by the Navier-Stokes equations pulled back into the stationary initial configuration Ω_0^f:

$$\rho^f J^f \dot{\mathbf{v}} + \rho^f J^f \nabla_{\widehat{\mathbf{x}}}\mathbf{v} \cdot (\mathbf{F}^f)^{-1}(\mathbf{v} - \dot{\mathbf{u}}^f) = \operatorname{div}_{\widehat{\mathbf{x}}}(J^f \boldsymbol{\sigma}^f_{\widehat{\mathbf{x}}}(\mathbf{v}, p)(\mathbf{F}^f)^{-T}) + \rho^f \mathbf{g} \text{ in } \Omega_0^f \times [0, T], \tag{4.90}$$

$$\operatorname{div}_{\widehat{\mathbf{x}}}(J^f(\mathbf{F}^f)^{-1}\mathbf{v}) = 0 \text{ in } \Omega_0^f \times [0, T], \tag{4.91}$$

where the fluid stress tensor is computed in Ω_0^f as

$$\boldsymbol{\sigma}^f_{\widehat{\mathbf{x}}}(\mathbf{v}, p) = -p\mathbf{I} + \rho^f \nu^f \Big(\nabla_{\widehat{\mathbf{x}}}\mathbf{v}(\mathbf{F}^f)^{-1} + (\mathbf{F}^f)^{-T} \nabla_{\widehat{\mathbf{x}}}\mathbf{v}^T \Big). \tag{4.92}$$

Assume that the motion of the fluid domain is caused by interaction with an elastic body. Let the elastic body in its initial configuration occupy the domain Ω_0^s, such that $\partial\Omega_0^s \cap \partial\Omega_0^f = \Gamma_0$. Then Γ_0 is called an *FSI interface*. The motion of the elastic body is described by the time-dependent equations of nonlinear elasticity (4.1):

$$\rho^s \ddot{\mathbf{u}}^s = \operatorname{div}_{\widehat{\mathbf{x}}} \mathbf{P}(\mathbf{u}^s) + \rho^s \mathbf{g} \text{ in } \Omega_0^s \times [0, T]. \tag{4.93}$$

[6] In this section, I use the superscripts f and s to indicate that a certain object is related to the fluid or to the structure.

[7] To understand the origin of the term, observe that an ALE mapping is just a deformation of the initial configuration of the fluid domain Ω_0^f. However, unlike in the elasticity theory, this deformation is not necessarily associated with the motion of fluid material points. In particular, $\dot{\mathbf{u}}^f \neq \mathbf{v}$. Still, it is a mapping between the initial and deformed configurations of the fluid domain or, in other words, a transformation between the Lagrangian and Eulerian frameworks.

[8] In what follows, I even use the terms *ALE mapping* and *mesh deformation* interchangeably.

[9] Just as in the elasticity theory, the subscript $\widehat{\mathbf{x}}$ indicates that the differentiation is performed with respect to coordinates $\widehat{\mathbf{x}}$ in Ω_0^f.

Since the motion of the elastic body drives the motion of the fluid domain, the following condition holds:

$$\mathbf{u}^f = \mathbf{u}^s \quad \text{on } \Gamma_0 \times [0,T]. \tag{4.94}$$

In this sense, the ALE mapping $\mathbf{D}^f$ is just a continuous extension of the structural deformation $\mathbf{D}^s$ into the fluid domain.

The physical interaction between the fluid and the structure is described by the coupling conditions

$$\mathbf{v} = \dot{\mathbf{u}}^s \quad \text{on } \Gamma_0 \times [0,T], \tag{4.95}$$

$$\mathbf{P}(\mathbf{u}^s) \cdot \mathbf{n} = J^f \boldsymbol{\sigma}^f_{\hat{\mathbf{x}}}(\mathbf{v}, p)(\mathbf{F}^f)^{-T} \cdot \mathbf{n} \quad \text{on } \Gamma_0 \times [0,T]. \tag{4.96}$$

Here, condition (4.95) is called the *kinematic continuity*; it is a form of the no-slip condition which assures that the fluid stays attached to the structure. Note that from (4.94), it follows that $\mathbf{v} = \dot{\mathbf{u}}^f$ on Γ_0. Condition (4.96) is called the *dynamic continuity*; it maintains the balance of forces on the FSI interface.

Together, equations (4.90–4.96) describe the interaction between the fluid and the structure, and I refer to them as the *ALE-FSI system*. Note that the ALE-FSI system is formulated in the stationary domain $\Omega_0^f \cup \Omega_0^s$. Therefore, standard finite element or isogeometric Galerkin methods for problems in stationary domains can be used to solve it. In that, the ALE-FSI system follows the same idea as the nonlinear elasticity equations (3.49), which are formulated in the reference configuration Ω_0^s instead of the a priori unknown deformed configuration Ω^s. Of course, this implicit treatment of motion comes at a price of increased complexity and highly nonlinear structure of the ALE-FSI system. The system is completed by an appropriate choice of the initial and boundary conditions for its components.

Solution approaches: monolithic vs partitioned

There are two ways to solve the ALE-FSI system (4.90–4.96) which are known as the *monolithic* and *partitioned* approaches. In a nutshell, the monolithic approach treats the system as a whole whereas the partitioned approach deals with each component (the fluid, the structure and the ALE mapping) separately and achieves coupling by exchanging information between the components. Using an abstract notation $\mathbf{f}_i$, $\mathbf{s}_i$ and $\mathbf{a}_i$ to denote the solution of each component at time t_i, the approaches can be schematically illustrated by the diagrams in Figure 4.8.

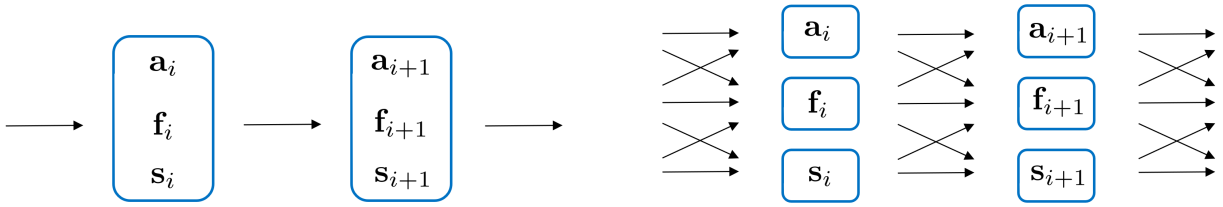

Figure 4.8. Monolithic approach to the FSI problem: the system is discretized and solved as a whole (left). Partitioned approach to the FSI problem: each component is treated separately, and the coupling is achieved by exchanging information between the components.

With the monolithic approach, all equations (4.90–4.96) are discretized together in space and time. The result is a system of nonlinear algebraic equations which has to be solved at each time

step, for example, using Newton's method. Because of the coupling and the ALE mapping, the linearization of the ALE-FSI system is more complex than linearizations of the Navier-Stokes and the nonlinear elasticity equations. In particular, the existing individual solvers cannot be reused, which is the main disadvantage of the monolithic approach. Moreover, the system of linear algebraic equations resulting from the linearization includes the degrees of freedom for each unknown, namely for the velocity, pressure, and displacement of the structure and the fluid domain. As a result, this large linear system sometimes requires prohibitively much computer memory to solve. On the other hand, the monolithic approach is known for its stability and accuracy and can be much more efficient than the partitioned approach. Nevertheless, I do not use the monolithic approach in this work.

In contrast to the monolithic approach, the partition approach seeks to reuse the existing solvers for individual components as much as possible. To that end, the ALE-FSI system (4.90–4.96) is split into three subsystems, one for each component. To achieve coupling, the subsystems are iteratively solved at each time step in a certain sequence. After each iteration, the information is exchanged between them via the boundary conditions on the FSI interface. Under certain conditions, this iterative procedure converges to the solution of the original system. Among advantages of the partitioned approach are implementation simplicity, a huge parallelization potential and smaller memory requirements (in comparison to the monolithic approach). The main disadvantage is a nasty nature of the iterative coupling which may have stability issues and a low convergence rate. Even though special techniques for accelerating convergence exists, the partitioned approach can be too slow or even fail. Still, the partitioned approach is a good starting point for FSI neophytes like myself. It allows one to quickly implement a minimal viable version of an FSI solver and leaves a lot of freedom to experiment with the individual components.

4.4.2 Partitioned approach with strong coupling

Let us dive into details. As I mention above, the partitioned approach splits the ALE-FSI system (4.90–4.96) into three *subsystems*. Let us denote these subsystems by $\mathcal{F}$ (for the fluid), $\mathcal{S}$ (for the structure) and $\mathcal{A}$ (for the ALE mapping). Each subsystem is described below.

The fluid subsystem $\mathcal{F}$ consists of the Navier-Stokes equations in the ALE formulation (4.90–4.91) and kinematic continuity condition (4.95). Since the subsystems in the partitioned approach are independent, they no longer have to be formulated in the same stationary domain $\Omega_0^f \cup \Omega_0^s$. In particular, we can push the fluid subsystem forward into the deformed domain $\Omega^f(t)$:

$$\rho^f \dot{\mathbf{v}} + \rho^f \nabla_{\mathbf{x}} \mathbf{v} \cdot (\mathbf{v} - \dot{\mathbf{u}}^f) = \rho^f \nu^f \Delta_{\mathbf{x}} \mathbf{v} - \nabla_{\mathbf{x}} p + \rho^f \mathbf{g} \ \text{ in } \Omega^f(t) \times [0,T], \tag{4.97}$$

$$\operatorname{div}_{\mathbf{x}} \mathbf{v} = 0 \ \text{ in } \Omega^f(t) \times [0,T], \tag{4.98}$$

$$\mathbf{v} = \dot{\mathbf{u}}^s \ \text{ on } \Gamma(t) \times [0,T]. \tag{4.99}$$

This much simpler formulation is equivalent to the original. Moreover, apart from subtraction of the *ALE velocity* $\dot{\mathbf{u}}^f$ from the advective velocity in the advection term, equations (4.97–4.98) are identical to the Navier-Stokes equations (4.26–4.27) in the stationary domain. Therefore, any standard Navier-Stokes solver can be applied with only minor modifications required.

Assume that the fluid solution $\mathbf{f}_i = (\mathbf{v}_i, p_i)$ and the ALE solution $\mathbf{a}_{i+1} = \mathbf{u}_{i+1}^f$ are known. Let us approximate the ALE velocity $\dot{\mathbf{u}}_{i+1}^f$ by

$$\dot{\mathbf{u}}_{i+1}^f = \frac{\mathbf{u}_{i+1}^f - \mathbf{u}_i^f}{\Delta t}. \tag{4.100}$$

Then the following procedure is used to compute the next fluid solution $\mathbf{f}_{i+1} = (\mathbf{v}_{i+1}, p_{i+1})$. First, the initial configuration of the fluid domain Ω_0^f is deformed using the ALE displacement $\mathbf{u}_{i+1}^f$ to

obtain the deformed configuration Ω^f_{i+1}. Since the ALE displacement is constructed using the NURBS basis associated with the parametrization $\mathbf{G}^f_0$ of Ω^f_0, this step amounts to moving the control points of $\mathbf{G}^f_0$ in the direction specified by the control points of $\mathbf{u}^f_{i+1}$. The result is the parametrization $\mathbf{G}^f_{i+1}$ of Ω^f_{i+1}. After that, equations (4.97–4.99) are solved in Ω^f_{i+1} for one time step using $\mathbf{v}_i$ as an initial condition and treating $\dot{\mathbf{u}}^f_{i+1}$ as fixed. As a result, one obtains the fluid solution $\mathbf{f}_{i+1} = (\mathbf{v}_{i+1}, p_{i+1})$ that takes the ALE solution $\mathbf{a}_{i+1}$ into account. In other words, this is a one-way coupling between the ALE and fluid subsystems, which I denote by

$$\mathbf{f}_{i+1} = \mathcal{F}(\mathbf{f}_i, \mathbf{a}_{i+1}). \tag{4.101}$$

Note that all isogeometric objects like $\mathbf{v}_*$, p_* and $\mathbf{u}^f_*$ can be easily mapped from Ω^f_0 to Ω^f_{i+1} because they share the same NURBS basis. When assembling the linear system for equations (4.97–4.98), the parametrization $\mathbf{G}^f_{i+1}$ is used to push forward the NURBS basis functions.

The structure subsystem $\mathcal{S}$ consists of equations of nonlinear elasticity (4.93) and the dynamic continuity condition (4.96). Provided the structure solution $\mathbf{s}_i = (\mathbf{u}^s_i, \dot{\mathbf{u}}^s_i)$, fluid solution $\mathbf{f}_{i+1} = (\mathbf{v}_{i+1}, p_{i+1})$ and ALE solution $\mathbf{a}_{i+1} = \mathbf{u}^f_{i+1}$, the subsystem computes the next solution $\mathbf{s}_{i+1} = (\mathbf{u}^s_{i+1}, \dot{\mathbf{u}}^s_{i+1})$. To that end, equations (4.93) are solved for one time step with the initial conditions $\mathbf{s}_i$. The force $J^f \boldsymbol{\sigma}^f_{\hat{\mathbf{x}}}(\mathbf{v}, p)(\mathbf{F}^f)^{-T}$ exerted by the fluid on the structure along the FSI interface is computed using $\mathbf{f}_{i+1}$, p_{i+1} and $\mathbf{u}^f_{i+1}$ and is treated as fixed. In this way, a one-way coupling between the fluid and the structure is achieved:

$$\mathbf{s}_{i+1} = \mathcal{S}(\mathbf{s}_i, \mathbf{f}_{i+1}). \tag{4.102}$$

Finally, the ALE subsystem $\mathcal{A}$ computes the ALE solution $\mathbf{a}_{i+1} = \mathbf{u}^f_{i+1}$ provided $\mathbf{a}_i$ and $\mathbf{s}_{i+1}$. Assuming that $\mathbf{u}^f_i = \mathbf{u}^s_i$ on Γ_0 holds already, the task of the ALE subsystem is to update the ALE displacement field $\mathbf{u}^f_i \to \mathbf{u}^f_{i+1}$ such that $\mathbf{u}^f_{i+1} = \mathbf{u}^s_{i+1}$ on Γ_0. In other words, this is a one-way coupling between the structure and the ALE mapping:

$$\mathbf{a}_{i+1} = \mathcal{A}(\mathbf{a}_i, \mathbf{s}_{i+1}). \tag{4.103}$$

I described concrete ways to implement an ALE update in Section 4.4.3.

Iterative coupling

After explaining each subsystem, we can formulate the iterative coupling algorithm for the partitioned approach. Assume that at time t_{i-1}, the solutions $\mathbf{a}_{i-1}$, $\mathbf{f}_{i-1}$ and $\mathbf{s}_{i-1}$ are known. The goal of the algorithm is to iteratively compute increasingly better predictions $\mathbf{a}^j_i$, $\mathbf{f}^j_i$ and $\mathbf{s}^j_i$ for the solution at time t_i. The algorithm is initialized with

$$\mathbf{a}^0_i = \mathbf{a}_{i-1}, \quad \mathbf{f}^0_i = \mathbf{f}_{i-1}, \quad \mathbf{s}^0_i = \mathbf{s}_{i-1}. \tag{4.104}$$

Then for $j \geqslant 0$, the subsystems are solved in the following sequence:

$$\mathbf{a}^{j+1}_i = \mathcal{A}(\mathbf{a}^0_i, \mathbf{s}^j_i) \quad \longrightarrow \quad \mathbf{f}^{j+1}_i = \mathcal{F}(\mathbf{f}^0_i, \mathbf{a}^{j+1}_i) \quad \longrightarrow \quad \mathbf{s}^{j+1}_i = \mathcal{S}(\mathbf{s}^0_i, \mathbf{f}^{j+1}_i). \tag{4.105}$$

The described algorithm is called the *Fluid-Dirichlet-Structure-Neumann* (FDSN) *algorithm* [78]. If the coupling loop (4.105) is executed once per time step, the coupling is called *weak* since the coupling conditions (4.95–4.96) are satisfied only approximately. The partitioned approach with weak coupling is known to be unconditionally unstable. On the other hand, if the coupling loop

(4.105) is executed until convergence, the coupling is called *strong*. Note that with strong coupling, the individual solvers for each subsystem have to be restarted from the initial conditions $\mathbf{a}_{i-1}$, $\mathbf{f}_{i-1}$ and $\mathbf{s}_{i-1}$ at each iteration. The FDSN algorithm with strong coupling is illustrated schematically in Figure 4.9.

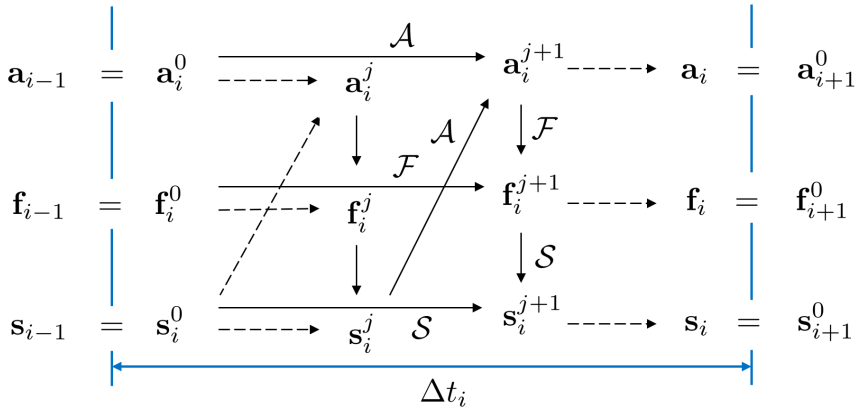

Figure 4.9. Fluid-Dirichlet-Structure-Neumann iterative coupling algorithm for the partitioned approach.

The convergence of the FDSN algorithm is defined in terms of the displacement of the FSI interface Γ. Concretely, let the *interface displacement* and the *interface residual* be defined as

$$\mathbf{u}_i^{\Gamma,j+1} = (\mathbf{u}_i^{s,j+1} - \mathbf{u}_i^{s,j})|_{\Gamma_0} \quad \text{and} \quad \mathbf{r}_i^{j+1} = \mathbf{u}_i^{\Gamma,j+1} - \mathbf{u}_i^{\Gamma,j}. \tag{4.106}$$

Then one says that the FDSN algorithm has converged after $j+1$ iterations if

$$||\mathbf{r}_i^{j+1}||_{L^2(\Gamma_0)} < \varepsilon. \tag{4.107}$$

The convergence rate of the FDSN algorithm depends on the ratio between the fluid and structure densities [78]. If the ratio is close to one, that is, $\rho^s \sim \rho^f$, then the convergence rate is lowest. In this case, the fluid domain and the structure together behave similarly to an elastic body that oscillates around the equilibrium state. In order to increase the convergence rate, several acceleration techniques can be applied, for example, the Aitken relaxation method described in the next section.

Aitken relaxation

The main idea of the *Aitken relaxation* method is to increase the convergence rate of the iterative coupling algorithm by improving the interface displacement $\mathbf{u}_i^{\Gamma,j+1}$ using data from the two previous iterations [73]. For $j \geqslant 2$, the *corrected interface displacement* $\widetilde{\mathbf{u}}_i^{\Gamma,j+1}$ is computed as

$$\widetilde{\mathbf{u}}_i^{\Gamma,j+1} = \mathbf{u}_i^{\Gamma,j} + \omega_{j+1}\mathbf{r}_i^{j+1} = (1-\omega_{j+1})\mathbf{u}_i^{\Gamma,j} + \omega_{j+1}\mathbf{u}_i^{\Gamma,j+1}, \tag{4.108}$$

where ω_{j+1} is a *relaxation parameter*. For $j \geqslant 2$, ω_{j+1} is computed from the recursive formula

$$\omega_{j+1} = -\omega_j \frac{(\mathbf{r}_i^j, \mathbf{r}_i^{j+1} - \mathbf{r}_i^j)_{L^2(\Gamma_0)}}{||\mathbf{r}_i^{j+1} - \mathbf{r}_i^j||^2_{L^2(\Gamma_0)}}, \tag{4.109}$$

with $\omega_1 = \omega_2 = 1$. Thus, the first two iterations of the iterative coupling algorithm are not corrected. Note that the interface residual $\mathbf{r}_i^{j+1}$ is defined for the not yet corrected interface displacement $\mathbf{u}_i^{\Gamma,j+1}$. After the relaxation step, the corrected interface displacement $\widetilde{\mathbf{u}}_i^{\Gamma,j+1}$ is used as an input for the ALE subsystem at the next coupling iteration.

4.4.3 Construction of ALE mappings

Let us now turn our attention to construction of ALE methods, or in other words, to fluid mesh deformation. Due to arbitrariness of ALE mappings, one has freedom to choose any method that is sufficiently fast or robust. One crucial requirement is that the chosen method produces bijective ALE mappings, that is, the condition

$$J^f(\mathbf{D}^f) = \det(\mathbf{I} + \nabla_{\widehat{\mathbf{x}}}\mathbf{u}^f) > 0 \tag{4.110}$$

must hold. Here, I describe two most popular options, namely the harmonic extension [32, 78, 79, 80] and the incremental linear elasticity (ILE) [11, 33, 34, 35, 36, 37]. Additionally, I propose using the TINE method introduced in Chapter 3 as a mesh update method. After that, I describe the Jacobian-based local stiffening [36] which can be used to improve all of the discussed methods.

Harmonic extension

As I explain above, the ALE displacement field $\mathbf{u}^f$ is a continuous extension of the structure displacement field $\mathbf{u}^s$ into the fluid domain. The simplest way to construct $\mathbf{u}^f$ is by means of the *harmonic extension* (HE). Given the interface displacement $\mathbf{u}_i^\Gamma$ (or corrected interface displacement $\widetilde{\mathbf{u}}_i^\Gamma$) at time t_i, the ALE displacement field $\mathbf{u}_i^f$ is computed by solving Laplace's equation in the initial configuration of the fluid domain Ω_0^f for every displacement component:

$$\Delta_{\widehat{\mathbf{x}}}\mathbf{u}_i^f = \mathbf{0} \text{ in } \Omega_0^f, \tag{4.111}$$

$$\mathbf{u}_i^f = \mathbf{u}_i^\Gamma \text{ on } \Gamma_0, \tag{4.112}$$

$$\mathbf{u}_i^f = \mathbf{0} \text{ on } \partial\Omega_0^f \setminus \Gamma_0. \tag{4.113}$$

The interface displacement serves as a Dirichlet boundary condition on the FSI interface Γ_0, while the prescribed displacement is zero at the rest of the boundary. Note that the method does not require information about $\mathbf{u}^f$ at the previous time step.

Harmonic extension is the cheapest method to compute the ALE displacement. However, it is not designed to preserve bijectivity of the ALE mapping and, therefore, can only handle rather small deformations. Moreover, in the vicinity of the domain's corners, the solution of Laplace's equation locally behaves like $r^{\pi/\omega}$, where r is the distance to the corner and ω is the angle. For reentrant corners ($\omega > \pi$), the solution does not belong to $H^1(\Omega_0^f)$ since the derivatives tend to infinity. As a consequence, the corresponding ALE mapping loses its bijectivity.

One possible improvement of the HE method is to use the bi-harmonic equation

$$\Delta_{\widehat{\mathbf{x}}}^2\mathbf{u}_i^f = \mathbf{0} \text{ in } \Omega_0^f. \tag{4.114}$$

The resulting *bi-harmonic extension* produces smoother displacement fields and does not have problems at the reentrant corners [32]. However, equation (4.114) is rather numerically costly to solve. One either has to use C^1-conforming elements, or mixed elements with an auxiliary variable $\mathbf{q}$ to replace the bi-harmonic equation with two Laplace's equations:

$$\Delta_{\widehat{\mathbf{x}}}\mathbf{u}_i^f = \Delta_{\widehat{\mathbf{x}}}\mathbf{q}_i, \quad \Delta_{\widehat{\mathbf{x}}}\mathbf{q}_i = \mathbf{0} \text{ in } \Omega_0^f. \tag{4.115}$$

Due to its complexity, I do not consider the bi-harmonic mesh deformation technique in this work. In the future, however, I would like to implement it and compare against other mesh deformation techniques.

Rather surprisingly, I have not encountered an incremental variation of the HE method. By this I understand the following algorithm. Assume that the ALE displacement $\mathbf{u}_i^f$ at time t_i is known. It is used to construct a deformed configuration of the fluid domain $\Omega_i^f = \Omega_0^f + \mathbf{u}_i^f$. In the deformed domain, Laplace's equation

$$\Delta_{\mathbf{x}}(\Delta\mathbf{u}_{i+1}^f) = \mathbf{0} \text{ in } \Omega_i^f, \tag{4.116}$$

$$\Delta\mathbf{u}_{i+1}^f = \mathbf{u}_{i+1}^\Gamma - \mathbf{u}_i^\Gamma \text{ on } \Gamma_i, \tag{4.117}$$

$$\Delta\mathbf{u}_{i+1}^f = \mathbf{0} \text{ on } \partial\Omega_i^f \setminus \Gamma_i \tag{4.118}$$

is solved, which yields the displacement increment $\Delta\mathbf{u}_{i+1}^f$. Then at time t_{i+1}, the ALE displacement $\mathbf{u}_{i+1}^f$ is constructed as $\mathbf{u}_i^f + \Delta\mathbf{u}_{i+1}^f$. Note that the incremental harmonic extension method (IHE) is not equivalent to the HE method because the increments are computed in the deformed domain. In this, the IHE method is similar to the incremental linear elasticity method.

Linear elasticity

Another commonly used mesh deformation technique is the already familiar to us incremental linear elasticity method (ILE). In the context of FSI, it has the structure similar to the IHE method. Concretely, given the ALE displacement $\mathbf{u}_i^f$ at time t_i, the following system is solved in the deformed configuration of the fluid domain Ω_i^f to compute the displacement increment $\Delta\mathbf{u}_{i+1}^f$:

$$\operatorname{div}_{\mathbf{x}} \boldsymbol{\sigma}^s(\Delta\mathbf{u}_{i+1}^f) = \mathbf{0} \text{ in } \Omega_i^f, \tag{4.119}$$

$$\Delta\mathbf{u}_{i+1}^f = \mathbf{u}_{i+1}^\Gamma - \mathbf{u}_i^\Gamma \text{ on } \Gamma_i, \tag{4.120}$$

$$\Delta\mathbf{u}_{i+1}^f = \mathbf{0} \text{ on } \partial\Omega_i^f \setminus \Gamma_i. \tag{4.121}$$

Here, $\boldsymbol{\sigma}$ is the Cauchy stress tensor computed using Hooke's law (3.52). Then the next ALE displacement $\mathbf{u}_{i+1}^f$ is defined as $\mathbf{u}_i^f + \Delta\mathbf{u}_{i+1}^f$. The ILE method is essentially the state-of-the-art method in the FSI applications. It is known for being able to handle large deformations while still remaining relatively computationally inexpensive.

For the sake of completeness, let us also study a non-incremental version of the linear elasticity mesh deformation technique. The displacement field $\mathbf{u}_i^f$ at time t_i is computed by solving the equations of linear elasticity in the initial configuration of the fluid domain Ω_0^f:

$$\operatorname{div}_{\widehat{\mathbf{x}}} \boldsymbol{\sigma}^s(\mathbf{u}_i^f) = \mathbf{0} \text{ in } \Omega_0^f, \tag{4.122}$$

$$\mathbf{u}_i^f = \mathbf{u}_i^\Gamma \text{ on } \Gamma_0, \tag{4.123}$$

$$\mathbf{u}_i^f = \mathbf{0} \text{ on } \partial\Omega_0^f \setminus \Gamma_0. \tag{4.124}$$

I refer to this as the linear elasticity (LE) method. Although one can only expect this method to perform well for small deformations, it is still interesting to compare it with other mesh deformation techniques.

Nonlinear elasticity

In addition to the described above mesh deformation techniques, I propose using the tangential incremental nonlinear elasticity method (TINE) with the neo-Hookean material law (3.46) to construct ALE mappings. When applied within the mesh deformation approach to the single-patch

domain parametrization problem, the TINE method is able to handle large prescribed boundary deformations. Therefore, it can be expected to perform well in the FSI context as well. Concretely, it assumes the following form when used as a mesh update method: given the displacement field $\mathbf{u}_i^f$ at time t_i, compute a tangent displacement increment $\Delta\mathbf{u}_{i+1}^f$ by solving the following variational problem in Ω_0^f (originally defined in (3.62)):

$$DP(\mathbf{u}_i^f, \mathbf{w}) \cdot \Delta\mathbf{u}_{i+1}^f = -P(\mathbf{u}_i^f, \mathbf{w}). \tag{4.125}$$

Here, test functions $\mathbf{w}$ belong to the space $(H_0^1(\Omega_0^f))^d$, and $\Delta\mathbf{u}_{i+1}^f$ should satisfy the following boundary conditions:

$$\Delta\mathbf{u}_{i+1}^f = \mathbf{u}_{i+1}^\Gamma - \mathbf{u}_i^\Gamma \text{ on } \Gamma_0 \quad \text{and} \quad \Delta\mathbf{u}_{i+1}^f = \mathbf{0} \text{ on } \partial\Omega_0^f \setminus \Gamma_0. \tag{4.126}$$

After that, define $\mathbf{u}_{i+1}^f$ as $\mathbf{u}_i^f + \Delta\mathbf{u}_{i+1}^f$.

Local stiffening

Most of the fluid mesh deformation happens along the FSI interface Γ_0, where the non-zero Dirichlet boundary condition on $\mathbf{u}^f$ is set. On the other hand, elements in the vicinity of the stationary boundary $\partial\Omega_0^f \setminus \Gamma_0$ undergo almost no deformation. Therefore, their contribution into processing of the applied interface displacement $\mathbf{u}_i^\Gamma$ is negligible. If the deformation could be redistributed away from the FSI interface towards the stationary boundary, the mesh could undergo larger deformations without becoming invalid. This is the idea behind *local stiffening*, which locally changes the way different elements react to the deformation.

One of the simplest ways to implement local stiffening is to drop the Jacobian determinant $\det \nabla_{\boldsymbol{\xi}} \mathbf{G}$ of the domain parametrization $\mathbf{G}$ when transforming the integrals from the physical to the parametric domain during matrix assembly:

$$\int_\Omega (\cdots) d\mathbf{x} = \int_{[0,1]^d} (\cdots) \det \nabla_{\boldsymbol{\xi}} \mathbf{G} d\boldsymbol{\xi} \to \int_{[0,1]^d} (\cdots) d\boldsymbol{\xi}. \tag{4.127}$$

This *Jacobian-based local stiffening* was first proposed in [34]. For elasticity problems, (4.127) can be interpreted as a local change of Young's modulus

$$E \to \frac{E}{\det \nabla_{\boldsymbol{\xi}} \mathbf{G}}, \tag{4.128}$$

which makes elements with low values $\det \nabla_{\boldsymbol{\xi}} \mathbf{G}$ stiffer and elements with high values softer. Therefore, the former elements undergo smaller deformation and are less likely to become invalid.

The Jacobian-based local stiffening acts differently depending on whether the chosen mesh deformation method is formulated in the initial configuration Ω_0^f or in the deformed configuration Ω_i^f of the fluid domain. Namely, if the integrals for matrix assembly are computed in Ω_0^f, the local stiffening is based only on the initial parametrization $\mathbf{G}_0$. However, in the case of Ω_i^f, the local stiffening takes into account also the already applied deformation $\mathbf{D}^f$ since it is based on the Jacobian determinant of $\mathbf{G} = \mathbf{D}^f \circ \mathbf{G}_0$. This effect has both advantages and disadvantages. From one point of view, if a particular element becomes ill-shaped after the deformation, its value of $\det \nabla_{\boldsymbol{\xi}} \mathbf{G}$ decreases. As a result, this element receives more stiffening, which prevents it from becoming even more ill-shaped or invalid. On the other hand, this *deformation-aware* local stiffening essentially makes the material properties deformation-dependent and can cause irreversible *plastic*

deformation. As I show later, this irreversible deformation accumulates over time and can affect the simulation results.

Regardless if the local stiffening is deformation-aware or not, the initial parametrization $\mathbf{G}_0$ has a major contribution in how much stiffening each element receives. Let us consider the fluid domain parametrization from Figure 4.4. In order to reduce the numerical cost of computing the ALE mapping, I choose to deform only three patches adjacent to the FSI. Figure 4.10 shows the distribution of $\det \nabla_{\boldsymbol{\xi}} \mathbf{G}_0$. Although only elements on the left side of the right trapezoidal patch receive significant local stiffening, it is enough to stiffen the fluid mesh around the beam end, which is the most vulnerable region of the mesh.

A more advanced local stiffening technique [37] does not drop the Jacobian determinant but changes the degree in which it enters the integrals:

$$\int_{\Omega} (\cdots) d\mathbf{x} = \int_{[0,1]^d} (\cdots) \det \nabla_{\boldsymbol{\xi}} \mathbf{G} d\boldsymbol{\xi} \to \int_{[0,1]^d} (\cdots) (\det \nabla_{\boldsymbol{\xi}} \mathbf{G})^{1-\chi} d\boldsymbol{\xi}. \tag{4.129}$$

Here, $\chi \geqslant 0$ is called the *stiffening degree*. The higher the stiffening degree is, the more local stiffening is achieved. Too high stiffening degrees, however, may result in a lot of mesh distortion. Since this technique is a generalization of the simple Jacobian dropping (4.127), I also refer to it as the Jacobian-based local stiffening.

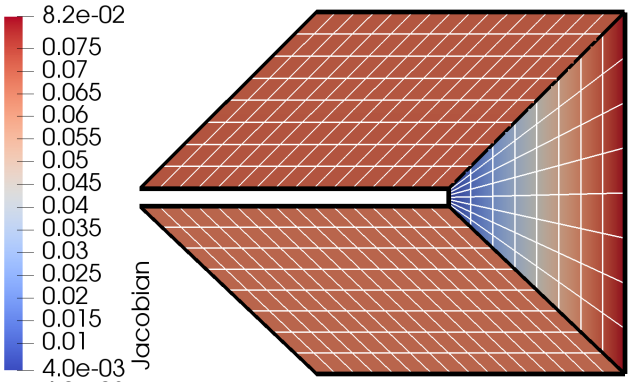

Figure 4.10. Patches of the initial parametrization $\mathbf{G}_0$ of the fluid domain which are used to compute the ALE mapping.

4.4.4 Partial benchmark ALE: mesh deformation

In order to test and compare the HE, IHE, LE, ILE and TINE mesh deformation methods, I use a simplified FSI-like scenario inspired by the original FSI benchmark from Section 4.1. It involves deforming the fluid domain in response to free oscillations of the elastic beam; no fluid flow is considered. Although this mesh deformation scenario is artificial, it mimics real deformations occurring in FSI simulations well enough.

In the fluid domain, only three patches adjacent to the FSI interface are deformed. In order to drive the deformation, I use the periodic motion of the elastic beam from benchmark CSM3 in Section 4.2.4. In order to regulate the magnitude of the fluid mesh deformation, I introduce a *loading level* l in the gravitational acceleration $\mathbf{g} = (0, l)^T\ m/s^2$. With values $l \in [1.5, 2]$, the deformation magnitude is comparable to deformation magnitudes in the ensuing FSI simulation. For the mesh deformation methods based on elasticity theory, I use Poisson's ratio of 0.4.

In order to simplify the data exchange between the solvers, I use matching parametrizations for the beam and the fluid domain. As a result, it is enough to transfer the displacement control points

from the beam solver to the ALE solver in order to set the boundary condition $\mathbf{u}^f = \mathbf{u}^s$ on the FSI interface. For the simulation, I refine both parametrizations three times using uniform h-refinement. Figure 4.11 shows the fluid computational mesh in its initial and in a typical deformed states.

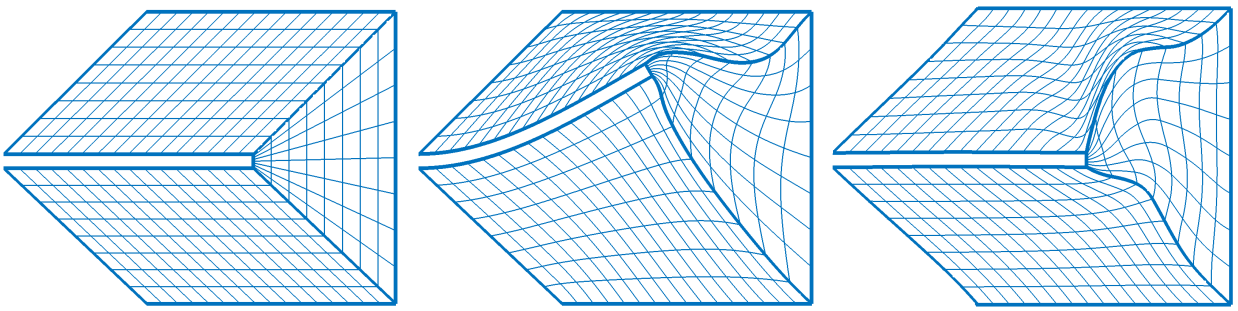

Figure 4.11. Benchmark ALE: fluid mesh in its initial state (left). Fluid mesh deformed with the TINE method ($l = 2$, $\chi = 2.3$, middle). Distortion of the fluid mesh accumulated after 18 oscillation periods with the ILE method ($l = 2$, $\chi = 3$, right).

The first test considers mesh deformation over one period of oscillations. The goal is to study how much mesh deformation each method can handle, first on its own and then with the Jacobian-based local stiffening. The results of the test are presented in Figure 4.12. According to the results, all methods perform poorly without local stiffening and can only handle loading levels of around 0.4-0.5. As a rule, the fluid mesh becomes invalid at the reentrant corners at the beam's end, where mesh elements tend to deform too much and develop angles larger than π.

With local stiffening, all methods can handle increasingly higher loading levels as the stiffening degree χ grows. The growth is similar for all methods until $\chi \approx 2$, when the HE, IHE and LE methods reach a plateau at $l = 1.8$, 1.9 and 1.5, respectively. In contrast to that, the ILE and TINE methods continue to grow until $\chi \approx 3$ and can reach maximal loading level of $l = 2.8$ and 3, respectively. For $\chi > 3.5$, all method show reduction in maximal achievable loading levels. This effect can be explained by an excessive mesh distortion introduced by the local stiffening. Judging by the test results, the ILE and TINE methods show similar behavior when combined with the Jacobian-based local stiffening and can handle much larger deformations than other methods.

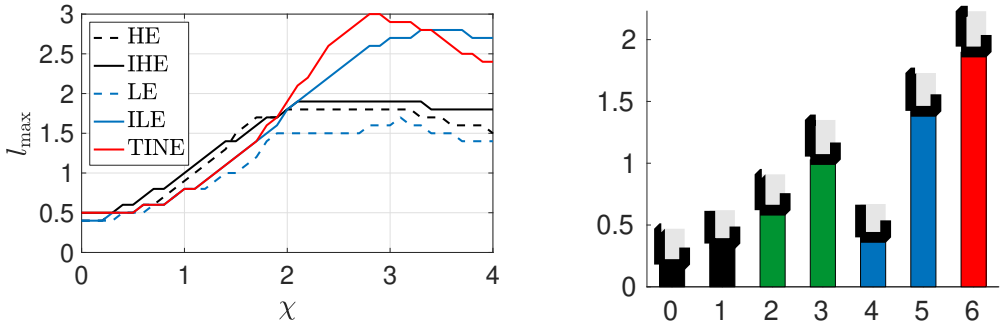

Figure 4.12. Benchmark ALE: maximal achievable loading level $l_{\max}$ vs the stiffening degree χ for different mesh deformation methods (left). Computational time per oscillation period for each method in comparison to the beam solver (right).

It is worth comparing the computational complexity of each method. Figure 4.12 shows computational time consumed by each mesh deformation method during one oscillation period. According

to it, the harmonic extension methods take about $1s - 2s$, the linear elasticity methods take $3s$, and the TINE method requires $5s$, making it the most expensive method. Although all methods require a solution of only one linear system per step, the complexity of nonlinear elasticity equations significantly increases the assembly time for the TINE method.

The second test studies the long-term effects of local stiffening on the fluid mesh. To that end, I perform the simulation over a period of time $[0, 20s]$, which includes roughly 22 oscillation periods. The quantity of interest is the *ALE norm* $||\mathbf{u}^f(\cdot, t)||_{L^2}$. Ideally, a mesh deformation method should return the fluid mesh into its initial state once no interface displacement is applied. That means that the ALE norm should be close to zero at the end of each oscillation period. Figure 4.13 shows the behavior of the ALE norm over time for the ILE and TINE methods with the loading level $l = 2$ and the stiffening degree $\chi = 3$. To put the values into perspective, I plot the *relative deformation* defined as

$$\text{RelDef}(t) = \frac{||\mathbf{u}^f(\cdot, t)||_{L^2}}{\max\limits_{t\in[0,T]} ||\mathbf{u}^f(\cdot, t)||_{L^2}}. \tag{4.130}$$

With the TINE method, the ALE norm behaves periodically and returns to zero at the end of each oscillation period. In contrast to that, with the ILE method the ALE norm at the end of each period grows monotonously. I call this effect *accumulated distortion*. The state of the fluid mesh after 18 oscillation periods is illustrated in Figure 4.11; the simulation can not continue beyond that because the mesh becomes invalid.

In order to study and compare the accumulated distortion effect for all mesh deformation methods, I plot values of relative deformation at the end of each oscillation period in Figure 4.13. From the plot it is clear that only methods based in the deformed configuration of the fluid domain, namely IHE and ILE, suffer from accumulated distortion. Moreover, the magnitude of accumulated distortion seems to grow as the stiffening degree χ increases. As I mention above, accumulated distortion is caused by the nature of the Jacobian-based local stiffening applied in the deformed configuration. It is analogous to making the material properties deformation-dependent which turns the material behavior inelastic.

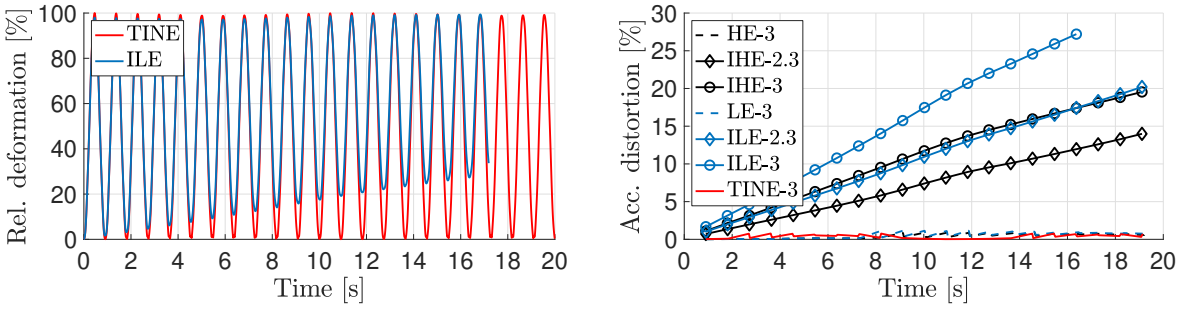

Figure 4.13. Benchmark ALE: relative deformation over time for the ILE and TINE methods ($l = 2$, $\chi = 3$, left). Accumulated distortion over time for different methods with a varying stiffening degree χ ($l = 2$, right).

Based on the two performed tests, the following conclusions can be made:

- with the chosen parametrization of the fluid domain, none of the considered methods can handle mesh deformations occurring in the FSI simulation without local stiffening;
- with the Jacobian-based local stiffening, all methods can handle rather large deformations; however, the ILE and TINE methods are able to handle significantly larger deformations;

- the IHE and ILE methods combined with the Jacobian-based local stiffening result in the accumulated distortion effect; the effect becomes more prominent for high stiffening degrees;
- the TINE method is almost twice as expensive as the ILE method and thrice as expensive as the HE method.

4.4.5 Benchmark FSI2: flow-induced structural vibrations

In this section, I validate my implementation of the isogeometric FSI solver and compare performance of various mesh deformation methods on the benchmark FSI2 from [38]. FSI2 is a time-dependent simulation of a viscous fluid flow interacting with an elastic beam. It involves solving the ALE-FSI system (4.90–4.96) over the time interval $[0, 15\text{s}]$ on the geometry specified in Section 4.1. The initial and boundary conditions for the structure and fluid coincide with the conditions used in the benchmarks CSM3 (Section 4.2.4) and CFD3 (Section 4.3.5).

For the fluid domain, I use the same seven-patch geometry parametrization as in CFD1 and CFD3, see Figure 4.4. Similarly to the ALE benchmark in Section 4.4.4, I allow only three patches adjacent to the FSI interface deform and keep the rest fixed. For the beam, a matching parametrization is used. For analysis, I apply uniform h-refinement to both parametrizations equal number of times. In order to construct the subgrid mixed elements, I keep the pressure space one level of h-refinement coarser. I perform time integration using the IMEX scheme with $\theta = 0.5$ for the fluid and the Newmark method with $\beta = 0.5$ and $\gamma = 1$ for the structure. Although parameter values $\beta = 0.25$ and $\gamma = 0.5$ would, in theory, grant better temporal accuracy, in my experience they lead to an unstable behavior of the FDSN iterative coupling algorithm.

For the simulation, the following parameter values are used: fluid density $\rho^f = 10^3$ kg·m^{-3}, fluid kinetic viscosity $\nu^f = 10^{-3}$ m^2·s^{-1}, maximum inflow velocity $v_{\max} = 1.5$ m·s^{-1}, structure density $\rho^s = 10^4$ kg·m^{-3}, structure Young's modulus $E = 1.4 \times 10^6$ kg·m^{-1}·s^{-2}, structure Poisson's ratio $\nu^s = 0.4$ and gravitational acceleration $\mathbf{g} = (0, 0)^T$ m·s^{-2}. The corresponding Reynolds number is $Re = 100$, which leads to an unstable flow and the development of vortex shedding. The resulting periodic drag and lift forces make the elastic beam oscillate, see Figures 4.16 and 4.17. These oscillations grow in amplitude until they reach a fully developed periodic regime. Figures 4.14 and 4.15 show behavior of the drag and lift forces and the displacement of the beam end point A over time.

h-lvl	Δt $[s]$	F_D $[N]$	F_L $[N]$	$u_x^s(A)$ $[10^{-3}m]$	$u_y^s(A)$ $[10^{-3}m]$	Fr $[1/s]$
4	0.01	153.77±31.80	2.44±140.04	-6.91±6.43	0.71±53.84	1.92
4	0.0025	168.85±50.58	1.96±163.94	-10.59±9.56	0.65±67.32	1.90
5	0.01	182.76±43.01	0.04±183.17	-9.30±8.54	1.22±62.26	1.92
5	0.0025	198.29±62.12	0.39±195.79	-12.72±11.39	1.09±74.16	1.92
6	0.0025	197.28±59.57	0.49±205.43	-12.56±11.26	1.13±73.48	1.92
Ref	0.001	208.83±73.75	0.88±234.2	-14.58±12.44	1.23±80.6	2.0

Table 4.6. Benchmark FSI2: simulation results with the TINE method.

The simulation results with the TINE method are presented in Table 4.6, including the h-refinement level, the time step size, the frequency of oscillations as well as the mean and amplitude values for the drag, lift and displacement of beam end point A. Additionally, Table 4.7 contains data regarding the numerical cost of the simulation. Overall, my results seem to undershoot the reference values by about 10%, which is not terrible given that the reference values have been computed using three times as many degrees of freedom and with a much smaller time step size. Nevertheless, my

implementation is more than capable of reproducing the qualitatively correct behavior of the system, and I can use it to study mesh deformation techniques. The simulation accuracy can be improved by increasing the temporal and spatial resolution[10].

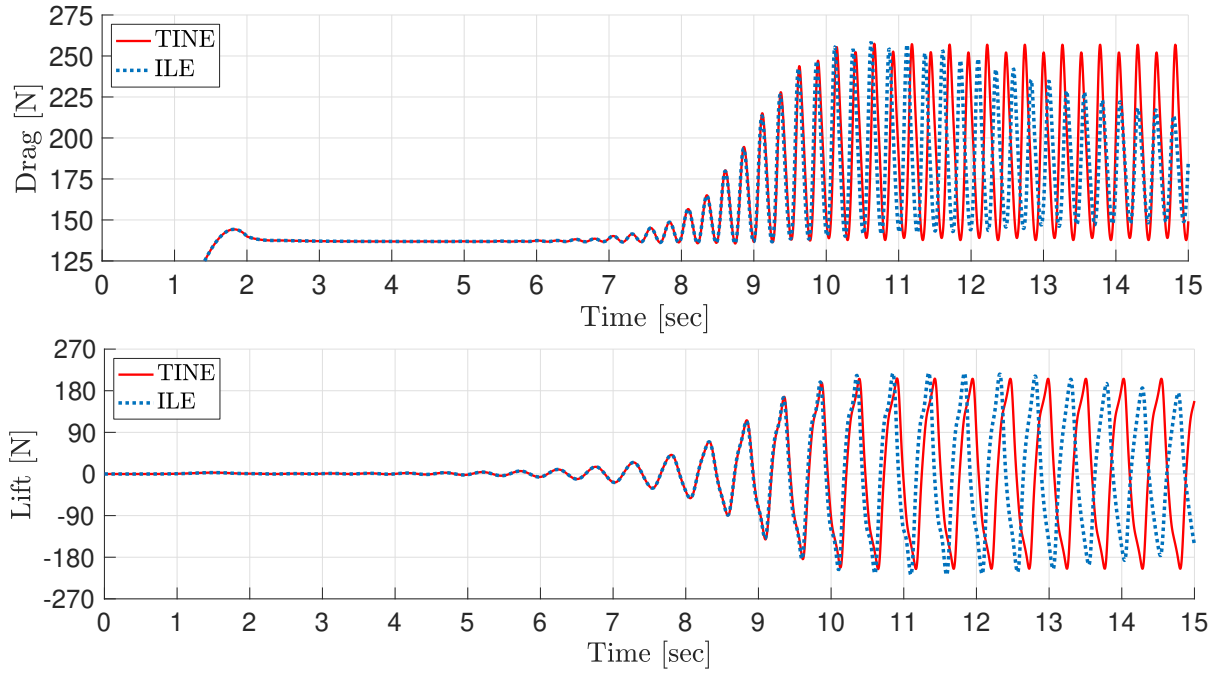

Figure 4.14. Benchmark FSI2: drag and lift.

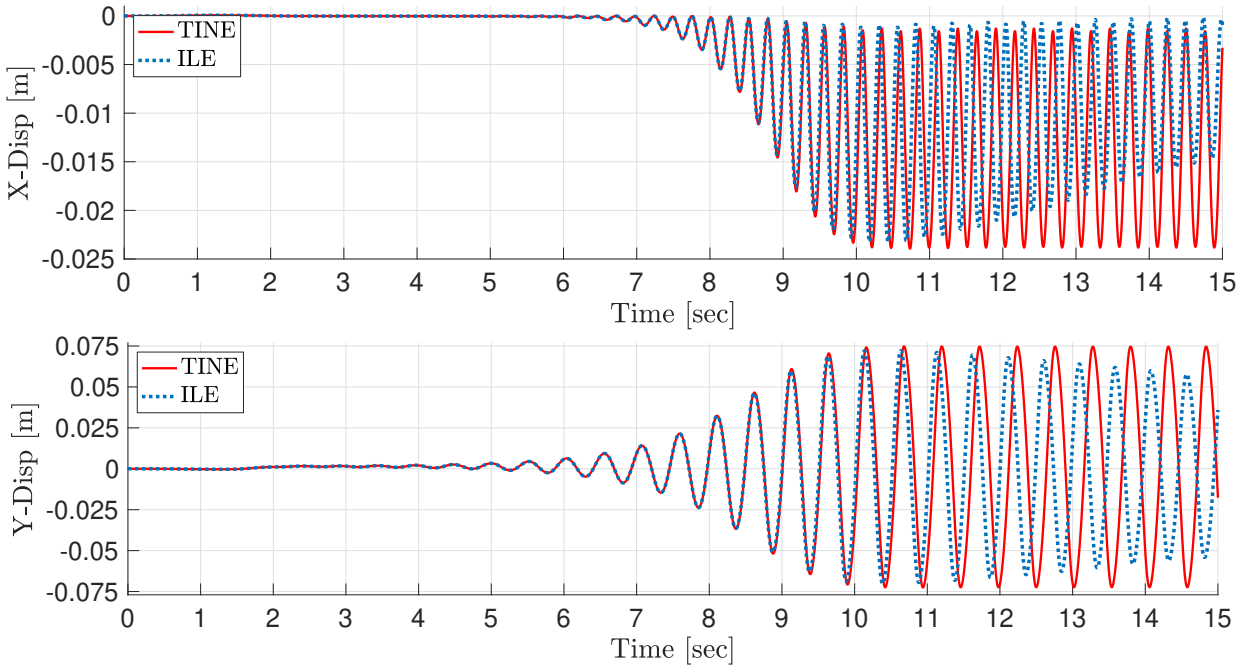

Figure 4.15. Benchmark FSI2: displacement of the beam point A.

[10]To that end, however, a computer cluster would be more appropriate than a laptop.

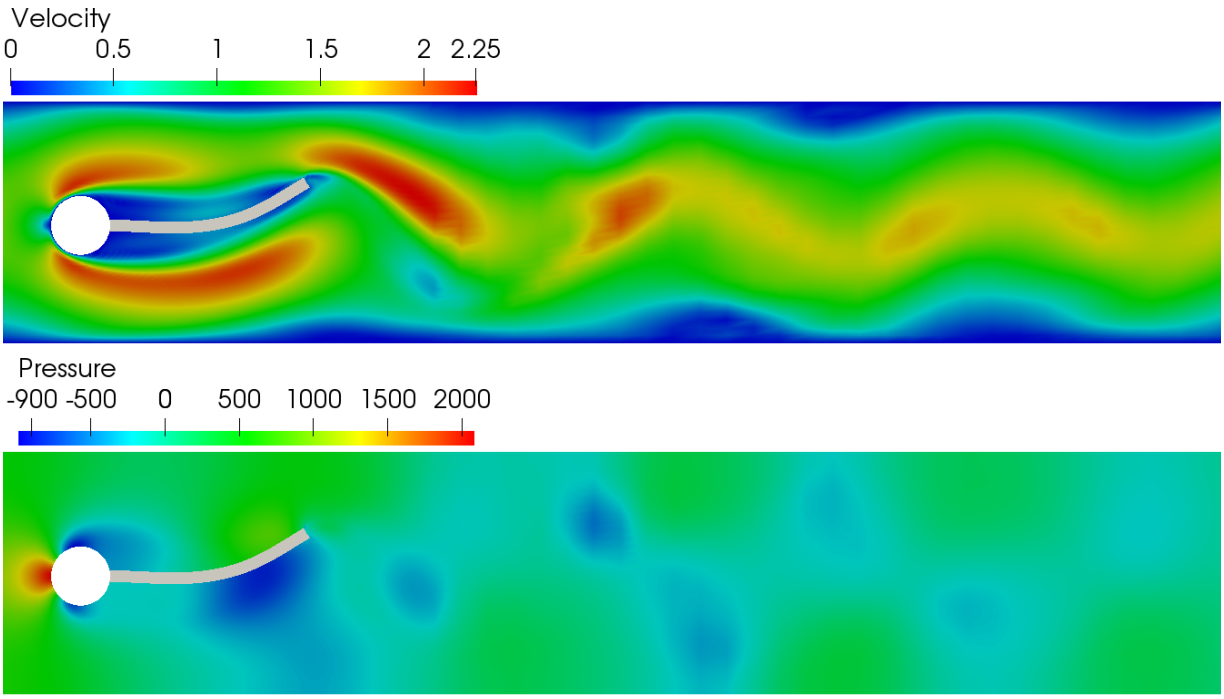

Figure 4.16. Benchmark FSI2: fully developed oscillation regime. Velocity and pressure of the fluid for the maximal upward beam deflection.

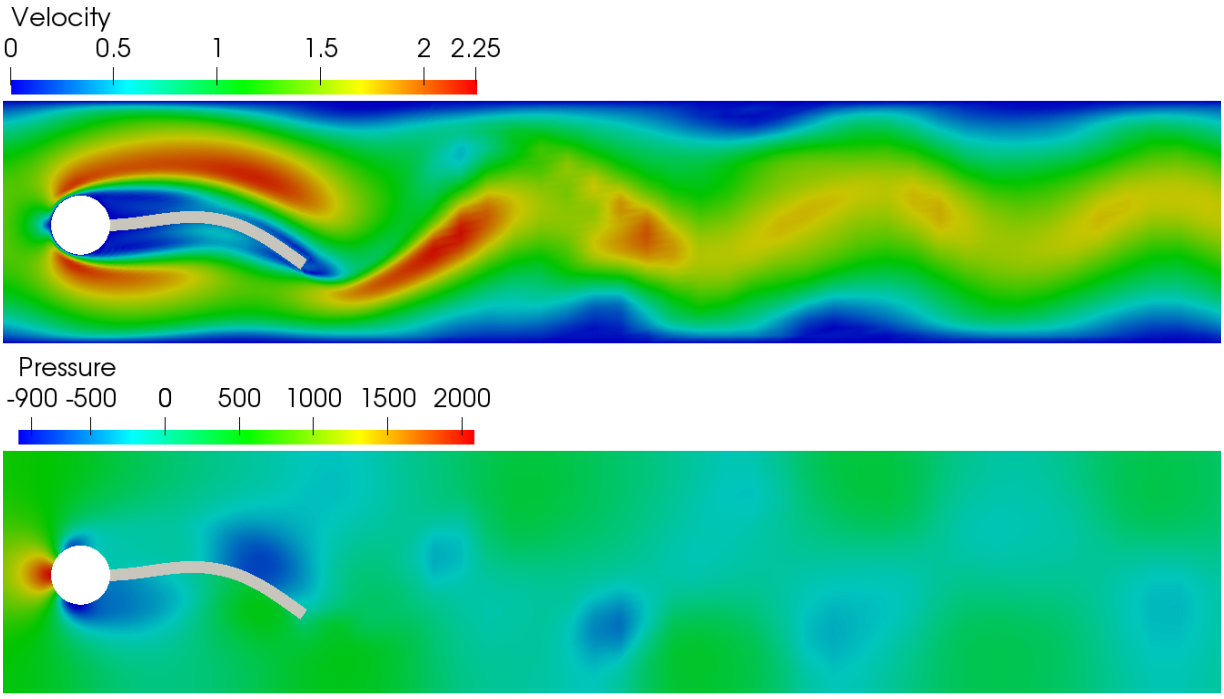

Figure 4.17. Benchmark FSI2: fully developed oscillation regime. Velocity and pressure of the fluid for the maximal downward beam deflection.

Let us now focus on the fluid mesh deformation. Out of five mesh deformation methods considered in Section 4.4.3, only the ILE and TINE methods were able to handle the deformations occurring in the simulation. For successful simulations, I have used the stiffening degree $\chi = 2.5$ and Poisson's ratio $\nu^m = 0.4$. Although a precise lower limit for the necessary stiffening degree is hard to determine due to an extremely high numerical cost of the simulation, I have found that both methods fail with $\chi = 2$. This means that $\chi = 2.5$ is a reasonable, not excessively high value. The HE, IHE and LE methods with $\chi = 2.5$ were only able to handle the fluid mesh deformation until $t = 8$s.

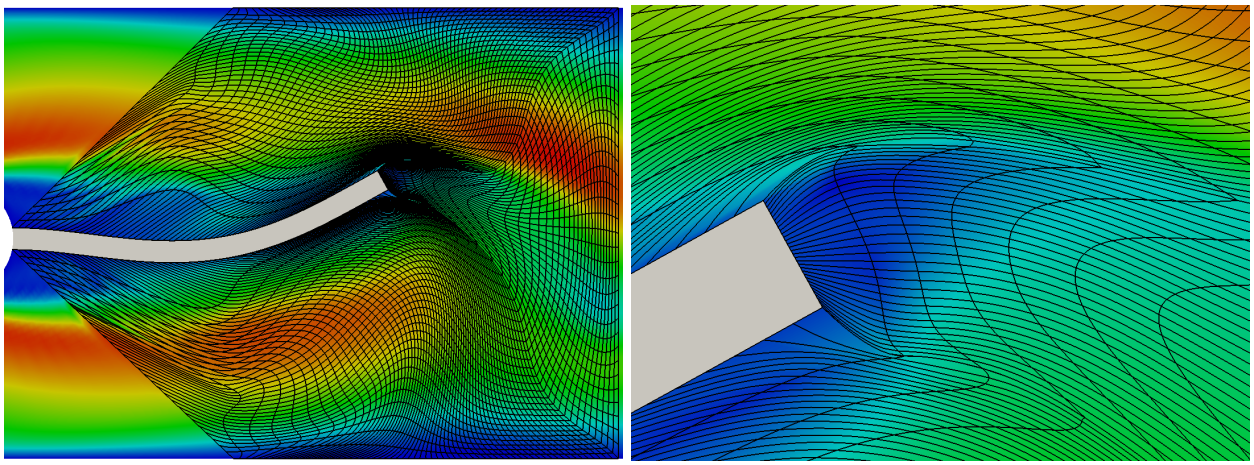

Figure 4.18. Benchmark FSI2: fluid mesh with the ILE method during the last oscillation period.

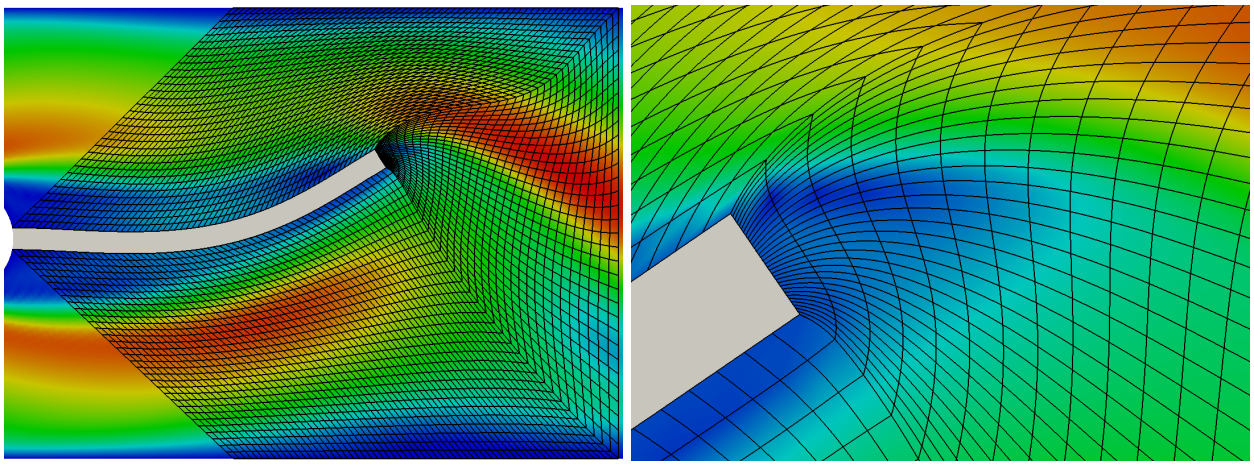

Figure 4.19. Benchmark FSI2: fluid mesh with the TINE method during last oscillation period.

Although both the ILE and TINE methods are able to handle large deformations occurring during the simulation, their performance with respect to the fluid mesh quality differs substantially. Similarly to the simplified FSI-like scenario in Section 4.4.4, the TINE method shows no signs of accumulated distortion. Even at the end of the simulation, the fluid mesh retains its initial quality, see Figure 4.19. In contrast to that, the ILE method results in noticeable mesh quality deterioration over the course of the simulation. Figure 4.18 shows the corresponding fluid mesh at the end of the simulation. Despite staying valid, the severely distorted mesh affects the results of the simulation. The simulations with the ILE and TINE method show the same behavior until $t \approx 10$s, after

which the ILE simulation starts to shows signs of damping, see Figures 4.14 and Figures 4.15. It is quite likely that the simulation would fail soon if it continued beyond 15s. Figure 4.20 compares behavior of the ALE displacement norm for the ILE and TINE methods. Note that for the latter, the norm eventually stabilizes and behaves periodically. However, this stable behavior comes at a price of increased computational cost. The TINE mesh deformation method consumes roughly twice as much computational time as the ILE method. When considered from the perspective of total numerical effort required for the FSI simulation, this translates into about a 10% increase in computational time. Figure 4.21 presents a breakdown of computational time for the simulations with the ILE and TINE methods.

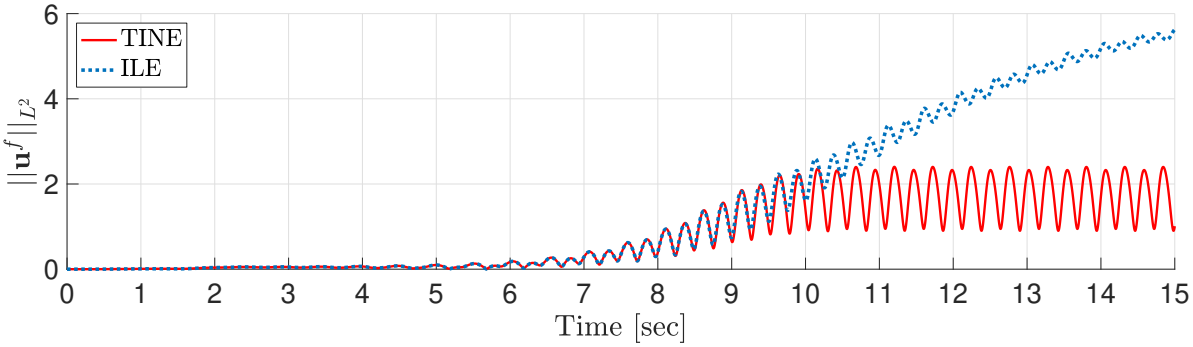

Figure 4.20. Benchmark FSI2: ALE displacement norm.

h-lvl	$\Delta t[s]$	#DoF-ALE	#DoF-Flow	#DoF-Beam	T_{comp}
4	0.01	2624	6838	1188	2h13m
4	0.0025	2624	6838	1188	8h17m
5	0.01	10368	26334	4420	9h26m
5	0.0025	10368	26334	4420	1d11h
6	0.0025	41216	103342	17028	5d18h
Ref	0.001	N/A	304128	N/A	N/A

Table 4.7. Benchmark FSI2: computational cost with the TINE method.

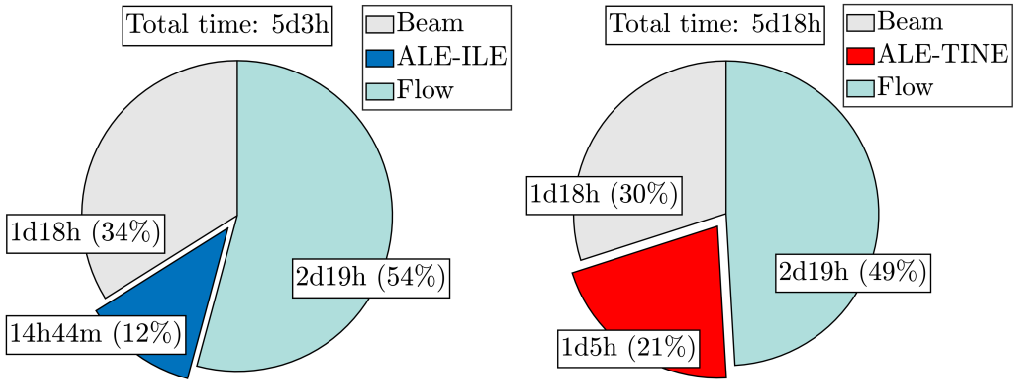

Figure 4.21. Benchmark FSI2: computational time breakdown with the ILE method (left) and the TINE method (right).

4.5 Summary

In this chapter, I have studied mesh deformation appearing in fluid-structure interaction (FSI) problems. In particular, I have considered an arbitrary Lagrangian-Eulerian (ALE) formulation of FSI. In the ALE formulation, motion of the fluid domain is described by an ALE mapping, which can be interpreted as a deformation of a computational mesh in the fluid domain. Among the state-of-the art methods for constructing ALE mappings are the harmonic extension method and the incremental linear elasticity (ILE) method, which appears in Chapter 3 in the context of the mesh deformation approach to domain parametrization. All methods can be enhanced by means of the Jacobian-based local stiffening, which redistributes the deformation and allows the fluid mesh to withstand larger deformations. The core idea of the chapter is to investigate a possibility and potential advantages of using the tangential incremental nonlinear elasticity (TINE) method, which was introduced in Chapter 3, to construct ALE mappings.

The performance of all mesh deformation methods has been first studied on an idealized FSI-like benchmark with no fluid simulations involved. Instead, the motion of the fluid domain is driven purely by free oscillations of an elastic beam. Despite being computationally inexpensive, this setting mimics well the fluid domain motion happening in real FSI simulations. I have found that without local stiffening, all methods perform similarly poorly and can handle only small deformation before the fluid mesh becomes invalid. On the other hand, with local stiffening the performance of all methods can be greatly improved. The ILE and the TINE methods benefit more than other methods from the local stiffening and can handle substantially larger deformations. Unfortunately for the ILE method, however, I have observed that it results in an irreversible deformation of the fluid mesh which accumulates over time. I refer to this effect as accumulated distortion. As the stiffening degree increases, the rate of distortion accumulation grows. On the contrary, the TINE method has shown no signs of accumulated distortion. However, it is roughly twice as computationally expensive as the ILE method.

After that, I have compared all mesh deformation methods on a classic FSI benchmark—vortex shedding past a flexible beam. To that end, I have developed an isogeometric FSI solver based on the partitioned approach with strong coupling. The FSI solver includes subsolvers for the incompressible Naiver-Stokes equations and for the time-dependent equations of nonlinear elasticity. The former is based on mixed isogeometric elements and uses the one-step IMEX θ-scheme for time integration; the latter uses the Newmark method for time integration. I have validated both subsolvers as well as the overarching FSI solver on several benchmarks and have obtained results which are in good agreement with the reference values.

Of all considered mesh deformation methods, only the ILE and TINE method are able to handle motion of the fluid domain in the FSI simulation. However, just like on the simplified benchmark, the ILE method has resulted in the accumulated distortion effect which has been substantial enough to significantly alter the simulation results. As a result, the simulation with ILE method has produced a damped oscillatory behavior of the system and was dangerously close to failing. In contrast to that, the TINE method again has shown no signs of accumulated distortion. The simulation with TINE method has successfully reproduced a fully developed periodic behavior of the system and was able to maintain it for an extended period of time. Since the TINE method is roughly twice as expensive as the ILE method, it results in a 10% increase in the total computational time required for the FSI simulation.

Overall, the TINE method appears to be a good options for FSI simulations where large deformations are expected. Its ability to deform the fluid mesh without introducing accumulated distortion is definitely worth a 10% increase in total computational time in comparison to the ILE method. Nevertheless, the ILE method is not necessarily bad; its poor performance in this work has

been largely caused by a high stiffening degree. It is possible that with a different parametrization of the fluid domain, less local stiffening would be necessary, and the ILE method would result in less accumulated distortion. On the other hand, the parametrization of the fluid domain that I have used in this work is easy to construct and has few patches. If a mesh deformation method can operate even on such a "naive" parametrization, it is a strong benefit. In other words, the main advantage of the TINE method over the ILE method may be its more modest requirements on the domain parametrization.

5 | Conclusion

5.1 Summary

In this work, I have developed a novel mesh deformation method which allows computational meshes to withstand large deformations without experiencing quality deterioration. The proposed method, called TINE (tangential incremental nonlinear elasticity), is based on numerical path following applied to the equations of nonlinear elasticity. By employing a logarithmic variation of the neo-Hookean hyperelastic material law, it preserves mesh elements from becoming inverted. Moreover, the TINE method is surprisingly inexpensive; it only requires solution of a linear system of algebraic equations per deformation step. In addition to its superior qualities, the method is quite flexible and can be further augmented by tuning Poisson's ratio of the underlying material model or by means of local stiffening.

Although all mesh-based numerical methods for solving partial differential equations can benefit from the proposed mesh deformation method, it has been primarily developed and applied in the context of isogeometric finite elements. One of the areas of active research in IGA is volumetric domain parametrization which concerns itself with the following problem: how does one construct a parametrization for a given computational domain provided only description of its boundary? In this work, I have proposed a new solution approach to this problem based on the TINE method where a parametrization for the target domain is constructed as a deformed configuration of a simplified domain. I refer to this as a mesh deformation approach.

To obtain the simplified domain, the boundary of the target domain is first projected in the L^2-sense onto a coarse NURBS basis. Then, the resulting domain can be parametrized using any computationally inexpensive method. Although the Coons patch or elliptic grid generation often suffice, for some domains I have obtained the best results by applying the mesh deformation approach recursively. At the deformation step, Poisson's ratio used in the material law plays a crucial role. It defines resistance of the material to volumetric changes and, thus, affects the extent to which the mesh elements change in size after the deformation. With values close to 0.5, the deformed mesh largely inherits the element size distribution of the initial mesh. However, high values of Poisson's ratio also make the displacement-only formulation of nonlinear elasticity numerically unstable. In practice, values between 0.45 and 0.49 are optimal. I have compared the mesh deformation approach to elliptic grid generation and the optimization-based methods on two 2D single-patch examples. For both examples, it has produced superior parametrizations. In terms of computational complexity, the mesh deformation approach is about twice as expensive as elliptic grid generation but is more than five times cheaper than the optimization-based methods. The approach is not bound to a 2D single-patch scenario and can also be applied to construct 3D and multi-patch parametrizations.

Experiments with the mesh deformation approach have revealed that the TINE method shows behavior similar to one of the state-of-are-art mesh update methods in fluid-structure interaction (FSI): the incremental linear elasticity (ILE) method. In order to study the performance of the TINE

method in the context of FSI, I have reproduced a classic 2D benchmark using an isogeometric partitioned solver with strong coupling. The results of the benchmark compare well against the reference values. Using the benchmark, I have compared the TINE method against other mesh updates methods, including ILE and harmonic extension. Additionally, I have studied the effect of the Jacobian-based local stiffening on all mesh update methods. According to the results, all mesh update methods perform similarly without local stiffening and can handle only small deformations. However, provided enough local stiffening, all methods can handle much larger deformations, but only the ILE and TINE methods can handle deformations arising in the benchmark.

Although similar in their short term behavior, the ILE and TINE methods demonstrate different behavior in the long run. In particular, the ILE method with the Jacobian-based local stiffening results in accumulation of mesh distortion over the course of time. This effect can significantly change the simulation results. The rate of distortion accumulation increases with higher degrees of local stiffening. In contrast to that, no signs of accumulated distortion have been observed using the TINE method. Therefore, TINE allows to perform FSI simulations over longer periods of time without affecting the results. Regarding the computational cost, the TINE method is roughly twice as expensive as the ILE method but leads only to a small increase in the total computational time for the FSI simulation.

5.2 Outlook

Throughout this thesis, I have often juxtaposed the proposed TINE mesh deformation method and the well-known in the scientific community ILE method. While behavior of the former can be explained by the theory of nonlinear elasticity and continuation methods, the surprisingly good behavior of the latter method is not so well understood. Why are its results similar to solutions of the equations of nonlinear elasticity with the neo-Hookean material law? Why are they not similar to solutions of the equations of nonlinear elasticity with the St. Venant-Kirchhoff material law? Why does the ILE method seem to prevent mesh elements from becoming inverted? An attempt to answer these questions could represent an interesting research direction.

The mesh deformation approach has proven to be a robust and efficient method to construct domain parametrizations. Although it can be applied in the multi-patch scenario, the segmentation step (which provides the multi-patch structure of the domain) currently has to be performed manually. However, it may be possible to combine the mesh deformation approach and the segmentation step in the following way: first, the target domain is simplified. Next, the segmentation step is performed on the simplified domain. Finally, the segmented simplified domain is deformed, which grants a multi-patch parametrization for the target domain. If applying the segmentation step to the simplified domain is easier that to the target domain, the described approach can be effective.

In this work, I have compared several mesh update method for FSI problems. However, I have omitted the bi-harmonic extension method due to its complexity. According to the literature, it is able to handle large deformations but tends to be extremely computationally expensive. To obtain a full picture of all mesh update methods, it would be interesting to include the bi-harmonic extension method into the comparison. Another promising direction related to FSI is development of alternative local stiffening methods. Although the Jacobian-based local stiffening is used in many FSI application, it is chiefly responsible for the accumulated distortion effect which I have studied in this work. Alternative local stiffening methods may solve the problem of accumulated distortion for the ILE method. It would also be interesting to implement a monolithic FSI solver based on the TINE mesh update method. To the best of my understanding, the ILE method is only viable in the partitioned approach. Therefore, the TINE method may be unrivaled.

Finally, another area where mesh deformation effects are relevant is shape optimization. Here, each iteration of the optimization loop yields an update of the domain's boundary. This updates need to be extended into the domain's interior in order to update the computational mesh and compute the next iteration. It would be interesting to study the nature of the boundary updates and find out if the field of shape optimization can benefit from the TINE method.

Bibliography

[1] D. Turim and D. Gates. Building the 787 dreamliner: a timeline. *Seattle Times*, 2009.

[2] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[3] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[4] J. A. Cottrell, A. Reali, Y. Bazilevs, and T. J. R. Hughes. Isogeometric analysis of structural vibrations. *Computer Methods in Applied Mechanics and Engineering*, 195(41-43):5257–5296, 2006.

[5] T. Elguedj, Y. Bazilevs, V. M. Calo, and T. J. R. Hughes. B-bar and F-bar projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40):2732–2762, 2008.

[6] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff–Love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49-52):3902–3914, 2009.

[7] Y. Bazilevs and T. J. R. Hughes. NURBS-based isogeometric analysis for the computation of flows about rotating components. *Computational Mechanics*, 43(1):143–150, 2008.

[8] A. Tagliabue, L. Dede, and A. Quarteroni. Isogeometric analysis and error estimates for high order partial differential equations in fluid dynamics. *Computers & Fluids*, 102:277–303, 2014.

[9] B. S. Hosseini, M. Möller, and S. Turek. Isogeometric analysis of the Navier–Stokes equations with Taylor–Hood B-spline elements. *Applied Mathematics and Computation*, 267:264–281, 2015.

[10] Y. Bazilevs, V. M. Calo, Y. Zhang, and T. J. R. Hughes. Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. *Computational Mechanics*, 38(4-5):310–322, 2006.

[11] Y. Bazilevs, K. Takizawa, and T. E. Tezduyar. *Computational fluid-structure interaction: methods and applications*. John Wiley & Sons, 2013.

[12] M.-C. Hsu, D. Kamensky, Y. Bazilevs, M. S. Sacks, and T. J. R. Hughes. Fluid–structure interaction analysis of bioprosthetic heart valves: significance of arterial wall deformation. *Computational Mechanics*, 54(4):1055–1071, 2014.

[13] A. Buffa, G. Sangalli, and R. Vázquez. Isogeometric analysis in electromagnetics: B-splines approximation. *Computer Methods in Applied Mechanics and Engineering*, 199(17-20):1143–1152, 2010.

[14] D. M. Nguyen, A. Evgrafov, and J. Gravesen. Isogeometric shape optimization for electromagnetic scattering problems. *Progress In Electromagnetics Research*, 45:117–146, 2012.

[15] H. Gómez, V. M. Calo, Y. Bazilevs, and T. J. R. Hughes. Isogeometric analysis of the Cahn–Hilliard phase-field model. *Computer Methods in Applied Mechanics and Engineering*, 197(49-50):4333–4352, 2008.

[16] L. Dedè, M. J. Borden, and T. J. R. Hughes. Isogeometric analysis for topology optimization with a phase field model. *Archives of Computational Methods in Engineering*, 19(3):427–465, 2012.

[17] Y. Bazilevs, L. Beirao da Veiga, J. A. Cottrell, T. J. R. Hughes, and G. Sangalli. Isogeometric analysis: approximation, stability and error estimates for h-refined meshes. *Mathematical Models and Methods in Applied Sciences*, 16(07):1031–1090, 2006.

[18] T. J. R. Hughes, J. A. Evans, and A. Reali. Finite element and NURBS approximations of eigenvalue, boundary-value, and initial-value problems. *Computer Methods in Applied Mechanics and Engineering*, 272:290–320, 2014.

[19] H.-J. Kim, Y.-D. Seo, and S.-K. Youn. Isogeometric analysis for trimmed CAD surfaces. *Computer Methods in Applied Mechanics and Engineering*, 198(37-40):2982–2995, 2009.

[20] R. Schmidt, R. Wüchner, and K.-U. Bletzinger. Isogeometric analysis of trimmed NURBS geometries. *Computer Methods in Applied Mechanics and Engineering*, 241-244:93–111, 2012.

[21] C. S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[22] D. Schillinger, L. Dede, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, and T. J. R. Hughes. An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces. *Computer Methods in Applied Mechanics and Engineering*, 249:116–150, 2012.

[23] J. Kiendl, Y. Bazilevs, M.-C. Hsu, R. Wüchner, and K.-U. Bletzinger. The bending strip method for isogeometric analysis of Kirchhoff–Love shell structures comprised of multiple patches. *Computer Methods in Applied Mechanics and Engineering*, 199(37-40):2403–2416, 2010.

[24] J. Kiendl, F. Auricchio, L. Beirão da Veiga, C. Lovadina, and A. Reali. Isogeometric collocation methods for the Reissner-Mindlin plate problem. *Computer Methods in Applied Mechanics and Engineering*, 284:489–507, 2015.

[25] J. Gravesen, A. Evgrafov, D.-M. Nguyen, and P. Nørtoft. Planar parametrization in isogeometric analysis. In *International Conference on Mathematical Methods for Curves and Surfaces 2012*, pages 189–212. Springer, 2014.

[26] M. Pauley, D.-M. Nguyen, D. Mayer, J. Špeh, O. Weeger, and B. Jüttler. The isogeometric segmentation pipeline. In *Isogeometric Analysis and Applications 2014*, pages 51–72. Springer, 2015.

[27] G. Farin and D. Hansford. Discrete Coons patches. *Computer Aided Geometric Design*, 16(7):691–700, 1999.

[28] S. K. Khattri. Grid generation and adaptation by functionals. *Computational & Applied Mathematics*, 26(2):235–249, 2007.

[29] J. Hinz, M. Möller, and C. Vuik. Elliptic grid generation techniques in the framework of isogeometric analysis applications. *Computer Aided Geometric Design*, 2018.

[30] P. Wriggers. *Nonlinear finite element methods*. Springer Science & Business Media, 2008.

[31] A. Shamanskiy, M. H. Gfrerer, J. Hinz, and B. Simeon. Isogeometric parametrization inspired by large elastic deformation. *Computer Methods in Applied Mechanics and Engineering*, 363:112920, 2020.

[32] T. Richter. *Numerical methods for fluid-structure interaction problems*. Institute for Applied Mathematics, University of Heidelberg, Germany, 2010.

[33] T. E. Tezduyar, M. Behr, S. Mittal, and A. A. Johnson. Computation of unsteady incompressible flows with the stabilized finite element methods: space-time formulations, iterative strategies and massively parallel implementations. In *ASME Pressure Vessels & Piping Conference 1992*, pages 7–24. American Society of Mechanical Engineers Digital Collection, 1992.

[34] T. E. Tezduyar, S. Aliabadi, M. Behr, A. A. Johnson, and S. Mittal. Parallel finite-element computation of 3D flows. *Computer*, 26(10):27–36, 1993.

[35] A. A. Johnson and T. E. Tezduyar. Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. *Computer Methods in Applied Mechanics and Engineering*, 119(1-2):73–94, 1994.

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

[37] T. E Tezduyar and S. Sathe. Modelling of fluid–structure interactions with the space–time finite elements: solution techniques. *International Journal for Numerical Methods in Fluids*, 54(6-8):855–900, 2007.

[38] S. Turek and J. Hron. Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. In *Fluid-Structure Interaction*, pages 371–385. Springer, 2006.

[39] L. Piegl and W. Tiller. *The NURBS book*. Springer, 1997.

[40] P. J. Schenider. NURB curves: a guide for the unitiated. *Develop*, 25, 1996.

[41] W. Dornisch, G. Vitucci, and S. Klinkel. The weak substitution method—an application of the mortar method for patch coupling in NURBS-based isogeometric analysis. *International Journal for Numerical Methods in Engineering*, 103:205–234, 2015.

[42] C. Hofer and I. Toulopoulos. Discontinuous Galerkin isogeometric analysis of elliptic problems on segmentations with non-matching interfaces. *Computers & Mathematics with Applications*, 72(7):1811–1827, 2016.

[43] D. Burkhart, B. Hamann, and G. Umlauf. Isogeometric finite element analysis based on Catmull-Clark subdivision solids. In *Computer Graphics Forum*, pages 1575–1584. Wiley Online Library, 2010.

[44] N. Jaxon and X. Qian. Isogeometric analysis on triangulations. *Computer-Aided Design*, 46:45–57, 2014.

[45] D. C. Thomas, L. Engvall, S. K. Schmidt, K. Tew, and M. A. Scott. U-splines: splines over unstructured meshes. `https://coreform.com/usplines`, 2018.

[46] T. Varady, R. R. Martin, and J. Cox. Reverse engineering of geometric models—an introduction. *Computer-Aided Design*, 29(4):255–268, 1997.

[47] R. Schneiders. Algorithms for quadrilateral and hexahedral mesh generation. In *Proceedings of the VKI Lecture Series on Computational Fluid Dynamic*. VKI Lecture Series, 2000.

[48] H. J. Fogg, C. G. Armstrong, and T. T. Robinson. Enhanced medial-axis-based block-structured meshing in 2-D. *Computer-Aided Design*, 72:87–101, 2016.

[49] J. Xu, F. Chen, and J. Deng. Two-dimensional domain decomposition based on skeleton computation for parameterization and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:541–555, 2015.

[50] N. Kowalski, F. Ledoux, and P. Frey. A PDE based approach to multidomain partitioning and quadrilateral meshing. In *Proceedings of the 21st International Meshing Roundtable*, pages 137–154. Springer, 2013.

[51] A.-V. Vuong, C. Giannelli, B. Jüttler, and B. Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 200(49):3554–3567, 2011.

[52] C. Giannelli, B. Jüttler, and H. Speleers. THB-splines: the truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29(7):485–498, 2012.

[53] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):229–263, 2010.

[54] M. R. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):264–275, 2010.

[55] K. A. Johannessen, T. Kvamsdal, and T. Dokken. Isogeometric analysis using LR B-splines. *Computer Methods in Applied Mechanics and Engineering*, 269:471–514, 2014.

[56] F. Calabro, G. Sangalli, and M. Tani. Fast formation of isogeometric galerkin matrices by weighted quadrature. *Computer Methods in Applied Mechanics and Engineering*, 316:606–622, 2017.

[57] F. Auricchio, L. Beirão da Veiga, T. J. R. Hughes, A. Reali, and G. Sangalli. Isogeometric collocation methods. *Mathematical Models and Methods in Applied Sciences*, 20(11):2075–2107, 2010.

[58] A. Collin, G. Sangalli, and T. Takacs. Analysis-suitable G^1 multi-patch parametrizations for C^1 isogeometric spaces. *Computer Aided Geometric Design*, 47:93–113, 2016.

[59] K. Birner, B. Jüttler, and A. Mantzaflaris. Approximation power of G^1-smooth isogeometric splines on volumetric two-patch domains. `https://hal.inria.fr/hal-02275654/document`, 2018.

[60] B. Jüttler, U. Langer, A. Mantzaflaris, S. Moore, and W. Zulehner. Geometry + simulation modules: implementing isogeometric analysis. In *Proceedings in Applied Mathematics and Mechanics*, pages 961–962. Wiley, 2014.

[61] D. Kourounis, A. Fuchs, and O. Schenk. Towards the next generation of multiperiod optimal power flow solvers. *IEEE Transactions on Power Systems*, PP(99):1–10, 2018.

[62] C. Arioli, A. Shamanskiy, S. Klinkel, and B. Simeon. Scaled boundary parametrizations in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 349:576–594, 2019.

[63] D. Braess. *Finite elements: theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, 2007.

[64] W. J. Gordon and C. A. Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *International Journal for Numerical Methods in Engineering*, 7(4):461–477, 1973.

[65] J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

[66] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[67] M. Harmel, R. A. Sauer, and D. Bommes. Volumetric mesh generation from T-spline surface representations. *Computer-Aided Design*, 82:13–28, 2017.

[68] P.-O. Persson and J. Peraire. Curved mesh generation and mesh refinement using Lagrangian solid mechanics. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, page 949, 2009.

[69] L. M. Bernal, V. M. Calo, N. Collier, G. A. Espinosa, F. Fuentes, and J. C. Mahecha. Isogeometric analysis of hyperelastic materials using PetIGA. In *Procedia Computer Science*. Elsevier, 2013.

[70] P. Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*. Springer Science & Business Media, 2011.

[71] A. Shamanskiy and B. Simeon. Isogeometric simulation of thermal expansion for twin screw compressors. *IOP Conference Series: Materials Science and Engineering*, 425:012–031, 2018.

[72] M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, and A.-V. Vuong. Swept volume parameterization for isogeometric analysis. In *IMA International Conference on Mathematics of Surfaces*, pages 19–44. Springer, 2009.

[73] U. Küttler and W. A. Wall. Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43(1):61–72, 2008.

[74] V. John. *Finite element methods for incompressible flow problems*. Springer, 2016.

[75] A. Bressan and G. Sangalli. Isogeometric discretizations of the Stokes problem: stability analysis by the macroelement technique. *IMA Journal of Numerical Analysis*, 33(2):629–651, 2013.

[76] M. ur Rehman, C. Vuik, and G. Segal. A comparison of preconditioners for incompressible Navier–Stokes solvers. *International Journal for Numerical methods in fluids*, 57(12):1731–1751, 2008.

[77] P. Benner and J. Heiland. Time-dependent Dirichlet conditions in finite element discretizations. *ScienceOpen Research*, pages 1–18, 2015.

[78] M. R. Dörfel. *Fluid-structure interaction: a differential-algebraic approach and acceleration techniques for strong coupling*. VDI Verlag, 2011.

[79] T. Richter and T. Wick. Finite elements for fluid–structure interaction in ALE and fully Eulerian coordinates. *Computer Methods in Applied Mechanics and Engineering*, 199(41-44):2633–2642, 2010.

[80] Y. Wu and X.-C. Cai. A fully implicit domain decomposition based ALE framework for three-dimensional fluid–structure interaction with application in blood flow computation. *Journal of Computational Physics*, 258:524–537, 2014.

Academic Curriculum Vitae

2000-2010 **Complete secondary education**
State secondary school №315, Moscow
Specialization: mathematics and physics

2010-2015 **Specialist in mathematics**
Department of Mechanics and Mathematics, Moscow Lomonosov State University
Thesis: *On Application of Differential Equations to the Solution of Audio-Information Transmission Problem*

2016-2020 **Doctor rerum naturalium (Dr. rer. nat.)**
Department of Mathematics, Technical University of Kaiserslautern
Dissertation: *Mesh Deformation in the Context of Isogeometric Analysis*

Wissenschaftlicher Lebenslauf

2000-2010 **Vollständige allgemeine Mittelschulbildung**
Staatliche Mittelschule №315, Moskau
Schwerpunkt: Mathematik und Physik

2010-2015 **Spezialist für Mathematik**
Fachbereich Mechanik und Mathematik, Lomonossow-Universität, Moskau
Diplomarbeit: *On Application of Differential Equations to the Solution of Audio-Information Transmission Problem*

2016-2020 **Doktor der Naturwissenschaften**
Fachbereich Mathematik, Technische Universität Kaiserslautern
Doktorarbeit: *Mesh Deformation in the Context of Isogeometric Analysis*