Westfälische
Wilhelms-Universität
Münster

Unfitted discontinuous Galerkin schemes for applications with PDEs on complex-shaped surfaces

Dissertation

Sebastian Westerheide

– 2018 –

wissen.leben
WWU Münster

Angewandte
Mathematik
Münster

Mathematik

Unfitted discontinuous Galerkin schemes for applications with PDEs on complex-shaped surfaces

Inauguraldissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften im Fachbereich
Mathematik und Informatik
der Mathematisch-Naturwissenschaftlichen Fakultät
der Westfälischen Wilhelms-Universität Münster

vorgelegt von
Sebastian Westerheide
aus Münster

Münster, 2018

Dekan: Prof. Dr. Xiaoyi Jiang
Erstgutachter: Prof. Dr. Christian Engwer (WWU Münster)
Zweitgutachter: Prof. Dr. Axel Voigt (TU Dresden)

Tag der mündlichen Prüfung: 19.06.2018
Tag der Promotion: 19.06.2018

Abstract

English version

The unfitted discontinuous Galerkin (UDG) method is a computational approach to solving partial differential equations (PDEs) that are posed on flat spatial geometries known as bulk domains. It allows for conservative discontinuous Galerkin discretizations based on cut cell meshes and hence is particularly suitable for continuity equations on complex-shaped bulk domains, especially on those arising from imaging data. It has been successfully applied in various applications, ranging from computational fluid dynamics to source analysis in neuroscience.

In this thesis, we show how the method can be transferred to PDEs on curved spaces that are known as hypersurfaces or surfaces. We introduce UDG schemes for a class of model problems which is biologically motivated and comprises continuity equations on a bulk domain and its surface. Our schemes are concerned with the model problems' formulation for static geometries, and also with its generalization for evolving geometries.

The underlying approaches combine ideas of methods that extend surface PDEs to higher-dimensional bulk domains with concepts of trace finite element methods, which employ extended solution spaces, but still focus on the sharp representation of the hypersurface only. The hybrid character of our approaches results in schemes with favorable properties, such as the recovery of discrete analogues to conservation laws that are embedded in the PDEs, and the reusability of existing implementations of the UDG method for bulk PDEs. At the same time, a high degree of flexibility with respect to the shape of the geometrical setup and its evolution is achieved by using a level set representation of the geometry.

We present theoretical investigations and numerical results which demonstrate that our computational approaches to surface PDEs yield promising schemes for the considered class of model problems.

German version

Die UDG-Methode ist ein numerischer Ansatz zur Lösung von partiellen Differentialgleichungen (PDEs) auf Geometrien, die im mathematischen Sinne als Gebiete in euklidischen Räumen dargestellt werden können. Dieser Ansatz ermöglicht konservative Discontinuous-Galerkin-Diskretisierungen auf Basis von Cut-Cell-Gittern und eignet sich daher besonders im Rahmen von Kontinuitätsgleichungen auf Gebieten mit komplizierten Rändern, vor allem wenn sich

diese aus Bilddaten ergeben. Die Methode wurde bereits in verschiedenen Anwendungen erfolgreich eingesetzt, etwa zur numerischen Strömungssimulation oder zur Quellenanalyse in der neurowissenschaftlichen Forschung.

In dieser Arbeit zeigen wir, wie sich die Methode auf PDEs übertragen lässt, welche auf gekrümmten, als Hyperflächen oder Oberflächen bekannten Räumen formuliert sind. Wir stellen UDG-Verfahren für eine biologisch motivierte Klasse von Modellproblemen vor, die Kontinuitätsgleichungen in einem Gebiet und auf dessen Oberfläche umfasst. Unsere Verfahren zielen sowohl auf jene Formulierung dieser Klasse ab, welche sich für statische Geometrien ergibt, als auch auf die entsprechende Verallgemeinerung für zeitlich veränderliche Geometrien.

Die dem zugrundeliegenden Ansätze kombinieren Ideen von Methoden, die auf der Erweiterung von PDEs auf Oberflächen auf höherdimensionale Gebiete basieren, mit Konzepten von Spur-Finite-Elemente-Methoden, die zwar Lösungsräume auf höherdimensionalen Gebieten einsetzen, sich aber trotzdem nur auf die eigentliche Oberfläche konzentrieren. Der daraus hervorgehende hybride Charakter unserer Ansätze resultiert in Verfahren mit vorteilhaften Eigenschaften. Unsere Verfahren bilden etwa in die PDEs verankerte Erhaltungseigenschaften im diskreten Sinne ab und sie ermöglichen es, bestehende Implementierungen der ursprünglichen UDG-Methode für PDEs auf Gebieten wiederzuverwenden. Ein hohes Maß an Flexibilität hinsichtlich der Gestalt des geometrischen Aufbaus und ihrer zeitlichen Entwicklung wird dabei durch den Einsatz der Level-Set-Methode erreicht.

Mit Hilfe theoretischer Untersuchungen und numerischer Studien zeigen wir, dass unsere Ansätze zur numerischen Behandlung von PDEs auf Oberflächen vielversprechende Verfahren für die betrachtete Klasse von Modellproblemen liefern.

Acknowledgements

I would like to thank my supervisor Christian Engwer for introducing me to his interesting research areas and for offering me to work on this fascinating topic. I am especially thankful for his patience throughout the years and for giving me the right amount of guidance in the first part of my research and the right amount of free development afterwards.

Furthermore, I am grateful to my colleagues and former colleagues at the Institute for Analysis and Numerics at WWU Münster for creating a pleasant working atmosphere, but also for those fun times which we had during breaks. This particularly applies to members of the workgroup "Anwendungen von partiellen Differentialgleichungen". You guys helped me a lot in discussing scientific ideas and their implementational difficulties in `DUNE`.

I extend my thanks to Marten Bornmann, Nils-Arne Dreier, Stephan Förster, Tobias Komurka, Andreas Nüßing, Stefan Ruhkamp, Sophie Schrader, and Johannes Vorwerk who kindly proofread different parts of this thesis.

Most importantly, I would like to express my gratitude to my friends and my family, especially Katryn Chamot and Stephan Förster, for their valuable support in putting me back on the right track whenever I needed to fight self-doubts and negative feelings about my work. Besides being grateful for all her moral and emotional support, I sincerely thank Katryn for her understanding and for covering my back while I was finally writing up the thesis.

This thesis was supported by the Deutsche Forschungsgemeinschaft (DFG), grant no. EN 1042/4-1: "Massenerhaltende Kopplung von Oberflächen- und Volumenprozessen auf impliziten, zeitabhängigen Gebieten".

Contents

List of Figures

List of Tables

1. Introduction

Since the widespread availability of computers at research laboratories and universities, computer simulations have taken their place alongside theory and practical experiments as the third pillar of scientific research. They allow for gaining new insights, even if properties or structures are experimentally too difficult to access, or if experiments are so expensive that only a small amount of them can be performed. In addition, scientific theories can be tested and refined systematically by comparing predictions derived from simulation results with results that are obtained in tailor-made experiments which focus on these predictions.

Mathematical modeling, bulk domains and embedded structures

Simulation science typically relies on mathematical modeling of processes that are observed in practical experiments performed in science and engineering. Its applications frequently involve processes in flat spatial geometries which mathematical models can represent as connected, open subsets of Euclidean space of a certain dimension. Such mathematical objects can be referred to as *bulk domains*. Being open subsets, they inherit their dimension from the Euclidean space in which they are contained.

Applications with processes in flat spatial geometries, or with processes in geometries which can be considered as being flat by reasonable modeling assumptions, can be found in many textbooks on mathematical modeling. Chorin and Marsden (2000) consider applications from fluid dynamics, e.g. flows in pipes and flows around obstacles. Various applications from different fields like acoustics or atmospheric dispersion are treated in Tayler (1986), e.g. submarine detection and smoke dispersion from a high chimney. In Friedman and Littman (1994), real-world applications from chemistry, electrophotography and other fields are considered, particularly oxidation reactions in catalytic converters in the exhaust system of cars and the capturing process of electric images in photocopy machines.

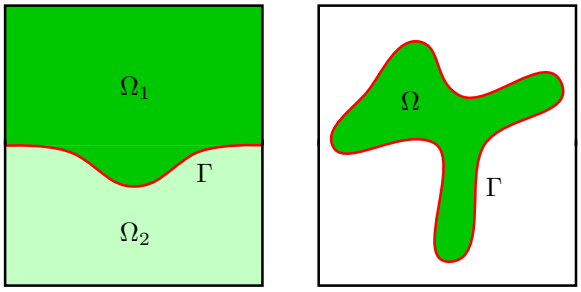
In many of those applications, structures of a certain dimension that are embedded in some higher-dimensional Euclidean space, such as curved surfaces, membranes or interfaces, also play an important role. Moreover, it can be necessary to consider processes on those embedded structures instead of processes in flat spatial geometries to obtain a reliable mathematical model.

This is the case, for instance, for applications from biology (e.g. transport of liquids and surface active agents (surfactants) through the lung airways and along their walls (Halpern et al., 1998), cell migration and chemotaxis (Neilson

et al., 2011b,a; Elliott et al., 2012), pattern formation on developing plant tips (Nagata et al., 2013) or on the skin of developing fishes (Venkataraman et al., 2011), as well as pattern formation on other biological surfaces and tumor growth (Barreira et al., 2011)). Other applications can be found in materials science (e.g. spreading of thin fluid films or coatings on curved substrates (Roy et al., 2002), ice formation on surfaces (Myers et al., 2002; Myers and Charpin, 2004), and phase separation of polymers on a surface (Tang et al., 2005)). Further applications are considered in image processing (e.g. processing of brain image data (Toga, 1998; Mémoli et al., 2004)), or appear in computer graphics (e.g. vector field visualization on surfaces (Diewald et al., 2000), virtual weathering (Dorsey and Hanrahan, 2000), and texture generation on surfaces (Turk, 1991; Witkin and Kass, 1991)). Some applications also arise from applied mathematics itself (e.g. pattern formation on the sphere and on other curved mathematical objects (Varea et al., 1999; Chaplain et al., 2001; Plaza et al., 2004; Rozada et al., 2014)).

Due to recent technological progress, processes on embedded structures increasingly gain attention, especially in combination with processes in flat spatial geometries. On the one hand, in practical experiments, modern imaging techniques allow for in-depth observations which underline the importance of processes on embedded structures. On the other hand, current computer hardware provides enough computational power to perform detailed simulations which include the influence of processes associated with either type of geometry at the same time. As a consequence, modelers have started to deal with coupled processes in flat spatial geometries and on embedded structures, rendering simulations for applications that entail both types of processes more and more realistic.

Applications of this kind include crystal growth in materials science (Kwon and Derby, 2001) and proton diffusion along biological membranes in physics (Medvedev and Stuchebrukhov, 2011, 2013), for example. Furthermore, many applications arise in mathematical biology and cell biology, e.g. pattern formation on biological membranes (Levine and Rappel, 2005), cell migration and chemotaxis (Marth and Voigt, 2014; MacDonald et al., 2016), lipid raft formation in cell membranes (Garcke et al., 2015), fluorescence loss in photobleaching which is used for examining the movement of molecules inside cells and their membranes (Novak et al., 2007), and cell polarization (Goryachev and Pokhilko, 2008; Rätz and Röger, 2012, 2014; Giese, Eigel, Westerheide, Engwer and Klipp, 2015a; Emken, 2016). Other important applications can be found in fluid dynamics, e.g. two-phase flows with soluble or insoluble surface active agents (surfactants) on the fluid–fluid interface (James and Lowengrub, 2004; Booty and Siegel, 2010; Teigen et al., 2011; Ganesan and Tobiska, 2012; Ganesan et al., 2012; Hahn et al., 2013), and surfactants in general (Hahn et al., 2014). A purely mathematical investigation of bulk-mediated surface diffusion is presented in Chechkin et al. (2012). The influence of bulk domain processes on pattern formation on surfaces is investigated mathematically in Madzvamuse et al. (2015) and Madzvamuse and Chung (2016).



(a) A fluid–fluid interface. (b) A cell membrane.

Figure 1.1.: Structures embedded in some higher-dimensional Euclidean space that are usually represented as hypersurfaces. The hypersurfaces representing those structures are depicted in red and denoted by Γ. Related bulk domains are depicted in shades of green and denoted by Ω_1, Ω_2 and Ω. The observation window in the ambient space is visualized by a black frame.

Hypersurfaces

From a mathematical point of view, embedded structures often can be represented as *hypersurfaces*, i.e. geometric objects that have codimension 1 in the higher-dimensional Euclidean space which is considered. This applies to a large number of the applications mentioned above. In the applications from fluid dynamics, for example, a special focus lies on interfaces between different immiscible fluids, and in the applications from cell biology, biological membranes play a prominent role. See Figure 1.1 for illustrations.

Depending on the application and its geometric entities, different types of hypersurfaces may be required. In particular, given a representation of distinct fluids by a set of different bulk domains, a fluidic interface corresponds to that part of the boundaries of two neighboring bulk domains which separates both bulk domains from each other. This part is a proper subset of both boundaries in many situations. On the contrary, a biological membrane usually is represented by the entire boundary of a bulk domain since it covers biological entities like cells or organelles. Hypersurfaces of the first kind, i.e. those depicted in Figure 1.1a, are referred to as *open hypersurfaces*. The second kind of hypersurfaces is known as *closed hypersurfaces*, see Figure 1.1b. In any case, hypersurfaces are mathematical models for embedded spatial structures of codimension 1 that are generally curved.

Instead of using the notion of hypersurfaces, it is also common practice to simply speak of surfaces. Although the latter term is mathematically correct for two-dimensional structures only, this abuse of terms is very convenient wherever mathematically precise terminology is not crucial. It accounts for the fact that the ambient space in real-world applications is three-dimensional.

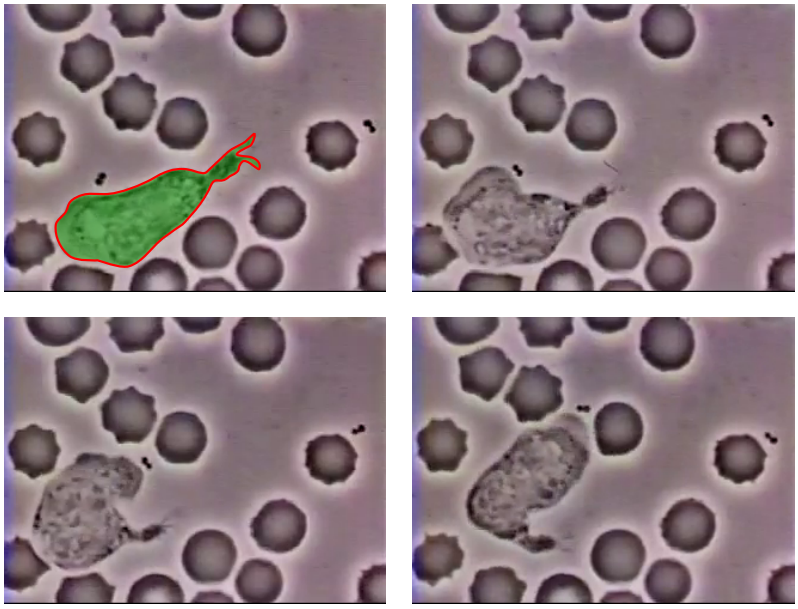
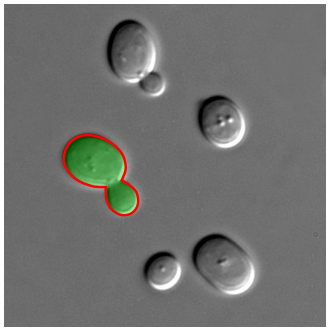


Figure 1.2.: Neutrophil (a type of white blood cell) chasing a bacterium of the species *Staphylococcus aureus* through a field of red blood cells on a blood film. Image sequence (top left to bottom right in reading order) extracted from a video clip[1] that is an excerpt from a 16 mm movie made by David E. Rogers at Vanderbilt University in the 1950s. Video clip courtesy of Thomas P. Stossel (Harvard Medical School; Brigham and Women's Hospital, Boston), and Philip G. Allen (Boston University) who digitized it from the analog footage and uploaded it on the internet. Color added in this thesis.

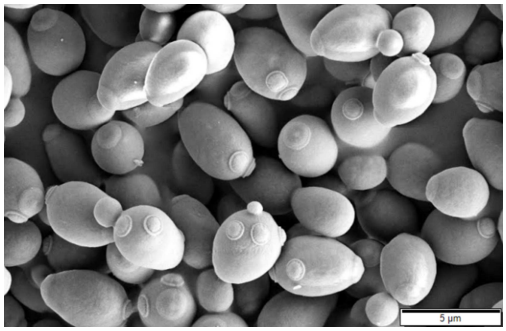
[1] `http://biochemweb.net/neutrophil.shtml`

Evolving geometries

In real-world applications, nature not only dictates the dimensionality of the spatial geometry that is mapped by bulk domains and hypersurfaces. It also often calls for models which take into account the evolution of this spatial geometry by employing bulk domains and hypersurfaces which evolve in time. For instance, the applications mentioned above include a biological process known as chemotaxis. Chemotaxis is a cellular crawling mechanism that is characterized by motion guided toward chemical cues which cells sense in their environment. Figure 1.2 shows an example of a particular cell type. Another well-known biological mechanism where geometrical evolution has a non-negligible impact is the division of cells. It is observable, for example, in the case of yeast cells, see Figure 1.3.



(a) Yeast cells, differential interference contrast microscopy image (color added).



(b) A fresh culture of yeast cells, scanning electron microscopy image.

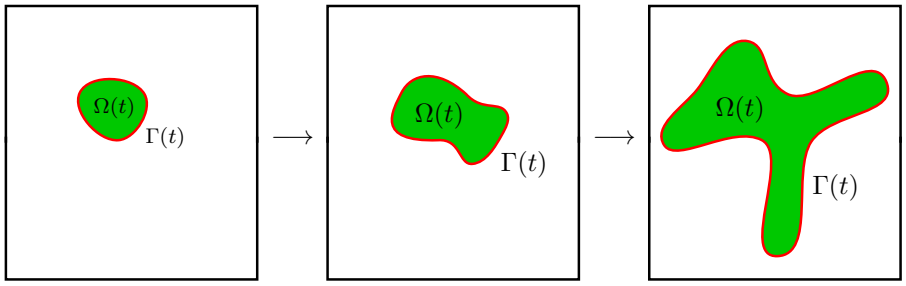
Figure 1.3.: Microscopy images of the yeast species *Saccharomyces cerevisiae*. In image (a), a single cell and its membrane have been marked green and red, respectively. The unmodified original image[2] has been released into the public domain. Image (b) was taken by Mogana Das Murtey (University of Science, Malaysia) and Patchamuthu Ramasamy (Quest International University Perak)[3], and is distributed under the CC-BY-SA-3.0 license[4].

[2] `https://commons.wikimedia.org/wiki/File:S_cerevisiae_under_DIC_microscopy.jpg`
[3] `https://commons.wikimedia.org/wiki/File:Saccharomyces_cerevisiae_SEM.jpg`
[4] `https://creativecommons.org/licenses/by-sa/3.0/legalcode`

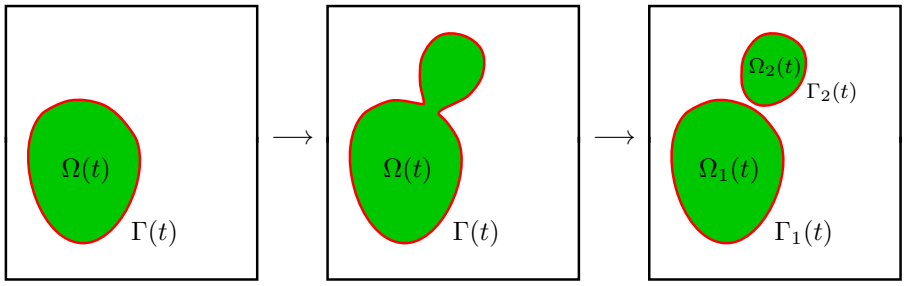
Both cellular mechanisms exhibit geometric properties that can be seen as role models for typical features of evolving geometries. In particular, chemotactic movement can be accompanied by strong, anisotropic deformations of the cell body. Moreover, cell division naturally leads to geometries with topological changes. These two typical features of evolving geometries are illustrated in Figure 1.4. A mathematical model which employs a static geometry representation in these cases can be expected to be realistic only for time scales that are very small, compared to time scales for which changes in geometry are observable. Such a model could, for example, focus on effects which trigger geometrical evolution that occurs on a larger time scale which is not captured by the model.

1.1. Bulk PDEs and surface PDEs

Mathematical models that are employed in applications like those mentioned above typically comprise partial differential equations (PDEs) formulated on bulk domains and hypersurfaces. A PDE is a mathematical equation which describes the relation of an unknown function of multiple continuous variables and its partial derivatives. Since such relations are extremely common, PDEs



(a) Evolving geometries with strong, anisotropic deformations, as observable in cell motility.



(b) Evolving geometries with topological changes, as observable in cell division.

Figure 1.4.: Evolving geometries and two typical features that can be observed in applications. Evolving bulk domains are depicted in green and denoted by $\Omega(t)$, $\Omega_1(t)$ and $\Omega_2(t)$. Evolving hypersurfaces are depicted in red and denoted by $\Gamma(t)$, $\Gamma_1(t)$ and $\Gamma_2(t)$.

are a standard tool in mathematical modeling of systems whose unknowns are distributions observed in some continuum of multiple dimensions, such as, in the majority of cases, a multidimensional spatial continuum or a space–time continuum. They can be used to model various processes from physics, biology and chemical science, and also processes from economics.

To distinguish between a PDE formulated on some bulk domain and a PDE that is formulated on a hypersurface, it is convenient to use the terms *bulk PDE* and *surface PDE*. Despite this classification, generally every PDE can be formulated for both types of geometry, which results in equations with a very similar structure.

1.1.1. Continuity equations on static geometries

One prominent example is Poisson's equation. Its formulations for a static bulk domain Ω and a static hypersurface Γ take the form

$$-\Delta u_b = g_b \quad \text{in} \quad \Omega, \tag{1.1a}$$

$$-\Delta_\Gamma u_s = g_s \quad \text{on} \quad \Gamma, \tag{1.1b}$$

with functions u_b and u_s that are the unknown solution, and functions g_b and g_s which represent given data. Both formulations are identical to each other except for the differential operator which they employ. The operator Δ in equation (1.1a) is the Laplacian which is given by the divergence of the gradient of a twice-differentiable function defined in Euclidean space. Applied to a suitable function u_b, it is frequently denoted as $\Delta u_b = \nabla \cdot \nabla u_b$, where $\nabla \cdot$ and ∇ denote the divergence operator and the transposed gradient operator in Euclidean space, respectively. In Cartesian space, i.e. Euclidean space with Cartesian coordinates, which is the standard representation of Euclidean space that will be considered from now on, these two operators take derivatives along the coordinate axes. The operator Δ_Γ in equation (1.1b) is known as the Laplace–Beltrami operator. Its concept is similar to that of the Laplacian Δ but it additionally takes into account the fact that the hypersurface Γ is a curved space. Analogous to Δ, its action on a function u_s on Γ can be defined as $\Delta_\Gamma u_s = \nabla_\Gamma \cdot \nabla_\Gamma u_s$, where $\nabla_\Gamma \cdot$ and ∇_Γ are a divergence operator and a transposed gradient operator for fields on Γ. Being defined using a suitable notion of partial derivatives in curved spaces, these two operators both respect the fact that Γ is curved. Instead of taking derivatives along the coordinate axes of the ambient Cartesian space, derivatives are taken with respect to local coordinate systems. Details will be given later on in this thesis.

Generalizing Poisson's equation by allowing for non-constant diffusion yields steady-state diffusion equations

$$-\nabla \cdot (\mathcal{D}_b \nabla u_b) = g_b \quad \text{in} \quad \Omega, \tag{1.2a}$$
$$-\nabla_\Gamma \cdot (\mathcal{D}_s \nabla_\Gamma u_s) = g_s \quad \text{on} \quad \Gamma, \tag{1.2b}$$

where $\mathcal{D}_b$ and $\mathcal{D}_s$ are parameters known as bulk and surface diffusivity tensors which determine the diffusive flux in Ω and on Γ, respectively. Being a surface diffusivity tensor, $\mathcal{D}_s$ maps the tangent space of Γ into itself at every point. With this requirement, equation (1.2b) is a model for a diffusion process which solely happens inside of Γ, even though Γ is a curved space in general.

Analogous diffusion equations which model dynamical systems in a static bulk domain Ω and on a static hypersurface Γ during an observation period $[0, T]$ are given by

$$\partial_t u_b - \nabla \cdot (\mathcal{D}_b \nabla u_b) = g_b \quad \text{in} \quad \Omega \times (0, T], \tag{1.3a}$$
$$\partial_t u_s - \nabla_\Gamma \cdot (\mathcal{D}_s \nabla_\Gamma u_s) = g_s \quad \text{on} \quad \Gamma \times (0, T]. \tag{1.3b}$$

In comparison with their steady-state analogues (1.2) considered above, both equations are still formulated on the same kind of static geometries, but each solution is time-dependent now. The additional terms on the left-hand side are partial derivatives with respect to time, which describe the temporal rate of change of the solution function and therefore vanish once the latter reaches a steady state, if any. In contrast to spatial differential operators, the same notion of time derivative is used for both formulations since their time domains

are subsets of the same flat space.

Generalizing the functions g_b and g_s on the right-hand side by permitting terms which depend on the unknown solution results in reaction–diffusion equations

$$\begin{aligned} \partial_t u_b - \nabla \cdot (\mathcal{D}_b \nabla u_b) &= f_b(u_b) \quad \text{in} \quad \Omega \times (0,T], \\ \partial_t u_s - \nabla_\Gamma \cdot (\mathcal{D}_s \nabla_\Gamma u_s) &= f_s(u_s) \quad \text{on} \quad \Gamma \times (0,T]. \end{aligned}$$

Moreover, incorporating transport in terms of an advective flux that is intrinsic to Ω and Γ yields reaction–advection–diffusion equations

$$\partial_t u_b + \nabla \cdot (-\mathcal{D}_b \nabla u_b + u_b \mathbf{w_b}) = f_b(u_b) \quad \text{in} \quad \Omega \times (0,T], \tag{1.4a}$$

$$\partial_t u_s + \nabla_\Gamma \cdot (-\mathcal{D}_s \nabla_\Gamma u_s + u_s \mathbf{w_s}) = f_s(u_s) \quad \text{on} \quad \Gamma \times (0,T]. \tag{1.4b}$$

Here, the advective flux is driven by velocity fields $\mathbf{w_b}$ and $\mathbf{w_s}$, respectively, where $\mathbf{w_s}$ points tangential to the hypersurface Γ in each point, such that the requirements of fluxes which are intrinsic to curved spaces are met.

All equations which we have considered so far for static geometries are members of classes of PDEs known as *bulk/surface continuity equations*. Continuity equations represent models for quantities that undergo various *conservative* processes. Since most of the processes observed in nature are conservative, continuity equations are a powerful tool in mathematical modeling. Details on their systematic derivation and their relationship with the conservation of some quantity will be given later on in this thesis. Next, we will focus on their formulation for evolving geometries.

1.1.2. Continuity equations on evolving geometries

For evolving geometries, the plainest possible continuity equations are advection equations accounting for conservative material transport which is solely driven by the geometrical evolution. In particular, for an evolving bulk domain $\Omega(t)$ and an evolving hypersurface $\Gamma(t)$, they are given by

$$\partial_t u_b + \mathbf{v_b} \cdot \nabla u_b + u_b(\nabla \cdot \mathbf{v_b}) = 0 \quad \text{in} \quad \bigcup\nolimits_{t\in(0,T]} \Omega(t) \times \{t\}, \tag{1.5a}$$

$$\partial^\bullet u_s + u_s(\nabla_\Gamma \cdot \mathbf{v_s}) = 0 \quad \text{on} \quad \bigcup\nolimits_{t\in(0,T]} \Gamma(t) \times \{t\}, \tag{1.5b}$$

where $\mathbf{v_b}$ and $\mathbf{v_s}$ are fields which describe the velocity of material points in $\Omega(t)$ and the velocity of material points on $\Gamma(t)$ from an Eulerian point of view, respectively. More precisely, along the trajectory of each individual point that moves with $\Omega(t)$ or $\Gamma(t)$, the relevant field specifies the temporal evolution of the velocity of this point. This information is known as the *material velocity* of $\Omega(t)$ and $\Gamma(t)$, respectively.

In comparison with time-dependent continuity equations for static geometries, such as (1.3), equations (1.5) use a different notion of time derivative

since the time domain and the space domain are no longer orthogonal to each other. More specifically, both formulations employ a Lagrangian derivative which is known as the *material derivative*. Applied to the solution function, it describes the temporal rate of change of the solution function while following the trajectory of material points in $\Omega(t)$ and on $\Gamma(t)$, respectively. Each formulation comes with its own version of this derivative. In equation (1.5a), the material derivative of u_b (with respect to the material velocity $\mathbf{v_b}$) is represented by the term $\partial_t u_b + \mathbf{v_b} \cdot \nabla u_b$. The material derivative of u_s (with respect to the material velocity $\mathbf{v_s}$) in equation (1.5b) is denoted by $\partial^\bullet u_s$. It does not have a similar representation since the required partial derivatives in Cartesian spaces that contain the space–time representation of an evolving hypersurface are not defined for fields which live on this hypersurface, unless an extension of those fields to a space–time neighborhood of the hypersurface is given. More details on $\partial^\bullet u_s$ will be given later on in this thesis. The right-most terms on the left-hand side of equations (1.5) render the material transport conservative.

To shorten notation in PDEs for evolving geometries, it is customary to write $\Omega(t)$ and $\Gamma(t)$ instead of the full space–time geometries $\bigcup_{t\in(0,T]} \Omega(t) \times \{t\}$ and $\bigcup_{t\in(0,T]} \Gamma(t) \times \{t\}$. By abusing notation this way and by combining terms in equation (1.5a), equations (1.5) can be equivalently formulated as

$$\partial_t u_b + \nabla \cdot (u_b \mathbf{v_b}) = 0 \quad \text{in} \quad \Omega(t), \tag{1.5$\tilde{\text{a}}$}$$

$$\partial^\bullet u_s + u_s (\nabla_\Gamma \cdot \mathbf{v_s}) = 0 \quad \text{on} \quad \Gamma(t). \tag{1.5$\tilde{\text{b}}$}$$

Apart from conservation of a bulk/surface quantity on an evolving geometry, equations (1.5) model no other physical effects. Taking into account all physical effects which we have considered above for static geometries results in reaction–advection–diffusion equations for evolving geometries:

$$\partial_t u_b + \nabla \cdot (u_b \mathbf{v_b}) + \nabla \cdot (-\mathcal{D}_b \nabla u_b + u_b \mathbf{w_b}) = f_b(u_b) \quad \text{in} \quad \Omega(t), \tag{1.6a}$$

$$\partial^\bullet u_s + u_s (\nabla_\Gamma \cdot \mathbf{v_s}) + \nabla_\Gamma \cdot (-\mathcal{D}_s \nabla_\Gamma u_s + u_s \mathbf{w_s}) = f_s(u_s) \quad \text{on} \quad \Gamma(t). \tag{1.6b}$$

These equations are similar to the static geometry reaction–advection–diffusion equations (1.4), but the time derivatives have been replaced by terms that are known from equations (1.5). In fact, equations (1.6) can be considered as a generalization of the static geometry reaction–advection–diffusion equations. This is revealed by taking $\mathbf{v_b} \equiv \underline{\mathbf{0}}$ and $\mathbf{v_s} \equiv \underline{\mathbf{0}}$ in equations (1.6). This special case corresponds to having a static bulk domain $\Omega(t) \equiv \Omega$ and a static hypersurface $\Gamma(t) \equiv \Gamma$. Furthermore, all terms vanish which contain the material velocity, and the material derivative $\partial^\bullet u_s$ degenerates to the partial derivative $\partial_t u_s$.

1.1.3. Non-conservative equations

In addition to continuity equations, which model conservative processes, PDEs can also be important that represent models for processes which do not con-

serve their associated quantities. A notable example is the non-conservative advection equation. For a static bulk domain Ω, it is given by the bulk PDE

$$\partial_t u_b + \mathbf{w_b} \cdot \nabla u_b = 0 \quad \text{in} \quad \Omega \times (0, T], \tag{1.7}$$

where $\mathbf{w_b}$ is a time-dependent vector field living in Ω, as with equation (1.4a). Interpreting Ω as an observation window, and $\mathbf{w_b}$ as a field which describes the velocity of moving particles as long as they reside in Ω, the equation can be seen as a model for a quantity whose concentration u_b is constant along the trajectory of each particle. Of course, this kind of equation can be also formulated as a surface PDE by using the appropriate differential operator and a suitable velocity field. However, in this thesis, we will restrict our attention to surface PDEs that are continuity equations. Except for equation (1.7), non-conservative bulk PDEs will neither be of particular importance.

1.2. A class of bulk–surface models

Mathematical models not necessarily consist of a solitary PDE which is posed on one single bulk domain or hypersurface. In most of the applications which have been mentioned at the beginning of this introduction, they usually rather comprise multiple PDEs that are coupled with each other in some way.

Especially in the last decade, models have become popular which include PDEs on a bulk domain and the hypersurface that is formed by the boundary of this bulk domain. Examples from the set of applications which has been mentioned at the beginning of this introduction include models for intracellular dynamics (Novak et al., 2007; Goryachev and Pokhilko, 2008; Medvedev and Stuchebrukhov, 2011; Rätz and Röger, 2012; Medvedev and Stuchebrukhov, 2013; Rätz and Röger, 2014), (Giese, Eigel, Westerheide, Engwer and Klipp, 2015a), (MacDonald et al., 2016; Emken, 2016), models for pattern formation on surfaces (Madzvamuse et al., 2015; Madzvamuse and Chung, 2016), and models for two-phase flows with surfactants respectively models dealing with surfactants in general (Booty and Siegel, 2010; Teigen et al., 2011; Ganesan and Tobiska, 2012; Ganesan et al., 2012; Hahn et al., 2013, 2014).

In this thesis, such systems of equations are referred to as models comprising *bulk–surface PDEs*, or in short, *bulk–surface models*. A class of bulk–surface models which will serve as a particular example will be introduced next. It is biologically motivated and encompasses many of the models for intracellular dynamics mentioned above. Models in the latter set, that do not fit in with the class of bulk–surface models which we are about to introduce, can be described by adding similar bulk PDEs or similar surface PDEs. To deal with the most general setting, we will consider a formulation for evolving geometries.

Let $\Omega(t)$ be an evolving bulk domain of some dimension $d \in \mathbb{N}$, bounded by an evolving hypersurface $\Gamma(t)$ and observed during a specified time period $[0, T]$. Assume that there exists a field of outward-pointing unit normal vectors to $\Gamma(t)$ denoted by $\boldsymbol{\nu}(\cdot, t)\colon \Gamma(t) \to \mathbb{R}^d$, where $\mathbb{R}^d$ denotes the real coordinate

space which models the d-dimensional Cartesian space containing $\Omega(t)$ and $\Gamma(t)$. Furthermore, let $\mathbf{v}(\cdot,t)\colon \Omega(t)\cup\Gamma(t)\to\mathbb{R}^d$ be a field which describes the material velocity of $\Omega(t)\cup\Gamma(t)$.

On the evolving geometry, we consider two conserved scalar quantities, namely a bulk quantity in $\Omega(t)$ and a surface-bound quantity on $\Gamma(t)$ that are represented by concentrations $u_b(\cdot,t)\colon \Omega(t)\to\mathbb{R}$ and $u_s(\cdot,t)\colon \Gamma(t)\to\mathbb{R}$, respectively, whose evolution is driven by bulk/surface reactions, bulk/surface diffusion, and reactive interactions of both quantities on $\Gamma(t)$. Given some initial values $u_b(\cdot,0)$ and $u_s(\cdot,0)$, the considered class of model problems reads

$$\partial_t u_b + \nabla\cdot(u_b\mathbf{v}) - \nabla\cdot(\mathcal{D}_b\nabla u_b) = f_b(u_b) \qquad \text{in } \Omega(t), \quad (1.8a)$$

$$-\mathcal{D}_b\nabla u_b\cdot\boldsymbol{\nu} = -f_{b,s}(u_b,u_s) \qquad \text{on } \Gamma(t), \quad (1.8b)$$

$$\partial^\bullet u_s + u_s(\nabla_\Gamma\cdot\mathbf{v}) - \nabla_\Gamma\cdot(\mathcal{D}_s\nabla_\Gamma u_s) = f_{s,b}(u_b,u_s) + f_s(u_s) \qquad \text{on } \Gamma(t). \quad (1.8c)$$

First, we note its relationship with reaction–advection–diffusion equations for evolving geometries, which have been presented in equations (1.6). Equation (1.8a) is a bulk reaction–diffusion equation for the evolving domain $\Omega(t)$. It is equal to equation (1.6a) in the case $\mathbf{w_b}\equiv\underline{\mathbf{0}}$, i.e. there is no advective flux that is intrinsic to $\Omega(t)$. Analogously, equation (1.8c) is a surface reaction–diffusion equation posed on the evolving hypersurface $\Gamma(t)$, which is similar to equation (1.6b) in the case $\mathbf{w_s}\equiv\underline{\mathbf{0}}$. As with equations (1.6), $\mathcal{D}_b$ and $\mathcal{D}_s$ are the bulk diffusivity tensor and the surface diffusivity tensor, respectively, with $\mathcal{D}_s\underline{\boldsymbol{\nu}^\perp}\cdot\boldsymbol{\nu}=0$ on $\Gamma(t)$ for every tangential vector $\underline{\boldsymbol{\nu}^\perp}$. The latter property is the requirement which has been discussed for equation (1.2b). Furthermore, reactions are described by two functions $f_b(u_b)(\cdot,t)\colon \Omega(t)\to\mathbb{R}$ and $f_s(u_s)(\cdot,t)\colon \Gamma(t)\to\mathbb{R}$, that are potentially nonlinear terms in u_b and u_s.

Apart from that, the above class of bulk–surface models takes into account two functions $f_{b,s}(u_b,u_s)(\cdot,t)\colon \Gamma(t)\to\mathbb{R}$ and $f_{s,b}(u_b,u_s)(\cdot,t)\colon \Gamma(t)\to\mathbb{R}$ which have the same property as $f_b(u_b)$ and $f_s(u_s)$ regarding u_b and u_s. They could, for example, describe transitions between u_b and u_s that are caused by additional reactions at the boundary. In general, they couple the processes in $\Omega(t)$ and $\Gamma(t)$. The coupling in equation (1.8a) is due to its Robin-like boundary condition (1.8b), whereas $f_{s,b}(u_b,u_s)$ appears as a standard surface reaction term in equation (1.8c). Those terms in (1.8a) and (1.8c) which account for material transport driven by the evolution of the bulk domain and its surface can provide an additional coupling between both equations. This applies if the material velocity $\mathbf{v}$ depends on the solution variables u_b and u_s.

Furthermore, since the class of bulk–surface models given by equations (1.8) comprises continuity equations and a suitable boundary condition, the solution (u_b,u_s) of each specific model satisfies certain *conservation properties*. As we will see later on in this thesis, those conservation properties entail that the total amount $m(t) := \int_{\Omega(t)} u_b \,\mathrm{d}x + \int_{\Gamma(t)} u_s \,\mathrm{d}\sigma$ of the system's quantities is an invariant with respect to time if the model parameters are chosen accordingly. This holds true, e.g., for models with $f_b\equiv 0$, $f_s\equiv 0$ and $f_{b,s}=-f_{s,b}$.

The considered class of evolving geometry model problems encompasses a similar class of bulk–surface models for static geometries. The latter corresponds to the family of evolving geometry model problems with $\mathbf{v} \equiv \underline{\mathbf{0}}$. In this special case, the evolving geometry degenerates to a static bulk domain $\Omega(t) \equiv \Omega$ and a static hypersurface $\Gamma(t) \equiv \Gamma$, and terms related to conservative material transport driven by $\mathbf{v}$ either vanish or simplify. Details have been discussed for equations (1.6).

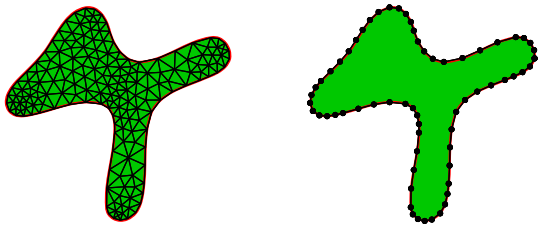
1.3. Numerical methods for bulk PDEs and surface PDEs

Once a mathematical model has been established, simulations are performed by solving its model equations. Exact solutions to PDE-based models can be obtained using analytical methods to solve PDEs, such as the concept of fundamental solutions. While such analytical tools have been successively applied to basic linear bulk PDEs like Poisson's equation (1.1a) or diffusion equation (1.3a) with a constant, scalar diffusivity tensor $\mathcal{D}_b$ and constant g_b, exact analytical solutions are usually unavailable for more complicated PDEs and systems of PDEs which arise in many interesting problems in science and engineering. Therefore, methods have been developed that determine numerical approximations to the solution. These methods are known as numerical methods for PDEs. Besides being required in many cases to obtain at least an approximate solution, they are perfectly suitable for performing simulations using computers.

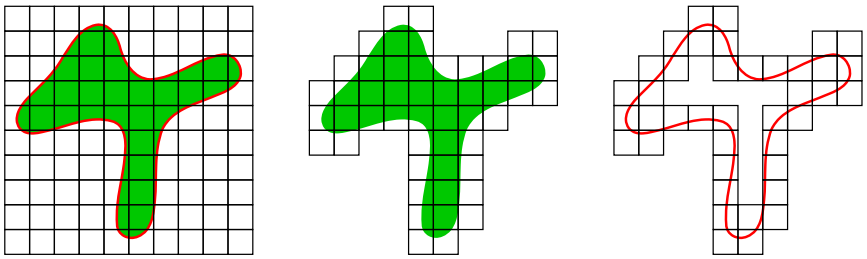
Numerical approaches to solving PDEs can be classified into methods which build upon a computational mesh, i.e. a subdivision of some geometric object into a set of smaller geometric objects, and meshfree methods. In this thesis, the focus lies on the first-mentioned class of numerical methods. Members of this class use meshes to construct finite-dimensional function spaces that are capable of approximating the solution of a PDE with a certain degree of accuracy. Such discrete approximation spaces typically consist of functions that are piecewise polynomials with respect to the mesh elements. The approximation error then depends on the size of the mesh elements and on the degree of the polynomial function space on each element. Depending on the specific method, either the element size or both can vary locally.

Among numerical approaches which employ a computational mesh, two different families of methods can be identified. We describe their general features now, and discuss specific members of each family later on in Section 1.3.1 and in Section 1.3.2, respectively.

On the one hand, there are *classical mesh-based methods*. Here, the mesh that is used is a decomposition of some geometric object which can be seen as an approximation of the geometrical setup associated with the PDE, i.e. the bulk domain or hypersurface of interest. For a bulk domain, such a mesh has the property that its outer vertices lie on the domain boundary. The mesh is said to be an approximation of the bulk domain. If also its outer facets lie on



(a) Classical meshes.



(b) Geometrically unfitted meshes.

Figure 1.5.: Types of computational meshes that are employed in mesh-based numerical methods for PDEs. An unstructured triangular mesh which approximates the bulk domain Ω from Figure 1.1b is shown on the left-hand side of (a), depicted in black. Its outer entities make up a surface mesh for $\Gamma = \partial\Omega$, which is shown on the right. Geometrically unfitted meshes for the same bulk domain Ω and hypersurface Γ are shown in (b). In particular, the left-most image of (b) depicts a structured Cartesian mesh that is suitable for Ω and for Γ. The other two images depict subsets of this mesh that are appropriate geometrically unfitted meshes as well, either for Ω or for Γ.

the boundary, we say that the mesh resolves the domain boundary exactly. In the latter case, the bulk domain is exactly represented by the mesh. Similarly, given some hypersurface, all vertices of a valid mesh need to be contained in the set of points on the hypersurface. Meshes of this kind are known by the name surface mesh and are said to approximate the given hypersurface. See Figure 1.5a for illustrations and Section 1.3.1 for specific methods.

On the other hand, there is the family of *geometrically unfitted mesh-based methods*, whose members are often just called *unfitted methods*. Its members use a mesh which can be nearly unrelated to the geometrical setup that is associated with the PDE. The only requirement regarding the relationship between this mesh and the geometrical setup is that bulk domains or hypersurfaces of interest need to be covered by the mesh. This property is necessary for

constructing a suitable discrete approximation space. In this context, a space is considered as being suitable if it comprises functions that can be restricted to the original geometry, with restrictions that make up a set of functions capable of approximating the solution. Since PDEs determine features of their solution and its approximations solely on the geometry on which they are posed, unfitted methods usually require some extra effort to gain control over that part of a discrete function which exceeds the original geometry. This is particularly crucial for surface PDEs, where the geometry of interest has codimension 1, compared to the geometrical dimension of the mesh and the associated approximation space. Examples for geometrically unfitted meshes are depicted in Figure 1.5b. For an overview of specific geometrically unfitted mesh-based methods, we refer to Section 1.3.2.

1.3.1. Classical mesh-based methods

Methods for bulk PDEs

Classical mesh-based numerical methods for bulk PDEs are a well-established tool in numerical mathematics and scientific computing. They can be classified into four categories.

Finite difference methods
First, there is the broadly-known category of *finite difference (FD) methods* (see e.g. LeVeque, 2007), which replace the partial derivatives in the PDE in a straightforward way by suitable discrete analogues on Cartesian meshes or, more generally, on rectilinear meshes. These analogues, known as finite differences, are difference quotients that approximate the corresponding partial derivatives, provided that the unknown solution has sufficient regularity. Since FD methods are quite restrictive regarding the geometry of the computational mesh, they are not considered as being the natural choice for PDE on bulk domains of complex shape.

Finite element methods
The second category comprises *finite element methods (FEMs)* for bulk PDEs, whose ideas are discussed in detail in many introductory courses on numerical methods for PDEs and in many textbooks, see e.g. Zienkiewicz et al. (2013). FEMs are more flexible with respect to the mesh which is used to approximate the bulk domain of interest. The PDEs are formulated in a weak sense using integral calculus and distribution theory. Subsequently, the corresponding weak solution to this reformulation is approximated in discrete function spaces that are constructed using the mesh. Those function spaces classically comprise continuous, piecewise polynomial functions living on the mesh elements. Central to FEMs is the concept of using basis functions whose support spreads over a small amount of adjacent mesh elements only.

More advanced FEMs extend the idea of FEMs for PDEs on static bulk domains to PDEs on evolving bulk domains $\Omega(t)$ by employing moving meshes,

particularly meshes whose vertices move with the material velocity $\mathbf{v_b}$ that has been introduced in Section 1.1.2. To ensure a better quality of the moving mesh, this idea is further extended in a family of methods known as *arbitrary Lagrangian–Eulerian finite element methods (ALE FEMs)*, which offer the possibility of using a non-physical, arbitrary velocity for the mesh vertices. See Donea et al. (2004), for instance.

Discontinuous Galerkin methods

The third catergory, which can alternatively be seen as a subcategory of FEMs, is the class of *discontinuous Galerkin (DG) methods*. As with FEMs, DG methods are based on the weak formulation of the PDE, and discrete approximation spaces are constructed using the mesh which approximates the bulk domain of interest. Those spaces, however, allow for discontinuities across the boundaries of mesh elements, building upon piecewise continuous basis functions that are supported on single mesh elements only. Where desirable, continuity across element boundaries is enforced in a weak sense using penalty terms that are added to the discrete analogue of the weak formulation.

The additional freedom which stems from considering discontinuities of the above kind is beneficial regarding conservation properties and it allows for naturally handling advection terms like those which arise from evolving geometries. Moreover, it allows for constructing higher order approximation spaces in a straightforward way and facilitates parallelization of DG schemes.

Introductions to DG methods for bulk PDEs can be found, e.g., in Rivière (2008) and in Di Pietro and Ern (2012). Further details will also be given in this thesis.

Finite volume methods

The fourth category is the class of *finite volume (FV) methods*, see LeVeque (2002), for instance. Those methods have been developed for numerically approximating solutions to continuity equations. Being directly based on the idea of reproducing physical fluxes over the boundary of control volumes in some discrete sense, they recover discrete analogues to the equation's underlying conservation properties in a natural way. While this concept generally allows for using higher order approximation spaces by employing a primal mesh for constructing the approximation space and an implicitly defined dual mesh that specifies the control volumes, basic FV methods typically are based on approximation spaces which comprise piecewise constant functions, i.e. functions that are piecewise polynomials of degree 0. These methods can be interpreted as DG methods of order 0.

Conversely, many DG methods can be seen as higher order generalizations of basic FV methods of order 0. Those DG methods approximate physical fluxes over the boundary of control volumes numerically, as with FV methods, and they can be derived from what is known as the flux formulation. Details on this aspect of DG methods will be given later on in this thesis.

Methods for surface PDEs

Surface FEMs

The development of classical mesh-based numerical methods for surface PDEs was initiated by the work of Dziuk (1988). In this seminal paper, the fundamental idea of FEMs for PDEs on static bulk domains is transferred to PDEs on static hypersurfaces, taking Poisson's equation (1.1b) as an example. After formulating the equation in some weak sense and generating a surface mesh which approximates the given hypersurface, the corresponding weak solution is approximated in discrete function spaces that are constructed using the surface mesh. Those function spaces comprise continuous, piecewise linear[5] functions living on the mesh elements.

Nowadays, methods of this kind are commonly known as *surface finite element methods (SFEMs)* and the original approach was developed further. For instance, SFEMs for parabolic equations like equation (1.3b) were investigated in Dziuk and Elliott (2007b), and SFEMs for higher order approximations in space have been introduced in Demlow (2009) in the context of elliptic problems.

As with FEMs for bulk PDEs, the concept of SFEMs has been generalized using moving surface meshes to deal with PDEs on evolving hypersurfaces $\Gamma(t)$. Such *evolving surface finite element methods (ESFEMs)* are introduced and applied, e.g., in Dziuk and Elliott (2007a) and in Barreira et al. (2011). By employing meshes whose vertices move with the material velocity $\mathbf{v_s}$ that has been introduced in Section 1.1.2, special features of continuity equations can be exploited which simplify the approach. In corresponding *arbitrary Lagrangian–Eulerian evolving surface finite element methods (ALE ESFEMs)* a non-physical, arbitrary velocity for the mesh vertices can be used to obtain moving surface meshes that maintain a better quality throughout the entire simulation (see e.g. Elliott and Styles, 2012; Elliott and Venkataraman, 2015).

A fairly recent overview of classical mesh-based FEMs for surface PDEs can be found in Dziuk and Elliott (2013).

Surface DG methods and surface FV methods

In addition to FEMs, also the ideas of DG methods and FV methods for bulk PDEs have been transferred to surface PDEs. *Surface discontinuous Galerkin (SDG) methods* for PDEs on static hypersurfaces are introduced and investigated in Dedner et al. (2013); Madhavan (2014); Antonietti et al. (2015); Dedner and Madhavan (2015, 2016).

FV methods for PDEs on static hypersurfaces are considered in Tang et al. (2005); Ju and Du (2009). Moreover, Lenz et al. (2011); Nemadjieu (2012); Giesselmann and Müller (2014) consider FV methods for PDEs on evolving hypersurfaces.

[5] Note that the notion of linear functions shall include affine functions here and in the following when talking about polynomial approximation spaces.

Methods for coupled bulk–surface PDEs

By combining the ideas of methods for bulk PDEs and of methods for surface PDEs, approaches have been obtained that are suitable for dealing with coupled bulk–surface PDEs. A classical mesh-based method for an elliptic bulk–surface model on static geometries, which combines the ideas of classical FEMs and SFEMs is investigated in Elliott and Ranner (2013). Similar approaches are utilized in Giese, Eigel, Westerheide, Engwer and Klipp (2015b) and in Emken (2016) for spatial discretization of parabolic bulk–surface models on static geometries, respectively. See Section 1.4.2 for more information. MacDonald et al. (2016) investigate a combined ALE FEM – ALE ESFEM for bulk–surface models on evolving geometries. A FV-based method for parabolic bulk–surface models on static geometries is introduced and investigated numerically in Novak et al. (2007).

1.3.2. Geometrically unfitted mesh-based methods

Methods for bulk PDEs

One central motivation for developing geometrically unfitted mesh-based approaches is to obtain methods for solving PDEs which are posed on complex-shaped bulk domains. By employing a mesh which can be nearly unrelated to the bulk domain that is associated with the PDE, it is possible to vary the size of the discrete approximation space independently of geometric properties and geometrical changes can be incorporated more easily.

Fictitious domain methods and immersed boundary methods

The history of geometrically unfitted mesh-based methods for bulk PDEs dates back to the development of *embedding domain methods* (see e.g. Buzbee et al., 1971). Those methods, that are more commonly known as *fictitious domain methods* (see e.g. Glowinski et al., 1994), are based on the idea of embedding the given bulk domain in some larger bulk domain of simple shape and extending the given bulk PDE to this larger bulk domain in such a way that a solution is obtained which matches the solution of the original PDE on the original bulk domain. More specifically, the original bulk domain boundaries are neglected and discretization is performed using concepts of classical FEMs on a mesh of the larger bulk domain. Meanwhile, the original boundary conditions are imposed as constraints on the extended PDE. This procedure yields a problem which is solved using the technique of Lagrange multipliers. The latter results in a saddle point problem with additional degrees of freedom that are associated with the constraints.

Immersed boundary methods (see e.g. Peskin, 1977, 2002) or *immersed interface methods* (see e.g. LeVeque and Li, 1994; Lee and LeVeque, 2003) are based on a similar idea. However, solving an expensive saddle point problem is avoided by implementing constraints as virtual forces that are based on regularized Dirac delta functions and incorporated into the right-hand side g_b of

the bulk PDE, cf. equation (1.1a).

The concepts of fictitious domain methods and immersed interface methods are frequently combined with an implicit description of the bulk domain boundaries by means of the *level set framework* (Osher and Sethian, 1988). See the works of Sussman et al. (1994); Calzada et al. (2011), for instance. In the level set framework, bulk domains and their boundaries are described using the level sets of functions which live on some larger, static bulk domain, such as a fictitious domain. Those functions are known by the name *level set functions*. Since they capture all information on the geometry and its motion in an implicit way by means of their level sets, the framework is particularly suitable for problems with evolving geometries. Details will be given later on in this thesis.

Diffuse domain methods

A different kind of implicit geometry description is used in methods that are known as phase field methods for bulk PDEs or *diffuse domain methods*, as introduced in Li et al. (2009). See also Reuter et al. (2012); Lervåg and Lowengrub (2015); Burger et al. (2017). While also being based on the idea of embedding bulk domains into a larger, static bulk domain of simple shape, diffuse domain methods replace domains with a sharp boundary by domains with a diffuse boundary, unlike methods which employ the level set framework. A bulk domain of interest is represented as a diffuse domain by means of a function which lives on the larger domain and is known as a *phase field function*. This function varies smoothly between the value 1 in the original bulk domain and 0 in its complement and has a rapid transition within a narrow diffuse boundary layer between them. Using this smeared-out version of the characteristic function of the original bulk domain, bulk PDEs are reformulated and extended to the larger embedding domain. Boundary conditions are approximated by incorporating additional source terms. The resulting extended equations can be dealt with numerically by applying classical mesh-based discretization methods for bulk PDEs, such as FD methods or classical FEMs. They converge to the original bulk PDEs and to the original boundary conditions as the width of the diffuse boundary layer tends to zero.

Diffuse domain methods are powerful approaches to solving PDEs on complex-shaped, potentially time-dependent bulk domains. They can be implemented using tools that are typically provided by standard PDE software frameworks and have been successfully applied in various applications (see e.g. Kockelkoren et al., 2003; Fenton et al., 2005; Levine and Rappel, 2005; Teigen et al., 2009; Aland et al., 2010; Teigen et al., 2011; Garcke et al., 2014; Marth and Voigt, 2014).

However, in this thesis, we are particularly interested in recovering discrete analogues to conservation properties that are embedded in models comprising continuity equations. Since diffuse domain approaches consider bulk domains with a diffuse boundary, conserved quantities, such as masses, spread over the corresponding diffuse boundary layer. Accordingly, conservation properties

that actually hold in the sharp domain which is associated with the model equations are replaced by conservation properties holding in the diffuse domain that is used for discretization. Even if a reconstruction of the sharp domain is extracted from the diffuse domain representation by identifying a selected level set of the phase field function with the bulk domain boundary, it is not clear how to obtain discrete conservation properties which hold in this reconstruction. Such discrete conservation properties are recovered only if the width of the diffuse boundary layer tends to zero and they only hold approximatively in this case.

The methods which we will discuss next, typically recover discrete conservation properties that hold in a sharp domain sense. Instead of modifying the problem at the PDE level and applying classical discretization methods on top, they build upon modifying the basis functions of standard approximation spaces according to the geometry.

Unfitted FEMs

Unfitted FEMs, as proposed in Barrett and Elliott (1987), consider standard FEM basis functions that are associated with some geometrically unfitted mesh and modify those basis functions according to the original bulk domain of interest. Employing that the unfitted mesh covers the latter bulk domain, say Ω, the basis functions are altered by defining them to take the value 0 outside Ω. As a result, evaluation of integrals in the weak formulation requires special quadrature rules which respect the modified support of each basis function near the boundary $\partial\Omega$. Essential boundary conditions on $\partial\Omega$ are imposed in a weak sense using a technique that is known as Nitsche's method (cf. Nitsche, 1971). Alternatively to thinking of unfitted FEM basis functions as standard basis functions with a modified support, they can be seen as standard basis functions that are solely integrated and evaluated over that subset of each mesh element which intersects the bulk domain Ω. The latter subset is often called a *cut cell*. Correspondingly, methods that are based on the concepts of unfitted FEMs are also known by the name *cut cell methods*.

An improved unfitted FEM is presented, e.g., in Hansbo and Hansbo (2002) and particularly recently, people have shown rising interest in methods of this kind. The concepts have been developed further under the name *cutFEMs* (Burman et al., 2015), most importantly by investigating suitable stabilization mechanisms known as *ghost penalties*, which target gaining control over the condition of the resulting systems of algebraic equations.

The UDG method

By combining the ideas of unfitted FEMs with DG discretizations, Bastian and Engwer (2009) developed the *unfitted discontinuous Galerkin (UDG) method*, see also Engwer (2009); Bastian et al. (2011); Heimann et al. (2013). Numerical analysis in the context of elliptic interface problems is presented in Massjung (2012). The UDG method offers the advantages that unfitted FEMs have when it comes down to solving PDEs on complex-shaped bulk domains. It

furthermore provides all benefits of DG approaches which have been mentioned in Section 1.3.1. Classical DG methods that are adapted to the specific needs of a given bulk PDE serve as *host DG formulations*. Accordingly, essential boundary conditions are imposed in a weak sense as part of these host DG formulations.

Due to the local nature of basis functions in DG approaches, the method allows for easily implementing stabilization mechanisms which built upon modification of basis functions, such as *rescaling* or *cell merging* techniques (cf. Johansson and Larson, 2013; Heimann et al., 2013). See also the *cell agglomeration* techniques in Müller et al. (2017); Kummer (2017); Kummer et al. (2018). Those stabilization mechanisms ensure that the resulting systems of algebraic equations are well-posed and well-conditioned by counteracting the effects of very small cut cells. This is specifically done by associating very small cut cells to a neighboring mesh element or to a cut cell that has a sufficiently large intersection with the bulk domain, and by rescaling basis functions according to cut cell bounding boxes, respectively. Stabilizing UDG discretizations similar to unfitted FEMs using ghost penalties has only just been investigated in Massing and Gürkan (2018).

Further details on the UDG method will be given later on in this thesis. We will investigate how the method can be applied to PDEs on complex-shaped hypersurfaces and to coupled bulk–surface PDEs on related geometries.

Extended FEMs

Geometrically unfitted mesh-based discretization methods for bulk PDEs can also be constructed using the concepts of *extended FEMs (XFEMs)*. Subsequent to starting as with classical FEMs, those methods incorporate discontinuities and other effects, e.g. at the boundary of some bulk domain of interest, by enriching the discrete approximation space by additional basis functions (cf. Moës et al., 1999; Dolbow, 1999; Belytschko et al., 2001). In the course of this, a mesh can be used that is independent of the geometry. On the downside, the enrichment process introduces additional degrees of freedom that need to be dealt with. In addition, the evaluation of integrals in the weak formulation requires special quadrature rules due to intra-element discontinuities that are introduced together with the additional basis functions.

Composite FEMs

Methods known as *composite FEMs* are based on a hierarchy of meshes comprising one fine mesh, which approximates the bulk domain of interest, and geometrically unfitted coarser meshes. Basis functions of discrete approximation spaces that are associated with the coarse meshes are constructed as linear combinations of standard FEM basis functions which live on the fine mesh.

Such methods are introduced and studied in Hackbusch and Sauter (1997b,a); Sauter (1997); Westerheide (2011). Employing their concepts solely in order to obtain coarse mesh solutions is theoretically possible. However, due to the necessity of having a fine mesh which approximates the bulk domain of in-

terest and the overhead that arises from constructing the latter, the concepts of composite FEMs are reasonable mainly in the context of geometric multigrid methods for PDE on complex-shaped bulk domains. Here, coarse mesh approximation spaces are not the main objective but need to be constructed as part of a fast iterative solver for fine mesh solutions. More recent variants of the approach which address the above shortcoming can be found in Rech et al. (2006); Liehr et al. (2009).

Methods for surface PDEs

As with classical mesh-based FEMs, a fairly recent overview of geometrically unfitted mesh-based methods for surface PDEs can be found in Dziuk and Elliott (2013). We extend this overview in the following.

Level set extension and FD based methods

The development of geometrically unfitted mesh-based methods for surface PDEs started with the work of Bertalmío et al. (2001); Bertalmío et al. (2003), the first authors who proposed to describe the geometry implicitly by means of a level set function and to subsequently employ this framework in a suitable manner to extend each surface PDE to some bulk domain which contains the hypersurface of interest. In particular, they showed how PDEs on static hypersurfaces that arise from the gradient descent of some energy can be extended by embedding techniques that are based on the level set framework. For each surface PDE, this process results in an associated bulk PDE which can be solved using the concepts of classical mesh-based discretization methods for bulk PDEs, provided that it is supplemented with an artificial boundary condition. Making use of the Cartesian structure of the differential operators which appear in those bulk PDEs, Bertalmío et al. discretize using FDs on Cartesian meshes in their work.

This level set extension and FD based approach for PDEs on static hypersurfaces is developed and analyzed further regarding different types of equations, improved embedding techniques and the effect of boundary conditions for the extended PDEs, e.g., in Cheng et al. (2002); Greer (2006); Greer et al. (2006). Moreover, extensions of the approach that deal with PDEs on evolving hypersurfaces are considered in Adalsteinsson and Sethian (2003); Xu and Zhao (2003), for example. In those extensions, another important advantage of using the level set framework for geometry description is that it is sufficient to employ a static mesh, even though dealing with PDEs on evolving hypersurfaces.

Level set extension and FEM based methods

In an article from the year 2005, which was published only later on, Burger (2009) started to investigate similar ideas in the context of FEMs. Subsequently, comparable level set extension and FEM based methods were developed in Dziuk and Elliott (2008), for PDEs on static hypersurfaces, and in Dziuk and Elliott (2010) for PDEs on evolving hypersurfaces. In this thesis,

the latter methods will be referred to as *Eulerian surface finite element method (Eulerian SFEM)* and *Eulerian evolving surface finite element method (Eulerian ESFEM)*, respectively. In view of evolving hypersurfaces, those names account for the fact that the implicit representation of the hypersurfaces by means of the level set framework follows an Eulerian point of view, whereas a Lagrangian point of view is taken by explicit representations using moving meshes. However, Eulerian methods of this kind are also commonly known by names such as *implicit SFEMs* and *implicit ESFEMs*.

While the above level set extension and FEM based methods mainly aim at employing structured Cartesian meshes, such as the one which is depicted in the left-most image in Figure 1.5b, Nemitz et al. (2009) investigate reducing the computational cost that results from this choice by employing meshes which correspond to some bulk domain representing a narrow band around the given hypersurface, i.e., meshes such as the one that is depicted in the right-most image in Figure 1.5b. Using *narrow bands with staircase-type boundaries*, that is, boundaries which are aligned with the coordinate axes of the ambient Cartesian space, the approach of Nemitz et al. (2009) required developing special boundary conditions for the extended PDEs. These special boundary conditions are designed to have minimal influence on the restriction of the discrete solution to the hypersurface of interest.

In the context of PDEs on static hypersurfaces, a more promising alternative is suggested by Deckelnick et al. (2010), who consider bulk domains that represent *narrow bands comprising a family of closed level sets* instead. To deal with the potential geometrical complexity that arises from dealing with those narrow bands, discretization is performed using the concepts of unfitted FEMs for bulk PDEs. This way of proceeding removes the need for special boundary conditions. Furthermore, by scaling the narrow band width in a suitable manner, an approach is obtained that allows for analyzing the discretization error not only in norms that are associated with the bulk approximation space, but also in norms that are intrinsic to the hypersurface. This approach and its error analysis is further improved in Deckelnick et al. (2014), where also an extension for PDEs on evolving hypersurfaces is proposed.

Level set extension and UDG based methods

In a proceedings article from the year 2012, which was published in 2014, Engwer and Westerheide (2014) made first efforts to combine an extension process based on the level set framework with UDG discretizations in order to deal with surface PDEs. These efforts have been continued in Engwer, Ranner and Westerheide (2016), aiming at difficulties that arise from material transport which is driven by the motion of evolving hypersurfaces, specifically in the context of continuity equations on such hypersurfaces.

This thesis will provide missing links and details. It particularly deals with questions related to the recovery of conservation properties that are embedded in many PDEs of practical relevance and it contributes to analyzing UDG-based approaches to surface PDEs.

Diffuse interface methods

A different kind of extension process is used in methods that are known as phase field methods for surface PDEs or *diffuse interface methods*, as considered in Rätz and Voigt (2006); Burger (2009); Elliott and Stinner (2009); Elliott et al. (2011). As with diffuse domain methods for bulk PDEs, diffuse interface methods employ implicit geometry description by means of a phase field function which lives on some larger, static bulk domain of simple shape. In the course of this, the sharp representation of a hypersurface of interest is replaced by a diffuse interface region containing the hypersurface. In this diffuse interface region, the phase field function rapidly transits from 0 on one side of the hypersurface to 1 on the other side in a smooth way. By employing the phase field function, say ϕ, indirectly by means of the function $\phi^2(1-\phi)^2$, which vanishes outside the diffuse interface, surface PDEs are reformulated and extended to the embedding bulk domain. The resulting extended equations can be dealt with numerically by applying classical mesh-based discretization methods for bulk PDEs, such as FD methods or classical FEMs. They converge to the original surface PDEs as the width of the diffuse interface region tends to zero.

Diffuse interface methods are powerful approaches to solving PDEs on complex-shaped, potentially time-dependent hypersurfaces. They can be implemented using tools that are typically provided by standard PDE software frameworks and have been successfully applied in various applications (see e.g. Rätz et al., 2006; Lowengrub et al., 2009; Torabi et al., 2009; Teigen et al., 2009, 2011; Garcke et al., 2014; Marth and Voigt, 2014).

However, due to the diffuse interface representation, similar considerations regarding conservation properties hold as with diffuse domain methods for bulk PDEs, i.e. discrete analogues to conservation properties that are embedded in models comprising continuity equations only hold in the diffuse interface region. On a sharp reconstruction of the original hypersurface in terms of some level set of the phase field function, they only hold approximatively.

The latter is typically also an issue of the level set extension based methods which we discussed above. However, in this thesis, we will show how the issue can be cured in the construction of such methods.

Closest point methods

Another interesting extension process is employed in *closest point methods*. Those methods compute an extension of the solution to a surface PDE using a function known as *closest point projection*. This function maps each point in a bulk neighborhood of the hypersurface onto the closest point which lies on the hypersurface. By replacing all evaluations of the solution and of data functions in the surface PDE by analogous evaluations which apply the closest point projection first, an associated bulk PDE is obtained that can be expressed via classical differential operators in the ambient Cartesian space. This PDE can be solved using the concepts of classical mesh-based discretization methods for bulk PDEs. Its solution is a normally constant extension of the solution

to the original surface PDE.

However, a sufficiently accurate representation of the closest point projection has to be computed first and evaluations using the closest point projection need to be performed afterwards. Both can be expensive due to the non-local nature of both processes in general: given a point in some mesh element, its associated closest point on the hypersurface may lie in a different mesh element.

Closest point methods that apply the closest point extension together with a FD discretizations are investigated in Ruuth and Merriman (2008); Macdonald and Ruuth (2009) for PDEs on static hypersurfaces, and in Petras and Ruuth (2016) for PDEs on evolving hypersurfaces.

Extension-free methods (sharp interface methods)

Considering a surface PDE and a bulk domain that is represented by some geometrically unfitted mesh for the corresponding hypersurface, the surface PDE not necessarily needs to be extended to the bulk domain at all to obtain some geometrically unfitted mesh-based method. The solution to a surface PDE may be searched in some FEM approximation space that is constructed using a given bulk mesh, even if the weak formulation of the surface PDE solely considers function values on the hypersurface of interest. As a matter of course, the weak formulation needs to be stabilized in a suitable manner in this case to obtain a well-posed problem with a solution that has a unique representation in the considered approximation space. Alternatively, the ill-posedness of the discrete problem which is obtained without any stabilization can also be dealt with at the algebraic level.

Such extension-free methods based on bulk meshes and associated FEM approximation spaces are known by the name *trace FEMs* or *sharp interface FEMs* and they are also called cutFEMs for surface PDEs by some authors. They were first proposed and investigated in Olshanskii et al. (2009); Deckelnick et al. (2014) for PDEs on static hypersurfaces that are represented implicitly using a level set function. Subsequently, Burman et al. (2016a); Grande et al. (2016) analyzed various stabilization mechanisms that can be applied on top, where Grande et al. particularly investigate a generalization of the approach which allows for higher order approximations in space.

Methods for coupled bulk–surface PDEs

Also by combining the ideas of geometrically unfitted mesh-based methods for bulk PDEs and of geometrically unfitted mesh-based methods for surface PDEs approaches have been obtained that are suitable for dealing with coupled bulk–surface PDEs. Phase field extension (diffuse domain and interface) and FD or finite element based methods for bulk–surface models on evolving geometries are considered in Teigen et al. (2009, 2011); Garcke et al. (2014); Marth and Voigt (2014). An analysis of similar methods for bulk–surface models on static geometries is presented in Abels et al. (2015). Recently, a method which combines ideas of cutFEMs for bulk PDEs and of trace FEMs

for surface PDEs to deal with a system of linear elliptic bulk–surface PDEs on static geometries has been considered in Burman et al. (2016b). In this thesis, we develop methods for coupled bulk–surface PDEs that build upon the UDG method for bulk PDEs which is mentioned above.

1.4. Studying spatial features in basic cell polarization models using a classical mesh-based finite element scheme

Cell polarization is a process happening in cells, the most basic structural units of all known living organisms. More precisely, it is the asymmetric redistribution of proteins and lipids in the plasma membrane of a cell (cf. Orlando and Guo, 2009), where asymmetric means that proteins and lipids localize to specific areas of the cell membrane. Usually, this localization occurs at the cytoplasmic side of the cell membrane and requires the recruitment of cytoplasmic proteins and lipids to the cell membrane. Cell polarization and particularly the spatial reorganization of membrane proteins are fundamental for cell division, cell migration and chemotaxis, as well as for signal transmission in neurons. For more information, see e.g. Orlando and Guo (2009) and the references given therein.

The yeast *Saccharomyces cerevisiae*, which can be seen in Figure 1.3, is often chosen as a model organism for studying cell polarization. In this organism, the redistribution process is caused by the shuttling of proteins known as small Rho GTPases, such as Cdc42 and Rho, between an active membrane-bound form and an inactive cytosolic form. This process results in the formation of a molecule cluster on the membrane. Cell polarization subsequently triggers a cell division mechanism known as budding, which happens on a larger time scale. During formation of the molecule cluster, the shape of a yeast cell remains virtually static.

In Giese, Eigel, Westerheide, Engwer and Klipp (2015a), the author of this thesis and co-authors use a classical mesh-based method to investigate the influence of spatial features in mathematical models for cell polarization in yeast cells. Representing the cytosol of a single cell as a static bulk domain Ω, and the enveloping cell membrane as the static hypersurface Γ that is formed by the boundary of Ω, the membrane–cytosol shuttling of proteins is described using members of the class of bulk–surface models which has been introduced in Section 1.2. In particular, two bulk–surface models are investigated that can be formulated by the static geometry special case $\mathbf{v} \equiv \underline{\mathbf{0}}$ of equations (1.8) and specific choices of the data functions $\mathcal{D}_b$, $\mathcal{D}_s$, $f_b(u_b)$, $f_s(u_s)$, $f_{b,s}(u_b, u_s)$ and $f_{s,b}(u_b, u_s)$. In the study, the two models are enriched by cavities in the bulk domain Ω and by an associated, additional boundary condition for equation (1.8a) on the boundaries $\partial\Omega \setminus \Gamma$ of those cavities. Both the cavities and the corresponding boundary condition model intracellular structures which proteins that are taken into account by the model can not enter or leave.

1.4.1. Basic cell polarization models

Each model which is investigated in our study considers one of two well-known polarization mechanisms. The first mechanism shows classical Turing-instability patterns and is employed in a model known as the *Goryachev model (GOR model)*. The latter was introduced in Goryachev and Pokhilko (2008) as a reduced model which captures the essential features of a more complex model that has been introduced in the supplementary material of the same work. Our study deals with a variant of the GOR model which we refer to as the *simplified GOR model* in this thesis. Without incorporating the bulk domain cavities and the associated boundary condition which have been discussed above, the simplified GOR model is characterized by equations (1.8) with $\mathbf{v} \equiv \underline{\mathbf{0}}$, together with the data functions specified in Table 1.1. Unlike the standard GOR model, where the term E_c in the data function $f_{s,b}^{\text{main}}(u_b, u_s)$ is a time-dependent field that is implicitly defined via the concentration u_s on the whole membrane, the simplified GOR model uses a constant parameter E_c. This allows for clusters on the membrane which recruit the entire pool of the cytosolic molecule with concentration u_b. Although the latter can not be observed in practical experiments (Goryachev and Pokhilko, 2008, Section "Model reduction" in supplementary material), this simplication is quite common, given that the resulting model is easier to handle from a mathematical point of view.

The other polarization mechanism exhibits wave-pinning dynamics. It is implemented by a model known as the *wave-pinning model (WP model)*, which was originally introduced in Mori et al. (2008) as a system of reaction–diffusion equations on some one-dimensional geometry representation. In Giese, Eigel, Westerheide, Engwer and Klipp (2015a), we introduce a consistent, higher-dimensional extension of this model, since one-dimensional models provide only limited capabilities in representing spatial features and usually do not account for a suitable surface to volume ratio. Our extension represents cells and their membrane as d-dimensional bulk domains and associated $(d-1)$-dimensional hypersurfaces. As with the simplified GOR model, this bulk–surface formulation of the WP model is characterized by equations (1.8) with $\mathbf{v} \equiv \underline{\mathbf{0}}$, together with the data functions specified in Table 1.1. To incorporate the bulk domain cavities and the associated boundary condition which have been discussed above, the model equations need to be changed slightly. See Giese, Eigel, Westerheide, Engwer and Klipp (2015a) for details.

1.4.2. A classical mesh-based finite element scheme

Without incorporating bulk domain cavities and the associated boundary condition, the classical mesh-based method which we employed in Giese, Eigel, Westerheide, Engwer and Klipp (2015a) to solve the model equations numerically can be described in the following way.

Model	Data function	Definition/property
GOR & WP	$\mathcal{D}_b$	some scalar constant in $\mathbb{R}$
	$\mathcal{D}_s$	some scalar constant in $\mathbb{R}$
	$f_b(u_b)$	0
	$f_s(u_s)$	0
	$f_{b,s}(u_b, u_s)$	$-f_{s,b}(u_b, u_s)$
	$f_{s,b}(u_b, u_s)$	$f_{s,b}^{\text{main}}(u_b, u_s) + k_{\text{stimulus}}\, u_b$
	k_{stimulus}	some scalar field $\Gamma \times (0, T] \to \mathbb{R}$
GOR	$f_{s,b}^{\text{main}}(u_b, u_s)$	$\alpha E_c u_s^2 u_b + \beta E_c u_s u_b - \gamma u_s$ with $\alpha, \beta, \gamma, E_c \in \mathbb{R}$
WP	$f_{s,b}^{\text{main}}(u_b, u_s)$	$u_b\Big(k_0 + \frac{\gamma u_s^2}{K^2 + u_s^2}\Big) - \delta u_s$ with $k_0, \gamma, K, \delta \in \mathbb{R}$

Table 1.1.: Data functions that characterize the simplified GOR model and the WP model, when being employed in the static geometry special case $\mathbf{v} \equiv \underline{\mathbf{0}}$ of equations (1.8). Suitable values of the parameters of $f_{s,b}^{\text{main}}(u_b, u_s)$ will be given later on in this thesis, together with specific choices of $\mathcal{D}_b$, $\mathcal{D}_s$ and k_{stimulus}.

Weak formulation

Our numerical method combines ideas of classical FEMs and SFEMs and is similar to the approach which is investigated in Elliott and Ranner (2013) in the context of an elliptic bulk–surface model. First, we derive a suitable weak formulation of the static geometry special case $\mathbf{v} \equiv \underline{\mathbf{0}}$ of model equations (1.8). Formally, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(\int_\Omega u_b\, \varphi_b \,\mathrm{d}x\Big) + \int_\Omega \mathcal{D}_b \nabla u_b \cdot \nabla \varphi_b \,\mathrm{d}x \quad = c_b(u_b, u_s, \varphi_b, \Omega), \tag{1.9a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(\int_\Gamma u_s\, \varphi_s \,\mathrm{d}\sigma\Big) + \int_\Gamma \mathcal{D}_s \nabla_\Gamma u_s \cdot \nabla_\Gamma \varphi_s \,\mathrm{d}\sigma = c_s(u_b, u_s, \varphi_s, \Gamma), \tag{1.9b}$$

by multiplying equations (1.8a) and (1.8c) by some bulk test function φ_b and by some surface test function φ_s of suitable regularity, respectively, and by subsequent application of integration by parts formulas for bulk domains and hypersurfaces. On the right-hand side, we have terms that are defined by

$$c_b(u_b, u_s, \varphi_b, \Omega) := \int_{\partial\Omega} \big(\mathcal{D}_b \nabla u_b \cdot \boldsymbol{\nu}\big)\varphi_b \,\mathrm{d}\sigma + \int_\Omega f_b(u_b)\, \varphi_b \,\mathrm{d}x,$$

$$c_s(u_b, u_s, \varphi_s, \Gamma) := \int_\Gamma \big(f_{s,b}(u_b, u_s) + f_s(u_s)\big)\varphi_s \,\mathrm{d}\sigma.$$

Due to boundary condition (1.8b), the first of these terms takes the form

$$c_b(u_b, u_s, \varphi_b, \Omega) = \int_{\partial\Omega} f_{b,s}(u_b, u_s)\, \varphi_b \, \mathrm{d}\sigma + \int_{\Omega} f_b(u_b)\, \varphi_b \, \mathrm{d}x.$$

Given initial values $u_b(\cdot, 0) \in H^1(\Omega)$ and $u_s(\cdot, 0) \in H^1(\Gamma)$, the resulting weak formulation is to look for functions $u_b \colon \Omega \times [0, T] \to \mathbb{R}$ and $u_s \colon \Gamma \times [0, T] \to \mathbb{R}$ with $u_b(\cdot, t) \in H^1(\Omega)$ and $u_s(\cdot, t) \in H^1(\Gamma)$, such that equations (1.9) hold for all test function pairs $(\varphi_b, \varphi_s) \in H^1(\Omega) \times H^1(\Gamma)$, and for each $t \in (0, T]$. Here, $H^1(\Omega)$ denotes the traditional Sobolev space which is used to seek weak solutions to elliptic bulk PDEs in Ω, and $H^1(\Gamma)$ is its natural counterpart for weak solutions to elliptic surface PDEs on Γ.

Meshes and discrete approximation spaces

We separate discretization in space and time by applying the well-known method of lines, see e.g. Schiesser (1991). To construct finite-dimensional function spaces $V_{b,h}(\Omega_h) \subset H^1(\Omega_h)$ and $V_{s,h}(\Gamma_h) \subset H^1(\Gamma_h)$ that are capable of approximating the solution at each time t on some discrete reconstruction $(\Omega_h, \Gamma_h := \partial\Omega_h)$ of the geometry (Ω, Γ), we use a triangular mesh $\mathcal{T}_h(\Omega_h)$ which approximates the bulk domain Ω, and the surface mesh $\mathcal{T}_h(\Gamma_h)$ which is made up by the outer entities of $\mathcal{T}_h(\Omega_h)$. See Figure 1.5a for illustrations.

In particular, we construct standard finite element spaces of continuous, piecewise linear functions over Ω_h and Γ_h. These are given by

$$V_{b,h}(\Omega_h) := \Big\{ v_{b,h} \in C^0(\Omega_h) \;\Big|\; v_{b,h}\big|_K \in P_1(K)\ \forall K \in \mathcal{T}_h(\Omega_h) \Big\},$$
$$V_{s,h}(\Gamma_h) := \Big\{ v_{s,h} \in C^0(\Gamma_h) \;\Big|\; v_{s,h}\big|_K \in P_1(K)\ \forall K \in \mathcal{T}_h(\Gamma_h) \Big\},$$

where $P_1(K)$ denotes the space of polynomial functions of total degree less than or equal to 1 over a mesh element K.

Employing Lagrange elements of order 1 on each mesh, we obtain classical Lagrange basis functions of polynomial degree 1 for each space. For $V_{b,h}(\Omega_h)$, these are characterized by $\varphi_{b,h,i}\big(\underline{\mathbf{x}^{\mathbf{j}}}\big) = \delta_{ij}$, where $\varphi_{b,h,i}$ denotes the i-th basis function, $\underline{\mathbf{x}^{\mathbf{j}}}$ denotes the j-th element of the corresponding set of Lagrange nodes, and $\delta_{ij} \in \{0, 1\}$ is the well-known Kronecker delta. Analogously, the basis functions for $V_{s,h}(\Gamma_h)$ are characterized by $\varphi_{s,h,i}\big(\underline{\mathbf{x}^{\mathbf{j}}}\big) = \delta_{ij}$, where $\underline{\mathbf{x}^{\mathbf{j}}}$ now denotes the j-th element of the set of Lagrange nodes associated with the surface mesh $\mathcal{T}_h(\Gamma_h)$. Given that we choose Lagrange elements of order 1, each set of Lagrange nodes equals the set of vertices of the corresponding mesh.

Discretization in space

Starting from the weak formulation of the model equations, we discretize in space by replacing the geometry (Ω, Γ) by its discrete reconstruction (Ω_h, Γ_h),

and by restricting the set of admissible functions, considering only functions that are representable using the discrete function spaces $V_{b,h}(\Omega_h)$ and $V_{s,h}(\Gamma_h)$. This yields the following semidiscretization.

Semidiscretization 1.4.1. *Let $u_{b,h}(\cdot,0) \in V_{b,h}(\Omega_h)$ and $u_{s,h}(\cdot,0) \in V_{s,h}(\Gamma_h)$ be suitable discrete approximations of the given initial values. We seek a pair of semidiscrete functions $u_{b,h}\colon \Omega_h \times [0,T] \to \mathbb{R}$ and $u_{s,h}\colon \Gamma_h \times [0,T] \to \mathbb{R}$ with $u_{b,h}(\cdot,t) \in V_{b,h}(\Omega_h)$ and $u_{s,h}(\cdot,t) \in V_{s,h}(\Gamma_h)$, such that for all test function pairs $(\varphi_{b,h}, \varphi_{s,h}) \in V_{b,h}(\Omega_h) \times V_{s,h}(\Gamma_h)$ and for each $t \in (0,T]$:*

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{\Omega_h} u_{b,h}\,\varphi_{b,h}\,\mathrm{d}x\right) + \int_{\Omega_h} \mathcal{D}_b \nabla u_{b,h} \cdot \nabla \varphi_{b,h}\,\mathrm{d}x &= c_b(u_{b,h}, u_{s,h}, \varphi_{b,h}, \Omega_h),\\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{\Gamma_h} u_{s,h}\,\varphi_{s,h}\,\mathrm{d}\sigma\right) + \int_{\Gamma_h} \mathcal{D}_s \nabla_\Gamma u_{s,h} \cdot \nabla_\Gamma \varphi_{s,h}\,\mathrm{d}\sigma &= c_s(u_{b,h}, u_{s,h}, \varphi_{s,h}, \Gamma_h).
\end{aligned}$$

Since all functions in $V_{b,h}(\Omega_h)$ and $V_{s,h}(\Gamma_h)$ can be represented by means of the fixed Lagrange basis functions of the corresponding discrete function space, Semidiscretization 1.4.1 results in a system of partial differential equations in one single continuous variable, namely in the time variable t. Equations of this type are known as ordinary differential equations.

Discretization in time and the resulting fully discrete scheme

For discretization in time, various numerical methods for ordinary differential equations can be used. Information about the method which has been employed to perform simulations in our study can be found in the study's supplementary material Giese, Eigel, Westerheide, Engwer and Klipp (2015b, Section 2). With the data functions that are specified in Table 1.1, discretization in time results in a system of nonlinear algebraic equations that can be solved, for instance, using Newton's method.

The resulting fully discrete scheme can be implemented using tools provided by standard PDE software frameworks, even if they do not support surface meshes and finite element spaces based on surface meshes. In implementations, the space $V_{s,h}(\Gamma_h)$ can optionally be replaced by the bulk space $V_{b,h}(\Omega_h)$ and additional constraints which disable those degrees of freedom that are associated with interior Lagrange nodes given by the mesh $\mathcal{T}_h(\Omega_h)$. Conceptually, these constraints are similar to constraints that are frequently used to implement Dirichlet boundary conditions. Exploiting the bulk space $V_{b,h}(\Omega_h)$ in this way is possible since its basis functions have suitable properties when being restricted to Γ_h. Performing this restriction for the subset of basis functions that are associated with Lagrange nodes on Γ_h exactly yields the corresponding Lagrange basis for $V_{s,h}(\Gamma_h)$. For more details on this aspect and more details on the fully discrete numerical scheme in general, we again refer to the study's supplementary material Giese, Eigel, Westerheide, Engwer and Klipp (2015b, Section 2).

1.4.3. Results of the study

In our study, we targeted different spatial features by means of four different sets of experiments which will be summarized in the following. See Giese, Eigel, Westerheide, Engwer and Klipp (2015a) for a detailed description.

In a first setup, we introduced a protrusion to an otherwise fully circular cell in order to understand the influence of a cell's shape on the polarization behavior of the models. Exciting both models with different kinds of external signals (i.e. by means of particular choices of the data function k_{stimulus} which has been introduced in Table 1.1), we compared simulation results obtained for the cell with a protrusion with simulation results obtained for the corresponding fully circular cell which we started from. With our simulations, we could show that protrusions locally limit molecule aggregations. In both models, they locally act as negative feedback and lower the sensitivity to external signals in the same region.

In a second set of experiments, we performed simulations for circular cells with cell diameters between $1.5\mu m$ and $15\mu m$. These experiments demonstrated that, given fixed kinetic parameters, there exists an optimal cell size with regard to cell polarization. For both models, we observed that polarization is not possible for small cells on the one hand, and takes very long for large cells on the other hand.

In a third setup, we placed various diffusion barriers on the membrane of a cell with a protrusion. With our simulations, we were able to show that those barriers have the potential to amplify formation of clusters on the membrane and are therefore able to accelerate, stabilize and steer cell polarization.

By means of a fourth set of experiments, we examined the effects of diffusion barriers in the cytosolic part of a cell, such as organelles. We introduced organelles of different size and shape into the cytosol and performed a vast number of simulations with different noisy signals on the cell membrane for each geometrical setup. Our simulations showed that, despite fast cytosolic diffusion, reduced transport due to organelles leads to a change in the polarization behavior. In particular, molecule clustering very close to organelles turned out to be unlikely. At the same time, molecule clustering tends to be intensified in a large-scale neighborhood of the organelles and in regions without organelles.

In conclusion, our computer simulations illustrated that the influence of spatial features like cell shape, cell size and inhomogeneities on the membrane or in the cytosol can be quite dramatic. Therefore, they should be considered in modeling and simulation of intracellular processes of spatial nature.

1.5. Challenges in applications with PDEs on complex-shaped surfaces

The application presented in Section 1.4 demonstrated the need for numerical tools which allow for comprehensive simulation studies. However, carrying

out our specific study also revealed drawbacks of the simulation framework which we employed. Each new geometrical setup which has been investigated required generating a new mesh. Generating these meshes was a tedious and time-consuming task for the person performing the simulations. Moreover, we studied cell polarization only in cells of simple geometry, which do not change their shape within the time period under consideration.

The framework would not be suitable when investigating similar processes in other cell types. Developing neurons, for instance, show complex geometrical shapes that change rapidly during the redistribution process which leads to polarized cells. Example geometries can be seen in Figure 1.6. It should be noted that the intracellular actin filaments depicted in Figure 1.6b and Figure 1.6c push against the outer cell membrane. They thus determine the shape of a neuron. This connection is clearly visible in Figure 1.6b.

When it comes down to designing a more advanced simulation framework that is suitable for investigating processes like cell polarization in objects which exhibit complex geometrical setups, challenges have to be met which we name as follows:

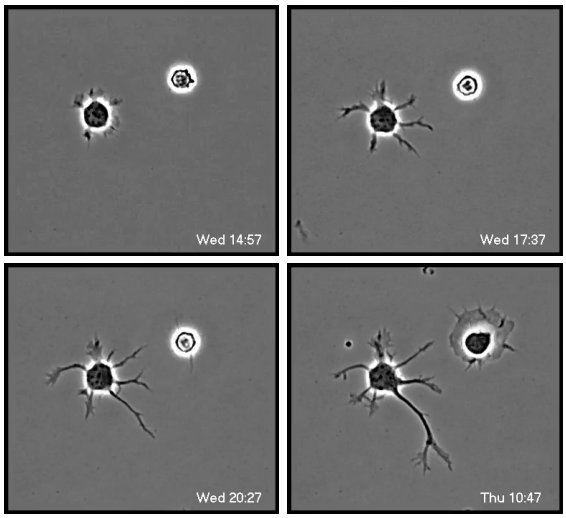
1. Complex static geometries
2. Evolving geometries
3. Geometry description
4. Conservation properties
5. Higher order schemes
6. Mixed dimensionality
7. Condition of the resulting systems of algebraic equations
8. Implementation

Details on these challenges are given below.

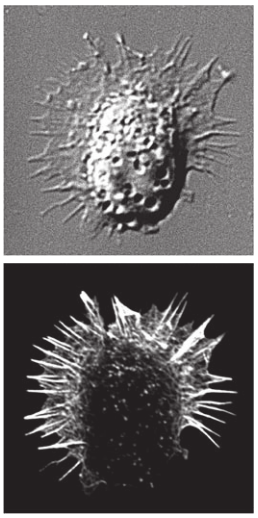
Complex static geometries:

Regarding spatial geometries, the numerical methods should offer efficient means of dealing with PDEs on bulk domains with complex-shaped boundaries and PDEs on complex-shaped hypersurfaces. As seen in Figure 1.6, especially geometrical setups that are given by high-resolution imaging data often comprise bulk domain boundaries (hypersurfaces in general) which resolve microscopic geometrical structures. Incorporating the full level of available geometric detail can be necessary for obtaining correct simulation results. For instance, the more accurate the geometry description is, the more accurate boundary conditions like equation (1.8b) can be imposed.

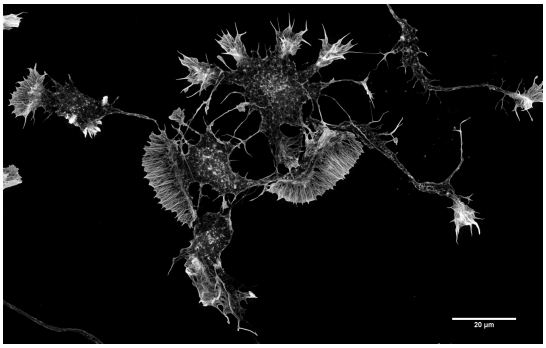
In general, classical mesh-based methods can be used for dealing with PDEs on complex static geometries. However, for each individual geometrical setup, an associated mesh needs to be generated at the beginning of the simulation. Meshes that have a sufficient quality to be suitable for classical mesh-based methods generally can not be generated in a fully automated



(a) A normal neuron (left) and a mutated neuron (right), cultured for 20 hours.



(b) A neuron grown in culture for 44 hours (top) and its actin filaments (bottom).



(c) Actin filaments in mouse cortical neurons grown in vitro for 2.5 days.

Figure 1.6.: Microscopy images of developing neurons. Image sequence (a) shows phase-contrast microscopy images from a time-lapse series[6] courtesy of Erik W. Dent (University of Wisconsin) and Frank B. Gertler (MIT). Part (b) shows a differential interference contrast microscopy image (top) and a fluorescence microscopy image (bottom) of a neuron which has been stained for F-actin. Adapted by permission from Macmillan Publishers Ltd: Nature Cell Biology (Dent et al., 2007), © 2007. Image (c) was taken by Howard Vindin[7] using summed slices z-projection on a confocal laser scanning microscope. It is distributed under the CC-BY-SA-4.0 license[8].

[6] http://techtv.mit.edu/videos/915

[7] https://commons.wikimedia.org/wiki/File:SUM_110913_Cort_Neurons_2.5d_in_vitro_488_Phalloidin_no_perm_4_cmle-2.png

[8] https://creativecommons.org/licenses/by-sa/4.0/legalcode

manner. Especially for complex-shaped three-dimensional structures, the process may require lots of user interaction. With classical mesh-based methods, the discrete solution spaces and the geometrical setup furthermore are directly linked via the computational mesh. It is hence not possible to avoid spaces with a number of degrees of freedom which is unnecessarily high, given a certain targeted precision of the solution, while still having the possibility of incorporating the full level of geometric detail.

Geometrically unfitted methods do not suffer from both issues. Meshes covering complex spatial geometries may have a very comfortable structure for which automated mesh generation is possible. By employing meshes of this kind, unfitted methods decouple discrete solution spaces from the geometrical setup. However, for taking into account the actual geometry, precise information on this geometry is nevertheless required. Given that the computational mesh does not contain such information, the latter has to be supplied by other means. Those means need to be able to provide detailed representations of complex static geometries in an efficient manner.

Evolving geometries:

Additional complexity arises from time-dependent geometrical setups.

On the one hand, regarding the model equations, the geometrical evolution is possily driven by the state of the system which is modeled and is coupling back to this state. This leads to additional couplings in the model equations, that have to be taken into account by the numerical methods which deal with those equations. Furthermore, as seen in equations (1.5) and in equations (1.8), the geometrical evolution drives advection terms which can be challenging to deal with.

On the other hand, moving bulk domains and hypersurfaces might exhibit large deformations with strong anisotropies or even topology changes, as depicted in Figure 1.4 and Figure 1.6a. In classical mesh-based methods, this entails the necessity of frequent remeshing to maintain a mesh of sufficient quality, if at all possible without user interaction. For geometrically unfitted methods, this is not an issue since it is possible to employ a computational mesh which covers the evolving geometry for every point in time. Instead, data management can be more difficult. Depending on the precise definition of the discrete approximation spaces that are constructed using the mesh, different sets of mesh elements might contribute to spaces associated with different points in time. Furthermore, the geometry needs to be represented in a way that is suitable for describing the geometrical effects listed above.

Geometry description:

A geometry description needs to be chosen that is suitable for developing numerical schemes which successfully address the geometry-related challenges discussed above. In the course of this, descriptions are preferable that are capable of representing potentially time-dependent, complicated

geometries or sufficiently accurate discrete reconstructions of these geometries in a comfortable and efficient manner, without requiring much user interaction.

The freedom to choose an appropriate geometry description exists only for geometrically unfitted methods since they do not employ the computational mesh for this purpose. Classical mesh-based methods are inherently bound to the computational mesh. For them, another type of geometry description can nevertheless be used during mesh generation at the beginning of the simulation. Furthermore, in the case of evolving geometries, it could serve as supplementary information for improving the mesh quality by performing remeshing.

Conservation properties:
As discussed for the continuity equations in Section 1.1, Section 1.2 and Section 1.4.1, conservation properties are embedded in many PDEs that are practically relevant. These conservation properties represent physical principles that often are a fundamental hypothesis in mathematical models comprising equations of this type.

For simulating those models, numerical methods should be developed which recover discrete analogues to the underlying conservation properties. They should be designed in such a way that approximations of the solution are obtained which reflect the considered physical principles in the best possible way. In particular, end users of simulation software usually expect discrete conservation properties that reflect properties of the original continuous quantities, not of some artificial quantities that are introduced in the course of the discretization process.

Developing such a numerical method can be challenging if techniques are used that are suitable for dealing with the geometrical challenges listed above. Particularly for geometrically unfitted methods, the decoupling of discrete solution spaces from the geometrical setup implies that special attention needs to be payed to the preservation of conservation properties.

Higher order schemes:
Sometimes solutions of high accuracy are required. For being able to increase the accuracy of numerical approximations significantly, it is beneficial to employ techniques which allow for obtaining higher order schemes, i.e., schemes that are at least second order accurate. However, it can be difficult to maintain higher order spatial accuracy, especially for complex spatial geometries. One particular difficulty lies in the representation of bulk domain boundaries and hypersurfaces. If discrete reconstructions of those structures are not sufficiently precise, they can have negative influence on the order of convergence of a scheme.

Mixed dimensionality:
Numerical approaches to solving PDEs need to account for the dimension-

ality of the geometric objects on which the considered PDEs are posed. On the one hand, this dimensionality itself can be mixed, as with the systems of equations which make up the bulk–surface models in Section 1.2 and Section 1.4.1. On the other hand, mixed dimensionality needs to be taken care of, if the discrete solution of a surface PDE is represented by means of an approximation space that is constructed using a mesh of some bulk neighborhood of the hypersurface. In this case, the dimension of the geometry of interest differs from the geometrical dimension of the mesh, respectively its associated approximation space. As a result, the discrete solution of the surface PDE does not have a unique representation in the approximation space. More precisely, special measures need to be taken to gain control in normal direction to the hypersurface. This is particularly the case with geometrically unfitted methods for surface PDEs.

Furthermore, the design of numerical methods can involve discretizing different parts of a PDE using different approaches. For surface PDEs, this can implicate that terms of different dimensions are mixed, e.g. terms resulting from extension-based unfitted methods and terms which arise in extension-free unfitted methods. In this case, a proper scaling of those terms is required.

Condition of the resulting systems of algebraic equations:
Numerical schemes for PDEs result in associated discrete problems. These discrete problems can be described by systems of algebraic equations whose solutions can be obtained using computers. The limited precision of the floating point arithmetic which is employed by computers inherently leads to errors that already enter the description of the discrete problems. Solutions obtained using computers will hence contain an unavoidable error. For this reason, numerical schemes should result in discrete problems that are posed in such a way that this unavoidable error can be controlled. In particular, it should stay within manageable limits which assure that solutions are practically usable.

It is well-known that special attention needs to be payed to this issue when designing geometrically unfitted methods.

Implementation:
In order to be practically applicable, numerical methods should allow for an easy implementation using efficient data structures and algorithms. Ideally, the methods should enable implementers to utilize tools that are freely available in frameworks for numerical software development or other software libraries.

Numerical schemes can be considered as being appropriately designed for the type of applications which we have in mind, if they enable to cope with the challenges listed above. They should particularly overcome the difficulties that arise from the combination of those challenges.

1.6. Contributions and outline of this thesis

In this thesis, we show how the unfitted discontinuous Galerkin (UDG) method (Bastian and Engwer, 2009; Engwer, 2009), which we sketched in Section 1.3.2, can be employed to obtain a simulation framework which successfully deals with the eight challenges that have been discussed in Section 1.5.

We construct numerical schemes which take advantage of the fact that the UDG method is tailor-made for continuity equations in complex-shaped bulk domains. Being based on classical DG formulations for bulk equations, the method recovers discrete analogues to underlying conservation properties of those equations, naturally handles advection terms like those which arise from evolving geometries, and allows for higher order schemes in a straightforward way (cf. Section 1.3.1). Moreover, it supports various mechanisms which ensure that well-conditioned discrete problems are obtained (cf. Section 1.3.2). By combining the method with a level set description of the geometry, complex static geometries and time-dependent geometrical setups can be treated in a comfortable and efficient manner. Furthermore, a free implementation is available, which is ready to use (Engwer and Heimann, 2012).

A central contribution of this thesis is to show how the UDG method can be employed to deal with continuity equations on complex-shaped hypersurfaces and on evolving hypersurfaces. By combining ideas of extension-based methods for surface PDEs with concepts of extension-free sharp interface FEMs, we allow for schemes that recover discrete analogues to conservation properties on the hypersurface and exhibit higher order convergence. In addition, our schemes can be realized using existing implementations of the UDG method for bulk PDEs.

Outline

The thesis is structured as follows. In Chapter 2, we begin by introducing concepts from elementary differential geometry which are essential for formulating PDEs on hypersurfaces and for dealing with those equations, particularly surface differential operators and integral calculus that builds upon them. In the course of this, we lay special focus on properties which can be exploited when using a level set description of the geometry, and calculus resulting from those properties. To our knowledge, parts of our form of presentation and some of the considerations relating thereto can not be found elsewhere in the literature, even though they help a lot in designing level set extension based numerical schemes for systems with surface PDEs. The latter can be seen from the construction of our schemes later on in this thesis.

In Chapter 3, we continue with further important mathematical background regarding our schemes, focussing on tools from pure and numerical mathematics which are important ingredients of those schemes. The chapter comprises a discussion of the fundamentals of bulk/surface continuity equations which will reveal their underlying conservation properties, an extensive introduction

to those fitted DG methods for elliptic and parabolic bulk equations that are relevant within the scope of this thesis, and a detailed description of the level set framework.

Chapter 4 and Chapter 5 are the central chapters of this thesis. In Chapter 4, we develop and analyze a new type of UDG schemes for bulk–surface models on static geometries. These schemes particularly build upon transferring the UDG method to PDEs on hypersurfaces. In the course of this, special focus is layed on recovering discrete analogues to the models' underlying conservation properties and we investigate approaches that increase numerical robustness. In Chapter 5, we subsequently present how the approaches from Chapter 4 can be extended to obtain UDG schemes for bulk–surface models that comprise continuity equations on evolving geometries. We introduce a special operator splitting approach which allows for recycling our schemes from Chapter 4 in such a way that overall schemes are obtained which still recover discrete analogues to the models' underlying conservation properties. Moreover, we derive and investigate a novel conservative UDG scheme for an essential type of continuity equations on evolving hypersurfaces. The latter type of equations corresponds to one component feeding into our operator splitting approach.

At the end of the latter two chapters, we summarize our findings and discuss future perspectives. A final conclusion is drawn in Chapter 6.

The appendix of this thesis contains useful additional information which is not supposed to distract the reader from the main topic of the thesis. In Chapter A, we describe software which has been created as part of this thesis, aiming at reproducibility of our numerical results. In Chapter B, we focus on the condition of an important algebraic problem which arises from all discretization methods that are similar to those introduced in this thesis, namely the problem of solving a system of linear equations. The chapter particularly provides the means for us to investigate the conditioning properties of our schemes. Finally, in Chapter C, we recall some concepts from elementary differential geometry that are complementary to those which are introduced in Chapter 2, primarily basic terminology and facts from elementary differential geometry.

2. Essential concepts from elementary differential geometry

In this chapter, we introduce concepts from elementary differential geometry which are essential for formulating PDEs on hypersurfaces and for dealing with those equations. We begin in Section 2.1 and Section 2.2 by defining and characterizing surface differential operators, and by introducing the notion of curvature. By the end of Section 2.2, we will have taken a close look at the surface divergence operator. We continue to investigate this operator in Section 2.3, focussing on its behavior in the level set framework. In Section 2.4, we introduce important theory from integral calculus on hypersurfaces. The concepts presented in Sections 2.1–2.4 are integrated with time-dependent problems in Section 2.5. Finally, in Section 2.6, we complement this chapter by looking at additional calculus on evolving hypersurfaces.

Please note that we are using basic terminology and facts from elementary differential geometry in this chapter, some of which are recalled in Appendix C, particularly the notion of hypersurfaces and considerations on differentiable functions on hypersurfaces. Both can be helpful in understanding the theory which is presented in this chapter.

Unless otherwise stated, let $\mathcal{M}$ be a smooth $(d-1)$-dimensional hypersurface embedded in $\mathbb{R}^d$ which is oriented by a field $\boldsymbol{\nu}_{\mathcal{M}} \colon \mathcal{M} \to \mathbb{R}^d$ of unit normal vectors to $\mathcal{M}$. A field defined on $\mathcal{M}$, like $\boldsymbol{\nu}_{\mathcal{M}}$, is called a *surface field* (on $\mathcal{M}$). Let $\eta \colon \mathcal{M} \to \mathbb{R}$ and $\boldsymbol{\xi} \colon \mathcal{M} \to \mathbb{R}^d$ be a surface scalar field and a surface vector field, respectively, that have differentiable extensions $\hat{\eta}$ and $\hat{\boldsymbol{\xi}}$ to a neighborhood $\mathcal{N}(\mathcal{M}) \subset \mathbb{R}^d$ with $\mathcal{M} \subset \mathcal{N}(\mathcal{M})$ and $\operatorname{meas}_{\mathbb{R}^d}\bigl(\mathcal{N}(\mathcal{M})\bigr) > 0$. Wherever we require that also $\boldsymbol{\nu}_{\mathcal{M}}$ has a differentiable extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ in the following, we implicitly assume that $\boldsymbol{\nu}_{\mathcal{M}}$ has exactly the same extension property as $\boldsymbol{\xi}$, without explicitly using the exact wording of the above assumption.

2.1. Surface differential operators

The definitions that we use are based on the extension property of the fields η and $\boldsymbol{\xi}$, and on the operator on the space of surface vector fields which projects each surface vector field onto the subspace of tangential surface vector fields. The latter consists of all surface vector fields $\boldsymbol{\zeta}$ on $\mathcal{M}$ which map into the tangent spaces of $\mathcal{M}$, so that $\boldsymbol{\zeta} \cdot \boldsymbol{\nu}_{\mathcal{M}} = 0$ at each $\underline{\mathbf{x}} \in \mathcal{M}$. The operator can be defined as

$$\mathcal{P}_{\mathcal{M}} := \mathcal{I} - \boldsymbol{\nu}_{\mathcal{M}} \, (\boldsymbol{\nu}_{\mathcal{M}})^{\mathrm{tr}}, \tag{2.1}$$

where $\mathcal{I}$ denotes the identity operator. It is easy to see that $\mathcal{P}_{\mathcal{M}}$ is idempotent, i.e. $\mathcal{P}_{\mathcal{M}}^2 = \mathcal{P}_{\mathcal{M}}$, self-adjoint, i.e. $\mathcal{P}_{\mathcal{M}}^{\mathrm{tr}} = \mathcal{P}_{\mathcal{M}}$ with $\mathcal{P}_{\mathcal{M}}^{\mathrm{tr}}$ denoting the adjoint operator, and that $\mathcal{P}_{\mathcal{M}}\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}} \equiv 0$.

The *surface gradient* of η (on $\mathcal{M}$) can be defined as

$$\operatorname{grad}_{\mathcal{M}} \eta := \big(\mathcal{P}_{\mathcal{M}} \nabla \hat{\eta}\big)^{\mathrm{tr}}, \tag{2.2}$$

where ∇ denotes the transposed classical gradient operator in $\mathbb{R}^d$. Defining the operator

$$\nabla_{\mathcal{M}} := \mathcal{P}_{\mathcal{M}} \circ \nabla, \tag{2.3}$$

the surface gradient of η can be written as

$$\nabla_{\mathcal{M}} \eta = \mathcal{P}_{\mathcal{M}} \nabla \hat{\eta} = \big(\operatorname{grad}_{\mathcal{M}} \eta\big)^{\mathrm{tr}},$$

yielding a transposed version. By definition, the surface gradient is a tangential vector field. Since it is the standard gradient's component tangential to $\mathcal{M}$, it is independent of the choice of the extension $\hat{\eta}$, i.e. well-defined, and also called the *tangential gradient* (on $\mathcal{M}$). This becomes more obvious if we consider the following remark.

Remark 2.1.1 (Local representation of $\nabla_{\mathcal{M}}\eta$ using the tangent spaces of $\mathcal{M}$)**.** *For every orthonormal basis $\{\underline{\mathbf{v_i}}\}_{i=1,\dots,d}$ for $\mathbb{R}^d$, the surface gradient of η can be expressed as*

$$\nabla_{\mathcal{M}} \eta = \nabla \hat{\eta} - \big(\nabla \hat{\eta} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}\big)\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}} = \sum_{i=1}^{d} \big(\nabla \hat{\eta} \cdot \underline{\mathbf{v_i}}\big)\underline{\mathbf{v_i}} - \big(\nabla \hat{\eta} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}\big)\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}. \tag{2.4}$$

At each fixed $\underline{\mathbf{x}} \in \mathcal{M}$, we can choose $\underline{\mathbf{v_1}} := \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}(\underline{\mathbf{x}})$ and an arbitrary orthonormal basis $\{\underline{\mathbf{v_2}}, \dots, \underline{\mathbf{v_d}}\} := \Big\{\underline{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}^{\perp,\mathbf{x},\mathbf{i}}}\Big\}_{i=1,\dots,d-1}$ for the tangent space of $\mathcal{M}$ at $\underline{\mathbf{x}}$. Locally, the surface gradient of η thus can be represented as

$$\big[\nabla_{\mathcal{M}}\eta\big](\underline{\mathbf{x}}) = \sum_{i=1}^{d-1} \Big(\nabla \hat{\eta}(\underline{\mathbf{x}}) \cdot \underline{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}^{\perp,\mathbf{x},\mathbf{i}}}\Big)\, \underline{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}^{\perp,\mathbf{x},\mathbf{i}}}.$$

Note that each term $\nabla \hat{\eta}(\underline{\mathbf{x}}) \cdot \underline{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}^{\perp,\mathbf{x},\mathbf{i}}}$ is the directional derivative of $\hat{\eta}$ at $\underline{\mathbf{x}}$ in the tangential direction $\underline{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}^{\perp,\mathbf{x},\mathbf{i}}}$. This directional derivative only depends on values of $\hat{\eta}$ on $\mathcal{M}$, i.e. only on $\hat{\eta}\big|_{\mathcal{M}} = \eta$.

Surface partial derivatives of η (on $\mathcal{M}$) are given by the components of $\nabla_{\mathcal{M}}\eta$, as

$$\partial_i^{\mathcal{M}} \eta := \big(\nabla_{\mathcal{M}}\eta\big) \cdot \underline{\mathbf{e_i}}, \quad i = 1, \dots, d, \tag{2.5}$$

with $\underline{\mathbf{e_i}}$ denoting the i-th vector of the standard basis for $\mathbb{R}^d$, i.e. the unit vector

pointing in the direction of the i-th axis of the chosen Cartesian coordinate system.

Analogous to the classical divergence in $\mathbb{R}^d$, the *surface divergence* of a vector field can be described as the sum of its components' surface partial derivatives. More precisely, the surface divergence of $\boldsymbol{\xi}$ (on $\mathcal{M}$) is defined as

$$\operatorname{div}_{\mathcal{M}} \boldsymbol{\xi} := \sum_{i=1}^{d} \partial_i^{\mathcal{M}} \xi_i, \tag{2.6}$$

where we write $\boldsymbol{\xi} =: (\xi_1, \dots, \xi_d)^{\mathrm{tr}}$. Formally, by interpreting operator (2.3) as a vector of scalar operators $\left[\underline{\mathbf{e_i}} \cdot \nabla_{\mathcal{M}}\right]$, $i = 1, \dots, d$, the surface divergence of $\boldsymbol{\xi}$ can also be written in terms of $\nabla_{\mathcal{M}}$ directly:

$$\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} = \sum_{i=1}^{d} \left[\underline{\mathbf{e_i}} \cdot \nabla_{\mathcal{M}}\right] \xi_i = \sum_{i=1}^{d} \underline{\mathbf{e_i}} \cdot \nabla_{\mathcal{M}} \xi_i = \sum_{i=1}^{d} \partial_i^{\mathcal{M}} \xi_i = \operatorname{div}_{\mathcal{M}} \boldsymbol{\xi}. \tag{2.7}$$

As we will see next, it is reasonable that it is also called *tangential divergence* (on $\mathcal{M}$).

2.2. A closer look at surface divergence

Theorem 2.2.1 (Representation of $\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}$ in terms of the classical divergence in $\mathbb{R}^d$)**.** *The surface divergence of $\boldsymbol{\xi}$ can be represented as*

$$\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} = \nabla \cdot \hat{\boldsymbol{\xi}} - \left[D\hat{\boldsymbol{\xi}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\right] \cdot \boldsymbol{\nu}_{\mathcal{M}}, \tag{2.8}$$

where $\nabla \cdot \mathbf{f}$ and $D\mathbf{f}$ denote the classical divergence in $\mathbb{R}^d$ and the Jacobian matrix of a vector-valued function $\mathbf{f}$ which is differentiably defined in $\mathcal{N}(\mathcal{M})$, respectively. If $\boldsymbol{\nu}_{\mathcal{M}}$ has a differentiable extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$, this is equivalent to

$$\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} = \nabla \cdot \hat{\boldsymbol{\xi}} - \Big[\nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} - \underbrace{\big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big]}_{(*)} \cdot \boldsymbol{\xi}\Big], \tag{2.9}$$

where $(*)$ is the normal derivative of $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$, which vanishes if $\boldsymbol{\nu}_{\mathcal{M}}$ is extended such that $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ is constant along normal lines through $\mathcal{M}$.

Proof. Using equations (2.7), (2.6), (2.5) and the first equality in equation (2.4), we get

$$\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} = \sum_{i=1}^{d} \frac{\partial \hat{\xi}_i}{\partial x_i} - \big(\nabla \hat{\xi}_i \cdot \boldsymbol{\nu}_{\mathcal{M}}\big)\nu_{\mathcal{M},i} = \nabla \cdot \hat{\boldsymbol{\xi}} - \left[D\hat{\boldsymbol{\xi}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\right] \cdot \boldsymbol{\nu}_{\mathcal{M}}.$$

The second part of the theorem directly follows from the following remark. □

Remark 2.2.2. *Given two surface vector fields $\boldsymbol{\zeta_1}, \boldsymbol{\zeta_2} \colon \mathcal{M} \to \mathbb{R}^d$, with $\boldsymbol{\zeta_1}$ having a differentiable extension $\hat{\boldsymbol{\zeta}}_\mathbf{1}$ to $\mathcal{N}(\mathcal{M})$, the product rule yields*

$$\underbrace{\nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\zeta}}_\mathbf{1}\big] \cdot \boldsymbol{\zeta_2}}_{\substack{\text{directional derivative of} \\ \hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\zeta}}_\mathbf{1} \text{ in the direction } \boldsymbol{\zeta_2}}} = \underbrace{\big[D\hat{\boldsymbol{\xi}} \cdot \boldsymbol{\zeta_2}\big] \cdot \boldsymbol{\zeta_1}}_{\substack{\text{directional derivative of} \\ \hat{\boldsymbol{\xi}} \text{ in the direction } \boldsymbol{\zeta_2}, \\ \text{projected onto } \boldsymbol{\zeta_1}}} + \underbrace{\big[D\hat{\boldsymbol{\zeta}}_\mathbf{1} \cdot \boldsymbol{\zeta_2}\big] \cdot \boldsymbol{\xi}}_{\substack{\text{directional derivative of} \\ \hat{\boldsymbol{\zeta}}_\mathbf{1} \text{ in the direction } \boldsymbol{\zeta_2}, \\ \text{projected onto } \boldsymbol{\xi}}}.$$

By means of representation (2.8), one can see that $\nabla_\mathcal{M} \cdot \boldsymbol{\xi}$ can be computed from the classical divergence in $\mathbb{R}^d$ by subtracting the normal component of the extended field's derivative in normal direction, i.e., in a manner of speaking, the rate of change of $\hat{\boldsymbol{\xi}}$ regarding $\boldsymbol{\nu_\mathcal{M}}$. To further understand this connection, we write the classical divergence in $\mathbb{R}^d$ in terms of Cartesian coordinates:

$$\nabla \cdot \hat{\boldsymbol{\xi}} = \sum_{i=1}^{d} \frac{\partial \hat{\xi}_i}{\partial x_i} = \sum_{i=1}^{d} \big[D\hat{\boldsymbol{\xi}} \cdot \underline{\mathbf{e_i}}\big] \cdot \underline{\mathbf{e_i}}. \tag{2.10}$$

Equation (2.10) reveals that, in the same manner of speaking as above, $\nabla \cdot \hat{\boldsymbol{\xi}}$ describes the total rate of change of $\hat{\boldsymbol{\xi}}$ regarding the coordinate axes. This indicates that what is subtracted from $\nabla \cdot \hat{\boldsymbol{\xi}}$ in representation (2.8) is exactly the contribution of the rate of change of $\hat{\boldsymbol{\xi}}$ in normal direction, such that $\nabla_\mathcal{M} \cdot \boldsymbol{\xi}$ only takes into account changes of $\hat{\boldsymbol{\xi}}$ tangential to $\mathcal{M}$. To make this more obvious, we consider the following lemma.

Lemma 2.2.3. *Although being expressed in terms of coordinates, the classical divergence in $\mathbb{R}^d$ is invariant under orthogonal transformations. For every orthonormal basis $\big\{\underline{\mathbf{v_i}}\big\}_{i=1,\dots,d}$ for $\mathbb{R}^d$, it can be expressed as*

$$\nabla \cdot \hat{\boldsymbol{\xi}} = \sum_{i=1}^{d} \big[D\hat{\boldsymbol{\xi}} \cdot \underline{\mathbf{v_i}}\big] \cdot \underline{\mathbf{v_i}}.$$

Proof. Denoting the change-of-basis matrix from the basis $\big\{\underline{\mathbf{v_i}}\big\}_{i=1,\dots,d}$ to the basis $\big\{\underline{\mathbf{e_i}}\big\}_{i=1,\dots,d}$ by Q, we have

$$\big[D\hat{\boldsymbol{\xi}} \cdot \underline{\mathbf{v_i}}\big] \cdot \underline{\mathbf{v_i}} = \big[D\hat{\boldsymbol{\xi}} \cdot (Q\underline{\mathbf{e_i}})\big] \cdot (Q\underline{\mathbf{e_i}}) = \big[Q^{\mathrm{tr}} D\hat{\boldsymbol{\xi}} Q \cdot \underline{\mathbf{e_i}}\big] \cdot \underline{\mathbf{e_i}}.$$

Using that Q is an orthogonal matrix and that the trace $\mathrm{tr}(\cdot)$ of a matrix is similarity-invariant, we get

$$\sum_{i=1}^{d} \big[D\hat{\boldsymbol{\xi}} \cdot \underline{\mathbf{v_i}}\big] \cdot \underline{\mathbf{v_i}} = \mathrm{tr}\big(Q^{\mathrm{tr}} D\hat{\boldsymbol{\xi}} Q\big) = \mathrm{tr}\big(Q^{-1} D\hat{\boldsymbol{\xi}} Q\big) = \mathrm{tr}\big(D\hat{\boldsymbol{\xi}}\big) = \sum_{i=1}^{d} \big[D\hat{\boldsymbol{\xi}} \cdot \underline{\mathbf{e_i}}\big] \cdot \underline{\mathbf{e_i}}$$

which completes the proof considering equation (2.10). □

Locally choosing $\underline{\mathbf{v_1}} := \boldsymbol{\nu}_{\mathcal{M}}(\underline{\mathbf{x}})$ and an orthonormal basis $\{\underline{\mathbf{v_2}}, \dots, \underline{\mathbf{v_d}}\}$ for the tangent space of $\mathcal{M}$ at $\underline{\mathbf{x}} \in \mathcal{M}$ and using Lemma 2.2.3 to rewrite representation (2.8) yields the following corollary of Theorem 2.2.1.

Corollary 2.2.4 (Local representation of $\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}$ using the tangent spaces of $\mathcal{M}$)**.** *At each fixed $\underline{\mathbf{x}} \in \mathcal{M}$, the surface divergence of $\boldsymbol{\xi}$ can be expressed as*

$$\big[\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}\big](\underline{\mathbf{x}}) = \sum_{i=1}^{d-1} \Big[D\hat{\boldsymbol{\xi}}(\underline{\mathbf{x}}) \cdot \underline{\boldsymbol{\nu}_{\mathcal{M}}^{\perp,\underline{\mathbf{x}},\mathbf{i}}}\Big] \cdot \underline{\boldsymbol{\nu}_{\mathcal{M}}^{\perp,\underline{\mathbf{x}},\mathbf{i}}}, \tag{2.11}$$

where $\Big\{\underline{\boldsymbol{\nu}_{\mathcal{M}}^{\perp,\underline{\mathbf{x}},\mathbf{i}}}\Big\}_{i=1,\dots,d-1}$ denotes an arbitrary orthonormal basis for the tangent space of $\mathcal{M}$ at $\underline{\mathbf{x}}$.

From representation (2.11) it is clear that $\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}$ is independent of the choice of the extension $\hat{\boldsymbol{\xi}}$.[1] Each term $D\hat{\boldsymbol{\xi}}(\underline{\mathbf{x}}) \cdot \underline{\boldsymbol{\nu}_{\mathcal{M}}^{\perp,\underline{\mathbf{x}},\mathbf{i}}}$ is the directional derivative of $\hat{\boldsymbol{\xi}}$ at $\underline{\mathbf{x}}$ in the tangential direction $\underline{\boldsymbol{\nu}_{\mathcal{M}}^{\perp,\underline{\mathbf{x}},\mathbf{i}}}$. This directional derivative only depends on values of $\hat{\boldsymbol{\xi}}\big|_{\mathcal{M}} = \boldsymbol{\xi}$. Moreover, representation (2.11) shows that it is indeed reasonable to alternatively call $\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}$ the tangential divergence of $\boldsymbol{\xi}$ (on $\mathcal{M}$) since it only takes into account changes of $\hat{\boldsymbol{\xi}}$ tangential to $\mathcal{M}$.

2.2.1. Surface divergence of the tangential/normal component of a surface vector field and the notion of curvature

Let us now consider the tangential component $\boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}}$ of $\boldsymbol{\xi}$ and its normal component $\boldsymbol{\xi}_{\boldsymbol{\nu}_{\mathcal{M}}}$, given by

$$\boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}} := \mathcal{P}_{\mathcal{M}}\boldsymbol{\xi} = \boldsymbol{\xi} - (\boldsymbol{\xi}\cdot\boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}} \quad \text{and} \quad \boldsymbol{\xi}_{\boldsymbol{\nu}_{\mathcal{M}}} := \boldsymbol{\xi} - \boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}} = (\boldsymbol{\xi}\cdot\boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}}.$$

Provided that $\boldsymbol{\nu}_{\mathcal{M}}$ has a differentiable extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$, both components have straightforward differentiable extensions

$$\hat{\boldsymbol{\xi}}_{\mathbf{tan}_{\mathcal{M}}} := \hat{\boldsymbol{\xi}} - \big(\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big)\hat{\boldsymbol{\nu}}_{\mathcal{M}} \quad \text{and} \quad \hat{\boldsymbol{\xi}}_{\boldsymbol{\nu}_{\mathcal{M}}} := (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}$$

to $\mathcal{N}(\mathcal{M})$, respectively. It is hence reasonable to investigate the surface divergence of $\boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}}$ and $\boldsymbol{\xi}_{\boldsymbol{\nu}_{\mathcal{M}}}$ in this case, which is what we will do next.

We begin by considering the normal component $\boldsymbol{\xi}_{\boldsymbol{\nu}_{\mathcal{M}}}$. It should be noted that even though $\boldsymbol{\xi}_{\boldsymbol{\nu}_{\mathcal{M}}}$ points in normal direction (provided that $\boldsymbol{\xi}_{\boldsymbol{\nu}_{\mathcal{M}}} \neq 0$), it may exhibit changes tangential to $\mathcal{M}$. This is exactly the case if $\mathcal{M}$ is curved, as the following lemma will show.

[1] Note that the surface divergence inherits this property from the surface gradient. See also equations (2.7), (2.6), (2.5) and Remark 2.1.1.

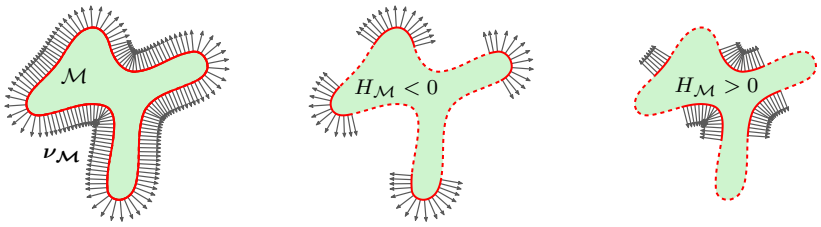


Figure 2.1.: The left picture illustrates a hypersurface $\mathcal{M}$ (red), its inside (light green) and the surface field $\boldsymbol{\nu}_{\mathcal{M}}$ (gray) providing its orientation. The pictures in the middle and on the right show regions $M \subset \mathcal{M}$ of negative/positive total curvature $H_{\mathcal{M}}$ as continuous red curves that are supplemented with corresponding restrictions $\boldsymbol{\nu}_{\mathcal{M}}|_M$.

Lemma 2.2.5 (Representation of the surface divergence of the normal component)**.** *If $\boldsymbol{\nu}_{\mathcal{M}}$ has a differentiable extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$, the surface divergence of normal component $\boldsymbol{\xi}_{\nu_{\mathcal{M}}} := (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}}$ of $\boldsymbol{\xi}$ can be represented as*

$$\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}_{\nu_{\mathcal{M}}} = \underbrace{(\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big)}_{\text{main term}} - \underbrace{\big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}}}_{\substack{\text{correction term accounting for} \\ \text{the influence of the extension} \\ \hat{\boldsymbol{\nu}}_{\mathcal{M}} \text{ of } \boldsymbol{\nu}_{\mathcal{M}} \text{ in the main term}}}. \tag{2.12}$$

By using representation (2.12) and applying representation (2.8) for $\boldsymbol{\xi} = \boldsymbol{\nu}_{\mathcal{M}}$ we deduce the following variant given by

$$\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}_{\nu_{\mathcal{M}}} = (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big) = -(\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})H_{\mathcal{M}}, \tag{2.13}$$

where the expression

$$H_{\mathcal{M}} := -\nabla_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}} = -\Big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}} - \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}}\Big) \tag{2.14}$$

is a measure for the curvature of $\mathcal{M}$.

Definition 2.2.6 (Total curvature)**.** The surface field $H_{\mathcal{M}} \colon \mathcal{M} \to \mathbb{R}$ defined in equation (2.14) is known as the *total curvature* of $\mathcal{M}$ (Cermelli et al., 2005). It is reasonable if $\boldsymbol{\nu}_{\mathcal{M}}$ has a differentiable extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ and describes $d-1$ times the mean curvature of $\mathcal{M}$ in this case. The field $H_{\mathcal{M}}$ itself is also called mean curvature of $\mathcal{M}$ by some authors. Furthermore, its definition sometimes differs from our definition in equation (2.14) by a factor of -1, depending on the choice of the field $\boldsymbol{\nu}_{\mathcal{M}}$ which provides the orientation of $\mathcal{M}$. See Figure 2.1 for illustrations that are consistent with our definition.

In view of representation (2.13), the normal component $\boldsymbol{\xi}_{\nu_{\mathcal{M}}}$ of $\boldsymbol{\xi}$ thus indeed exhibits changes tangential to $\mathcal{M}$ if and only if $\boldsymbol{\xi}_{\nu_{\mathcal{M}}} \neq 0$ and $\mathcal{M}$ is curved. These changes are characterized by the length of $\boldsymbol{\xi}_{\nu_{\mathcal{M}}}$, which is given by $\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}}$, and the total curvature $H_{\mathcal{M}}$ of $\mathcal{M}$.

We continue with the tangential component $\boldsymbol{\xi}_{\mathbf{tan}_\mathcal{M}}$ of $\boldsymbol{\xi}$ and the particular extension $\hat{\boldsymbol{\xi}}_{\mathbf{tan}_\mathcal{M}}$ given above, where we assume again that $\boldsymbol{\nu}_\mathcal{M}$ has a differentiable extension $\hat{\boldsymbol{\nu}}_\mathcal{M}$. If $\hat{\boldsymbol{\nu}}_\mathcal{M}$ is constant along normal lines through $\mathcal{M}$, given representation (2.9), it can be expected that the surface divergence of $\boldsymbol{\xi}_{\mathbf{tan}_\mathcal{M}}$ and the classical divergence of $\hat{\boldsymbol{\xi}}_{\mathbf{tan}_\mathcal{M}}$ in $\mathbb{R}^d$ coincide. In fact, $\hat{\boldsymbol{\xi}}_{\mathbf{tan}_\mathcal{M}} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M}$ does not exhibit changes in normal direction in this case since

$$
\begin{aligned}
& |\hat{\boldsymbol{\nu}}_\mathcal{M}| = 1 \text{ on } \mathcal{M} \quad \text{and} \quad D\hat{\boldsymbol{\nu}}_\mathcal{M} \cdot \boldsymbol{\nu}_\mathcal{M} = 0 \text{ on } \mathcal{M} \\
\Rightarrow\ & |\hat{\boldsymbol{\nu}}_\mathcal{M}| = 1 \text{ in } \mathcal{N}(\mathcal{M}) \\
\Rightarrow\ & \hat{\boldsymbol{\xi}}_{\mathbf{tan}_\mathcal{M}} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M} = 0 \text{ in } \mathcal{N}(\mathcal{M}) \\
\Rightarrow\ & \nabla\big[\hat{\boldsymbol{\xi}}_{\mathbf{tan}_\mathcal{M}} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M}\big] \cdot \boldsymbol{\nu}_\mathcal{M} = 0 \text{ on } \mathcal{M}.
\end{aligned}
$$

Dropping the assumption on $\hat{\boldsymbol{\nu}}_\mathcal{M}$, it can be shown that the surface divergence $\nabla_\mathcal{M} \cdot \boldsymbol{\xi}_{\mathbf{tan}_\mathcal{M}}$ matches $\nabla \cdot \hat{\boldsymbol{\xi}}_{\mathbf{tan}_\mathcal{M}}$ up to a term which describes the influence of the extension $\hat{\boldsymbol{\nu}}_\mathcal{M}$. This is the subject of the following lemma.

Lemma 2.2.7 (Representation of the surface divergence of the tangential component)**.** *If $\boldsymbol{\nu}_\mathcal{M}$ has a differentiable extension $\hat{\boldsymbol{\nu}}_\mathcal{M}$, the surface divergence of tangential component $\boldsymbol{\xi}_{\mathbf{tan}_\mathcal{M}} := \boldsymbol{\xi} - (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_\mathcal{M})\boldsymbol{\nu}_\mathcal{M}$ of $\boldsymbol{\xi}$ can be represented as*

$$
\begin{aligned}
&\nabla_\mathcal{M} \cdot \boldsymbol{\xi}_{\mathbf{tan}_\mathcal{M}} \\
&\quad = \underbrace{\nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M})\hat{\boldsymbol{\nu}}_\mathcal{M}\Big]}_{\textit{main term}} \quad + \quad \underbrace{\big[D\hat{\boldsymbol{\nu}}_\mathcal{M} \cdot \boldsymbol{\nu}_\mathcal{M}\big] \cdot \big[\boldsymbol{\xi} + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_\mathcal{M})\boldsymbol{\nu}_\mathcal{M}\big]}_{\substack{\textit{correction term accounting for} \\ \textit{the influence of the extension} \\ \hat{\boldsymbol{\nu}}_\mathcal{M} \textit{ of } \boldsymbol{\nu}_\mathcal{M} \textit{ in the main term}}}.
\end{aligned}
\tag{2.15}
$$

Since we did not find Lemma 2.2.5 and Lemma 2.2.7 in the literature, we will provide a proof for them now.

Proof of Lemma 2.2.5 and Lemma 2.2.7. The proof given here proceeds in three steps, with steps 2) and 3) being the actual proofs of the lemmata.

1) If $\boldsymbol{\nu}_\mathcal{M}$ has a differentiable extension $\hat{\boldsymbol{\nu}}_\mathcal{M}$, the product rule, the equality $|\boldsymbol{\nu}_\mathcal{M}| = 1$ on $\mathcal{M}$ and Remark 2.2.2 yield

$$
\begin{aligned}
&\nabla\Big[(\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M})\hat{\boldsymbol{\nu}}_\mathcal{M} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M}\Big] \cdot \boldsymbol{\nu}_\mathcal{M} \\
&\qquad = \nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M}\big](\boldsymbol{\nu}_\mathcal{M} \cdot \boldsymbol{\nu}_\mathcal{M}) \cdot \boldsymbol{\nu}_\mathcal{M} + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_\mathcal{M})\nabla\big[\hat{\boldsymbol{\nu}}_\mathcal{M} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M}\big] \cdot \boldsymbol{\nu}_\mathcal{M} \\
&\qquad = \nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M}\big] \cdot \boldsymbol{\nu}_\mathcal{M} + 2\big[D\hat{\boldsymbol{\nu}}_\mathcal{M} \cdot \boldsymbol{\nu}_\mathcal{M}\big] \cdot (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_\mathcal{M})\boldsymbol{\nu}_\mathcal{M}.
\end{aligned}
$$

For the term

$$
\text{(I)} := \nabla\Big[(\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M})\hat{\boldsymbol{\nu}}_\mathcal{M} \cdot \hat{\boldsymbol{\nu}}_\mathcal{M}\Big] \cdot \boldsymbol{\nu}_\mathcal{M} - \big[D\hat{\boldsymbol{\nu}}_\mathcal{M} \cdot \boldsymbol{\nu}_\mathcal{M}\big] \cdot (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_\mathcal{M})\boldsymbol{\nu}_\mathcal{M},
$$

we hence have the equality

$$(\mathrm{I}) = \nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} + \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}}.$$

2) Using representation (2.9), the product rule and step 1), we get

$$\begin{aligned}
&\nabla_{\mathcal{M}} \cdot \big[(\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}}\big] \\
&\quad = \nabla \cdot \Big[(\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] - (\mathrm{I}) \\
&\quad = \nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big) - (\mathrm{I}) \\
&\quad = \nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big) \\
&\qquad\quad - \Big[\nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} + \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}}\Big] \\
&\quad = (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big) - \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}},
\end{aligned}$$

which is representation (2.12).

3) Using the same arguments as in step 2), furthermore

$$\begin{aligned}
&\nabla_{\mathcal{M}} \cdot \big[\boldsymbol{\xi} - (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}}\big] \\
&\quad = \nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] \\
&\qquad\quad - \Big[\nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} - \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\xi} - (\mathrm{I})\Big] \\
&\quad = \nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] \\
&\qquad\quad - \nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} + \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\xi} \\
&\qquad\quad + \nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} + \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}} \\
&\quad = \nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] + \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \big[\boldsymbol{\xi} + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}}\big],
\end{aligned}$$

which is representation (2.15). □

2.2.2. Splitted representation of surface divergence

Representations (2.12) and (2.15) together yield another interesting representation which decomposes $\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}$ into essential constituents with respect to $\boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}}$, $\boldsymbol{\xi}_{\boldsymbol{\nu}_{\mathcal{M}}}$ and the extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$. This is stated in the following theorem.

Theorem 2.2.8 (Splitted representation of $\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}$)**.** *Let $\boldsymbol{\nu}_{\mathcal{M}}$ have a differentiable extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$. Then the surface divergence of $\boldsymbol{\xi}$ decomposes into*

$$\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} \;=\; \underbrace{\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}}}_{\substack{\text{contribution} \\ \text{of } \boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}}}} \;+\; \underbrace{\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}_{\boldsymbol{\nu}_{\mathcal{M}}}}_{\substack{\text{contribution} \\ \text{of } \boldsymbol{\xi}_{\boldsymbol{\nu}_{\mathcal{M}}}}},$$

which can be represented as

$$\begin{aligned}\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} \\ = \underbrace{\nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big]}_{\substack{\textit{main term of}\\ \textit{contribution of}\\ \boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}}}} + \underbrace{\big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\xi}}_{\substack{\textit{correction term accounting}\\ \textit{for the cumulative influence}\\ \textit{of the extension } \hat{\boldsymbol{\nu}}_{\mathcal{M}} \textit{ of } \boldsymbol{\nu}_{\mathcal{M}}\\ \textit{in the main terms of}\\ \textit{the contributions of}\\ \boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}} \textit{ and } \boldsymbol{\xi}_{\nu_{\mathcal{M}}}}} + \underbrace{(\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big)}_{\substack{\textit{main term of}\\ \textit{contribution of}\\ \boldsymbol{\xi}_{\nu_{\mathcal{M}}}}}.\end{aligned} \tag{2.16}$$

Proof. For surface vector fields $\boldsymbol{\xi_1}, \boldsymbol{\xi_2} \colon \mathcal{M} \to \mathbb{R}^d$ with the same extension property as $\boldsymbol{\xi}$, representation (2.8) immediately yields

$$\nabla_{\mathcal{M}} \cdot \big(\boldsymbol{\xi_1} + \boldsymbol{\xi_2}\big) = \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi_1} + \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi_2}.$$

Using that $\boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}}$ and $\boldsymbol{\xi}_{\nu_{\mathcal{M}}}$ have suitable differentiable extensions, we can employ this additivity to derive the first statement of Theorem 2.2.8. Together with representations (2.15) and (2.12), we subsequently get

$$\begin{aligned}\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} &= \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}} + \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}_{\nu_{\mathcal{M}}} \\ &= \nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] + \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \big[\boldsymbol{\xi} + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}}\big] \\ &\quad + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big) - \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\boldsymbol{\nu}_{\mathcal{M}} \\ &= \nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] + \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\xi} + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big).\end{aligned}$$

Alternatively, representation (2.16) can also be obtained in a straightforward way without employing representations (2.15) and (2.12). Note that representation (2.8), the linearity of the classical divergence operator in $\mathbb{R}^d$ and the product rule yield

$$\begin{aligned}\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} &= \nabla \cdot \hat{\boldsymbol{\xi}} - \big[D\hat{\boldsymbol{\xi}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} \\ &= \nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] + \nabla \cdot \Big[(\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] - \big[D\hat{\boldsymbol{\xi}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} \\ &= \nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] \\ &\quad + \nabla\big[\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big) - \big[D\hat{\boldsymbol{\xi}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} \\ &= \nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}})\hat{\boldsymbol{\nu}}_{\mathcal{M}}\Big] + \big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\xi} + (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}\big),\end{aligned}$$

where we have furthermore applied Remark 2.2.2 in the last step. But this derivation is not as constructive as the derivation given above since it does not expose how $\boldsymbol{\xi}_{\mathbf{tan}_{\mathcal{M}}}$ and $\boldsymbol{\xi}_{\nu_{\mathcal{M}}}$ contribute to the term $\big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\xi}$. □

2.2.3. Additional remarks

The influence of the extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ in representations (2.9), (2.12), (2.14), (2.15) and (2.16) vanishes if $\boldsymbol{\nu}_{\mathcal{M}}$ is extended such that the normal derivative of $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ vanishes, i.e., if $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ is constant along normal lines through $\mathcal{M}$. In the next section, we will investigate related properties for a particular extension of $\boldsymbol{\nu}_{\mathcal{M}}$ which is the canonical one in a framework for geometry description that will be used in this thesis.

2.3. Surface divergence in the level set framework

In this section, let furthermore exist a twice differentiable scalar field

$$\Phi\colon \mathcal{N}(\mathcal{M}) \to \mathbb{R} \qquad \text{with} \qquad |\nabla\Phi| \neq 0 \quad \text{in} \quad \mathcal{N}(\mathcal{M})$$

which has $\mathcal{M}$ as one of its level sets and can be used to represent $\boldsymbol{\nu}_{\mathcal{M}}$ as

$$\boldsymbol{\nu}_{\mathcal{M}} = \frac{\nabla\Phi}{|\nabla\Phi|}\bigg|_{\mathcal{M}}.$$

Let $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ be the canonical extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}} = \frac{\nabla\Phi}{|\nabla\Phi|}$ of $\boldsymbol{\nu}_{\mathcal{M}}$ in this framework which is known as the *level set framework* (see Section 3.3).

Investigating the canonical extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ in the level set framework reveals an interesting property of its normal derivative. This is the subject of the following lemma.

Lemma 2.3.1 (Normal derivative of the canonical extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$)**.** *Assume that Φ has symmetric second derivatives, e.g. $\Phi \in C^2\big(\mathcal{N}(\mathcal{M})\big)$. Then the normal derivative of the canonical extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ of $\boldsymbol{\nu}_{\mathcal{M}}$ has the property*

$$D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}} = \mathcal{P}_{\mathcal{M}} \frac{\nabla|\nabla\Phi|}{|\nabla\Phi|} \tag{2.17a}$$

and thus

$$\big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\xi} = \frac{\nabla|\nabla\Phi|}{|\nabla\Phi|} \cdot \mathcal{P}_{\mathcal{M}}\boldsymbol{\xi}. \tag{2.17b}$$

Property (2.17) shows that, provided that Φ has symmetric second derivatives, $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$ is constant along normal lines through $\mathcal{M}$ if and only if $\nabla|\nabla\Phi|$ points in normal direction on $\mathcal{M}$ or $|\nabla\Phi|$ is constant in $\mathcal{N}(\mathcal{M})$. The latter particularly holds true if Φ is a signed distance function (see Section 3.3.1). In addition, the term $\big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\xi}$ vanishes for surface vector fields with $\boldsymbol{\xi} = \boldsymbol{\xi}_{\nu_{\mathcal{M}}}$. Therefore, representation (2.14) of the total curvature of $\mathcal{M}$ simplifies to $H_{\mathcal{M}} = -\nabla \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}}$ and, regarding equation (2.16), the correction term $\big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big] \cdot \boldsymbol{\xi}$ only accounts for the influence of the canonical extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$

in the main term of the contribution of $\boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}}$. In the level set framework with a Φ that has symmetric second derivatives, representation (2.16) thus takes the form

$$\begin{aligned}&\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}\\ &\quad= \underbrace{\nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}})\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\Big]}_{\substack{\text{main term of}\\ \text{contribution of}\\ \boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}}}} + \underbrace{\big[D\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}\big] \cdot \boldsymbol{\xi}}_{\substack{\text{correction term accounting}\\ \text{for the influence of the}\\ \text{canonical extension } \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\\ \text{in the main term of the}\\ \text{contribution of } \boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}}}} + \underbrace{(\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)}_{\substack{\text{contribution of}\\ \boldsymbol{\xi}_{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}}}}.\end{aligned}$$

Furthermore, the first two terms on the right-hand side can be combined. Using (2.17) and the product rule, we get

$$\begin{aligned}&\nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}})\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\Big] + \big[D\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}\big] \cdot \boldsymbol{\xi}\\ &\qquad= \nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}})\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\Big] + \frac{\nabla|\nabla\Phi|}{|\nabla\Phi|} \cdot \big[\boldsymbol{\xi} - (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}\big]\\ &\qquad= \frac{1}{|\nabla\Phi|}\,\nabla \cdot \Big(|\nabla\Phi|\Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}})\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\Big]\Big).\end{aligned}$$

Summarizing these implications, we arrive at the following two theorems.

Theorem 2.3.2 (Splitted representation of $\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}$ in the level set framework)**.** *Assume that Φ has symmetric second derivatives, e.g. $\Phi \in C^2\big(\mathcal{N}(\mathcal{M})\big)$. Then*

$$\begin{aligned}&\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}\\ &\quad= \underbrace{\frac{1}{|\nabla\Phi|}\,\nabla \cdot \Big(|\nabla\Phi|\Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}})\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\Big]\Big)}_{\substack{\textit{contribution of } \boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}} \textit{ (main term combined}\\ \textit{with correction term accounting for the}\\ \textit{influence of the canonical extension } \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}})}} + \underbrace{(\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)}_{\substack{\textit{contribution of}\\ \boldsymbol{\xi}_{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}}}}. \qquad (2.18)\end{aligned}$$

If $\nabla|\nabla\Phi|$ points in normal direction on $\mathcal{M}$ or $|\nabla\Phi|$ is constant in $\mathcal{N}(\mathcal{M})$, this indentity simplifies to

$$\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} = \underbrace{\nabla \cdot \Big[\hat{\boldsymbol{\xi}} - (\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}})\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\Big]}_{\textit{contribution of } \boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}}} + \underbrace{(\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})\big(\nabla \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)}_{\textit{contribution of } \boldsymbol{\xi}_{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}}}. \qquad (2.19)$$

Theorem 2.3.3 (Representation of the total curvature $H_{\mathcal{M}}$ in the level set framework)**.** *Under the assumption from the first part of Theorem 2.3.2, the total curvature $H_{\mathcal{M}} := -\nabla_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}$ of $\mathcal{M}$ can be represented as*

$$H_{\mathcal{M}} = -\nabla \cdot \hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}.$$

In the remainder of this section, we will provide a proof for Lemma 2.3.1 since we did not find Lemma 2.3.1 and some of its implications in the literature, particularly not Theorem 2.3.2. We would like to point out, however, that for a tangential surface vector field $\boldsymbol{\xi}$ and a particular extension $\hat{\boldsymbol{\xi}}$ which is tangential to each level set (such that $\hat{\boldsymbol{\xi}} \cdot \hat{\boldsymbol{\nu}}_{\mathcal{M}} \equiv 0$), a result similar to the special case

$$\nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} = |\nabla\Phi|^{-1} \, \nabla \cdot \big(|\nabla\Phi| \hat{\boldsymbol{\xi}}\big)$$

of representation (2.18) is presented in Dziuk and Elliott (2008, Remark 3.3). Using the same requirements with respect to $\hat{\boldsymbol{\xi}}$, this result is furthermore presented and proved in Deckelnick et al. (2010, Lemma 2.1). The proof given therein makes tacit use of the assumption from the first part of Theorem 2.3.2. Representation (2.18) thus can be considered as a true generalization.

Remark 2.3.4 (Derivatives of $|\nabla\Phi|$)**.** *For $i = 1, \dots, d$, the chain rule yields*

$$\begin{aligned}\partial_i |\nabla\Phi| &= \partial_i \left[\Big(\sum\nolimits_{k=1}^{d} \big(\partial_k \Phi\big)^2 \Big)^{1/2} \right] \\ &= |\nabla\Phi|^{-1} \sum\nolimits_{k=1}^{d} \partial_k\Phi \, \partial_i \partial_k \Phi = \frac{\nabla\Phi}{|\nabla\Phi|} \cdot \partial_i \big[\nabla\Phi\big] = \partial_i \big[\nabla\Phi\big] \cdot \boldsymbol{\nu}_{\mathcal{M}}.\end{aligned}$$

If Φ has symmetric second derivatives, moreover $\partial_i |\nabla\Phi| = \nabla\big[\partial_i\Phi\big] \cdot \boldsymbol{\nu}_{\mathcal{M}}$ for $i = 1, \dots, d$.

Proof of Lemma 2.3.1. Using the quotient rule, we get

$$\begin{aligned}\big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big]_i &= \nabla \hat{\nu}_{\mathcal{M},i} \cdot \boldsymbol{\nu}_{\mathcal{M}} = \nabla \left[\frac{\partial_i \Phi}{|\nabla\Phi|} \right] \cdot \boldsymbol{\nu}_{\mathcal{M}} \\ &= |\nabla\Phi|^{-2} \Big(\nabla\big[\partial_i\Phi\big] \, |\nabla\Phi| - \partial_i\Phi \, \nabla|\nabla\Phi| \Big) \cdot \boldsymbol{\nu}_{\mathcal{M}} \\ &= |\nabla\Phi|^{-1} \Big(\nabla\big[\partial_i\Phi\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} \Big) - \frac{\partial_i\Phi}{|\nabla\Phi|} \frac{\nabla|\nabla\Phi|}{|\nabla\Phi|} \cdot \boldsymbol{\nu}_{\mathcal{M}} \\ &= |\nabla\Phi|^{-1} \Big(\nabla\big[\partial_i\Phi\big] \cdot \boldsymbol{\nu}_{\mathcal{M}} \Big) - \left(\frac{\nabla|\nabla\Phi|}{|\nabla\Phi|} \cdot \boldsymbol{\nu}_{\mathcal{M}} \right) \nu_{\mathcal{M},i}\end{aligned}$$

for $i = 1, \dots, d$. For each i, the second part of Remark 2.3.4 subsequently yields

$$\begin{aligned}\big[D\hat{\boldsymbol{\nu}}_{\mathcal{M}} \cdot \boldsymbol{\nu}_{\mathcal{M}}\big]_i &= \frac{\partial_i |\nabla\Phi|}{|\nabla\Phi|} - \left(\frac{\nabla|\nabla\Phi|}{|\nabla\Phi|} \cdot \boldsymbol{\nu}_{\mathcal{M}} \right) \nu_{\mathcal{M},i} \\ &= \left[\frac{\nabla|\nabla\Phi|}{|\nabla\Phi|} - \left(\frac{\nabla|\nabla\Phi|}{|\nabla\Phi|} \cdot \boldsymbol{\nu}_{\mathcal{M}} \right) \boldsymbol{\nu}_{\mathcal{M}} \right]_i = \left[\mathcal{P}_{\mathcal{M}} \frac{\nabla|\nabla\Phi|}{|\nabla\Phi|} \right]_i\end{aligned}$$

which proves the first statement of Lemma 2.3.1. The second statement of Lemma 2.3.1 directly follows from the self-adjointness of $\mathcal{P}_{\mathcal{M}}$. □

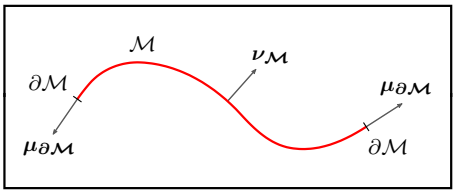


Figure 2.2.: An open hypersurface $\mathcal{M}$ which is oriented by a field $\boldsymbol{\nu}_{\mathcal{M}}$ of unit normal vectors to $\mathcal{M}$, and the field $\boldsymbol{\mu}_{\partial\mathcal{M}}$ of intrinsic outward-pointing unit normal vectors to the boundary $\partial\mathcal{M}$.

2.4. Integral calculus on hypersurfaces

Next, we revert back to the general setting without the additional assumptions from Section 2.3 (those which are related to the level set framework), and briefly recall theory from integral calculus on hypersurfaces. Following Ecker (2004, Appendix A), compactly supported surface fields can be integrated over properly embedded hypersurfaces. Here, a $(d-1)$-dimensional hypersurface embedded in $\mathbb{R}^d$, like $\mathcal{M}$, is called properly embedded if the embedding map $F\colon \mathcal{D} \to \mathcal{M}$, $\mathcal{D} \subset \mathbb{R}^{d-1}$ open, has the property that $F^{-1}(\mathcal{M} \cap K) \subset \mathcal{D}$ is compact whenever $K \subset \mathbb{R}^d$ is compact. Throughout this thesis, we will simply assume that $\mathcal{M}$ is given such that integrals over $\mathcal{M}$ are reasonable wherever this is necessary.

A fundamental integration formula for hypersurfaces that we will need is the divergence theorem for hypersurfaces. Analogous to the classical divergence theorem for bulk domains in $\mathbb{R}^d$, it relates the integral of the surface divergence of a surface vector field over a hypersurface to the outward flow of the field through its boundary. In contrast to the classical divergence theorem, the surface vector field may point away from the hypersurface and hence exhibit a component normal to the hypersurface. As with the splitted representation of surface divergence from Theorem 2.2.8, this normal component yields an additional contribution to the integral of the field's surface divergence. This is stated in the following theorem.

Theorem 2.4.1 (Divergence theorem for hypersurfaces (see also Ecker, 2004, Appendix A))**.** *Let the boundary $\partial\mathcal{M}$ of $\mathcal{M}$ be either empty or a smooth $(d-2)$-dimensional manifold, and let $\boldsymbol{\xi} \in C^1\big(\mathcal{M} \cup \partial\mathcal{M}; \mathbb{R}^d\big)$ be a continuously differentiable surface vector field which is not necessarily tangential to $\mathcal{M}$ and has the usual extension property defined at the beginning of this chapter. Furthermore, let $\boldsymbol{\nu}_{\mathcal{M}}$ have a differentiable extension $\hat{\boldsymbol{\nu}}_{\mathcal{M}}$. Then the identity*

$$\int_{\mathcal{M}} \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} \, \mathrm{d}\sigma = \int_{\partial\mathcal{M}} \boldsymbol{\xi} \cdot \boldsymbol{\mu}_{\partial\mathcal{M}} \, \mathrm{d}\varsigma - \int_{\mathcal{M}} (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\mathcal{M}}) H_{\mathcal{M}} \, \mathrm{d}\sigma \tag{2.20}$$

holds, where $\boldsymbol{\mu}_{\partial\mathcal{M}}$ is the field of intrinsic outward-pointing unit normal vectors to $\partial\mathcal{M}$. Here, the term intrinsic refers to the fact that those vectors are normal

to $\partial\mathcal{M}$ and tangential to $\mathcal{M}$ (also known as conormal vectors to $\partial\mathcal{M}$). See Figure 2.2 for an illustration. If $\boldsymbol{\xi}$ has compact support or if $\partial\mathcal{M} = \emptyset$, identity (2.20) *reduces to*

$$\int_{\mathcal{M}} \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} \, \mathrm{d}\sigma = -\int_{\mathcal{M}} (\boldsymbol{\xi} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}) H_{\mathcal{M}} \, \mathrm{d}\sigma.$$

Proof. As with the divergence theorem for bulk domains, the identity

$$\int_{\mathcal{M}} \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} \, \mathrm{d}\sigma = \int_{\partial\mathcal{M}} \boldsymbol{\xi} \cdot \boldsymbol{\mu}_{\boldsymbol{\partial\mathcal{M}}} \, \mathrm{d}\varsigma$$

holds if $\boldsymbol{\xi}$ is a surface vector field tangential to $\mathcal{M}$ (see e.g. Burstall, 1999, Section 2.3.3). Splitting $\boldsymbol{\xi}$ into its tangential component $\boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}}$ and its normal component $\boldsymbol{\xi}_{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}}$ using the first statement of Theorem 2.2.8 subsequently yields

$$\begin{aligned}\int_{\mathcal{M}} \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi} \, \mathrm{d}\sigma &= \int_{\mathcal{M}} \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}} \, \mathrm{d}\sigma \quad + \int_{\mathcal{M}} \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}_{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}} \, \mathrm{d}\sigma \\ &= \int_{\partial\mathcal{M}} \boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}} \cdot \boldsymbol{\mu}_{\boldsymbol{\partial\mathcal{M}}} \, \mathrm{d}\varsigma + \int_{\mathcal{M}} \nabla_{\mathcal{M}} \cdot \boldsymbol{\xi}_{\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}} \, \mathrm{d}\sigma.\end{aligned}$$

We conclude the proof by employing equation (2.13) to rewrite the second integral on the right-hand side and noticing that $\boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}} \cdot \boldsymbol{\mu}_{\boldsymbol{\partial\mathcal{M}}} = \boldsymbol{\xi} \cdot \boldsymbol{\mu}_{\boldsymbol{\partial\mathcal{M}}}$ since $\boldsymbol{\xi}_{\mathbf{tan}_{\boldsymbol{\mathcal{M}}}} = \mathcal{P}_{\mathcal{M}}\boldsymbol{\xi}$, $\mathcal{P}_{\mathcal{M}}$ is self-adjoint and $\boldsymbol{\mu}_{\boldsymbol{\partial\mathcal{M}}}$ is tangential to $\mathcal{M}$, such that $\mathcal{P}_{\mathcal{M}}\boldsymbol{\mu}_{\boldsymbol{\partial\mathcal{M}}} = \boldsymbol{\mu}_{\boldsymbol{\partial\mathcal{M}}}$. □

From the divergence theorem for hypersurfaces, we obtain a formula for integration by parts on hypersurfaces.

Corollary 2.4.2 (Integration by parts formula on hypersurfaces)**.** *Let the boundary $\partial\mathcal{M}$ of $\mathcal{M}$ be either empty or a smooth $(d-2)$-dimensional manifold, and let $\eta \in C^1(\mathcal{M} \cup \partial\mathcal{M})$ be a continuously differentiable surface scalar field which has the usual extension property defined at the beginning of this chapter. Furthermore, let $\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}$ have a differentiable extension $\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}$. Then the identity*

$$\int_{\mathcal{M}} \partial_i^{\mathcal{M}} \eta \, \mathrm{d}\sigma = \int_{\partial\mathcal{M}} \eta \, \mu_{\partial\mathcal{M},i} \, \mathrm{d}\varsigma - \int_{\mathcal{M}} \eta \, \nu_{\mathcal{M},i} \, H_{\mathcal{M}} \, \mathrm{d}\sigma$$

holds for $i = 1, \ldots, d$, where $\mu_{\partial\mathcal{M},i}$ is the i-th component of the field of intrinsic outward-pointing unit normal vectors to $\partial\mathcal{M}$. If η has compact support or if $\partial\mathcal{M} = \emptyset$, the boundary term on the right-hand side of the above formula vanishes.

Proof. We apply Theorem 2.4.1 for $\boldsymbol{\xi} = \eta \, \underline{\mathbf{e_i}}$. An alternative proof which employs arguments from the level set framework can be found in Dziuk and Elliott (2013, Section 2.3), where the definition of $H_{\mathcal{M}}$ differs from our definition in equation (2.14) by a factor of -1. □

Remark 2.4.3 (Equivalence of both formulas)**.** *Although being presented as a consequence of the divergence theorem for hypersurfaces, Corollary 2.4.2 is in fact equivalent to Theorem 2.4.1. To see this, we assume that a surface vector field $\boldsymbol{\xi}$ is given as requested in the divergence theorem for hypersurfaces, apply the corollary for $\eta = \xi_i$, and use equations* (2.7) *and* (2.6) *to rewrite the left-hand side of identity* (2.20)*.*

2.5. Integration of those concepts into the time-dependent case

In the time-dependent case, two main situations have to be considered. On the one hand, we can have time-dependent fields which we want to differentiate or integrate on a static hypersurface. On the other hand, the fields can live on a hypersurface which evolves in time and hence is time-dependent itself. Next, we will integrate the concepts from Sections 2.1–2.4 into each of these two situations.

2.5.1. Time-dependent fields on static hypersurfaces

In case of a static hypersurface and time-dependent fields, we are interested in a hypersurface $\mathcal{M}$ that is oriented by a static surface field $\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}$ of unit normal vectors to $\mathcal{M}$, both defined exactly as in the beginning of this chapter. Furthermore, we are interested in surface differential operators acting on time-dependent surface fields $\eta(\cdot,t)\colon \mathcal{M} \to \mathbb{R}$ and $\boldsymbol{\xi}(\cdot,t)\colon \mathcal{M} \to \mathbb{R}^d$ that are observed during a specified time period $[0,T]$. Since, so far, only surface fields have been considered that are independent of time, this setting is different from the one in the previous sections.

Nevertheless, requiring that η and $\boldsymbol{\xi}$ have spatially differentiable extensions $\hat{\eta}(\cdot,t)\colon \mathcal{N}(\mathcal{M}) \to \mathbb{R}$ and $\hat{\boldsymbol{\xi}}(\cdot,t)\colon \mathcal{N}(\mathcal{M}) \to \mathbb{R}^d$ to a neighborhood $\mathcal{N}(\mathcal{M}) \subset \mathbb{R}^d$ of $\mathcal{M}$ with $\operatorname{meas}_{\mathbb{R}^d}\big(\mathcal{N}(\mathcal{M})\big) > 0$, all our definitions, theorems and statements from Sections 2.1–2.4 trivially carry over since they only act on the spatial domain of the surface fields. They can simply be applied pointwise with respect to time.

For instance, without changing the definition of the projection operator $\mathcal{P}_{\mathcal{M}}$ in equation (2.1), we can define a time-dependent surface gradient of η on $\mathcal{M}$. It is given by

$$\begin{aligned}\big[\operatorname{grad}_{\mathcal{M}}\eta\big](\cdot,t) &:= \Big(\mathcal{P}_{\mathcal{M}}\big[\nabla\hat{\eta}(\cdot,t)\big](\cdot)\Big)^{\mathrm{tr}} \\ &= \Big(\Big[\mathcal{I} - \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}(\cdot)\,\big(\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}(\cdot)\big)^{\mathrm{tr}}\Big]\nabla\hat{\eta}(\cdot,t)\Big)^{\mathrm{tr}} \quad \text{on} \quad \mathcal{M},\end{aligned}$$

cf. the corresponding definition in equation (2.2). It is easy to see that the rest of the theory which has been presented up to now can be adapted in the same straightforward way.

2.5.2. Evolving hypersurfaces

In case of a hypersurface that is time-dependent itself, we are interested in an evolving hypersurface $\mathcal{M}(t)$ embedded in $\mathbb{R}^d$, observed during a specified time period $[0,T]$, and in surface differential operators which act on time-dependent surface fields $\eta(\cdot,t)\colon \mathcal{M}(t) \to \mathbb{R}$ and $\boldsymbol{\xi}(\cdot,t)\colon \mathcal{M}(t) \to \mathbb{R}^d$ that live on the evolving hypersurface. This setting is even more general than the one considered in Section 2.5.1.

However, we can still build upon what we already have if we assume that the restriction of $\mathcal{M}(t)$ to an arbitrary but fixed time t_0 is a smooth $(d-1)$-dimensional hypersurface which is oriented by the associated $t = t_0$ time slice of a surface field $\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}(\cdot,t)\colon \mathcal{M}(t) \to \mathbb{R}^d$ of unit normal vectors to $\mathcal{M}(t)$. Moreover, we need to require that each time slice of the fields η and $\boldsymbol{\xi}$ has a spatially differentiable extension $\hat{\eta}(\cdot,t)\colon \mathcal{N}\big(\mathcal{M}(t)\big) \to \mathbb{R}$ and $\hat{\boldsymbol{\xi}}(\cdot,t)\colon \mathcal{N}\big(\mathcal{M}(t)\big) \to \mathbb{R}^d$ to a neighborhood $\mathcal{N}\big(\mathcal{M}(t)\big) \subset \mathbb{R}^d$ of $\mathcal{M}(t)$ with $\mathrm{meas}_{\mathbb{R}^d}\big[\mathcal{N}\big(\mathcal{M}(t)\big)\big] > 0$. In this case, all our definitions, theorems and statements from Sections 2.1–2.4 carry over again since they only act on the spatial domain of the surface fields and since we consider a suitable type of hypersurface at each fixed point in time. They can be applied separately for each time t and the associated fixed-in-time hypersurface.

In particular, without changing the definition of the projection operator in equation (2.1), which is the essential ingredient in the previous sections, we obtain an operator

$$\mathcal{P}_{\mathcal{M}} := \mathcal{I} - \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}\,(\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})^{\mathrm{tr}}$$

on the space of time-dependent surface vector fields on $\mathcal{M}(t)$. It projects each such field onto the subspace of time-dependent surface vector fields on $\mathcal{M}(t)$ that are tangential to $\mathcal{M}(t)$ at each time t. Using this projection operator, we can now define the time-dependent surface gradient of η on $\mathcal{M}(t)$, for example. Applying the definition in equation (2.2) pointwise with respect to time, it is given by

$$\big[\mathrm{grad}_{\mathcal{M}}\,\eta\big](\cdot,t) := \Big(\big[\mathcal{P}_{\mathcal{M}}\nabla\hat{\eta}\big](\cdot,t)\Big)^{\mathrm{tr}} \quad \text{on} \quad \mathcal{M}(t),$$

which can be written in simplified form as

$$\mathrm{grad}_{\mathcal{M}}\,\eta = \big(\mathcal{P}_{\mathcal{M}}\nabla\hat{\eta}\big)^{\mathrm{tr}},$$

cf. the corresponding definition in equation (2.2) as well as the corresponding definition for time-dependent fields on static hypersurfaces in Section 2.5.1. Again, it is easy to see that the rest of the theory from Sections 2.1–2.4 can be generalized in a similar manner. We would like to mention explicitly, though, that the concepts from Section 2.3 carry over if the existance of a time-dependent scalar field $\Phi(\cdot,t)\colon \mathcal{N}\big(\mathcal{M}(t)\big) \to \mathbb{R}$ is assumed which fulfills time-dependent analogues to the Φ-related requirements in Section 2.3, e.g. being twice differentiable in space.

Surface differential operators on evolving hypersurfaces, which also consider the time domain of a surface field will be defined and investigated in the next section.

2.6. Additional calculus on evolving hypersurfaces

In this section, let $\mathcal{M}(t)$ be an evolving, smooth $(d-1)$-dimensional hypersurface embedded in $\mathbb{R}^d$ which is orientable. Let $\mathcal{M}_t := \bigcup_{t\in[0,T]} \mathcal{M}(t) \times \{t\}$ be its space–time representation and $\mathcal{N}(\mathcal{M}_t)$ denote an associated space–time neighborhood that is of the similar form $\mathcal{N}(\mathcal{M}_t) := \bigcup_{t\in[0,T]} \mathcal{N}\big(\mathcal{M}(t)\big) \times \{t\}$, where $\mathcal{N}\big(\mathcal{M}(t)\big) \subset \mathbb{R}^d$, at each fixed time t, is some neighborhood of $\mathcal{M}(t)$ with $\mathrm{meas}_{\mathbb{R}^d}\big[\mathcal{N}\big(\mathcal{M}(t)\big)\big] > 0$. Furthermore, let $\mathbf{v}_{\boldsymbol{\mathcal{M}}} \colon \mathcal{M}_t \to \mathbb{R}^d$ be a surface field which describes the material velocity of $\mathcal{M}(t)$ and $\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}} \colon \mathcal{M}_t \to \mathbb{R}^d$ denote a surface field of unit normal vectors to $\mathcal{M}(t)$ which specifies an orientation for $\mathcal{M}(t)$. Finally, let $\eta \colon \mathcal{M}_t \to \mathbb{R}$ be some time-dependent surface scalar field living on $\mathcal{M}(t)$.

For formulating PDEs on evolving hypersurfaces and dealing with those equations, additional surface differential operators and associated integral calculus are required. In particular, we need the *material derivative* of scalar fields on evolving hypersurfaces. Having briefly introduced this derivative in Section 1.1.2, we now start with its detailed discussion.

The material derivative $\partial^\bullet \eta$ of η (with respect to the material velocity $\mathbf{v}_{\boldsymbol{\mathcal{M}}}$) describes the temporal rate of change of η while following the trajectory of material points on $\mathcal{M}(t)$. Locally, at an arbitrary but fixed point $\big(\underline{\mathbf{x_0}}, t_0\big)$ with $\underline{\mathbf{x_0}} \in \mathcal{M}(t_0)$, it can be defined as

$$\big[\partial^\bullet \eta\big]\big(\underline{\mathbf{x_0}}, t_0\big) := \left[\frac{\mathrm{d}}{\mathrm{d}t} \eta\big(\tilde{\mathbf{x}}_{\mathbf{0}}(t), t\big)\right]_{t=t_0}, \tag{2.21}$$

where $\tilde{\mathbf{x}}_{\mathbf{0}}(t)$ denotes the trajectory of the material point visiting $\underline{\mathbf{x_0}}$ at time t_0. The latter is the trajectory through $\big(\underline{\mathbf{x_0}}, t_0\big)$ corresponding to the material velocity $\mathbf{v}_{\boldsymbol{\mathcal{M}}}$, which is characterized by the ordinary differential equation

$$\tilde{\mathbf{x}}_{\mathbf{0}}{}'(t) = \mathbf{v}_{\boldsymbol{\mathcal{M}}}\big(\tilde{\mathbf{x}}_{\mathbf{0}}(t), t\big), \qquad \tilde{\mathbf{x}}_{\mathbf{0}}(t_0) = \underline{\mathbf{x_0}}. \tag{2.22}$$

If the field η has a differentiable extension $\hat{\eta}$ to a space–time neighborhood of the evolving hypersurface, the right-hand side in equation (2.21) can be reformulated in terms of the extension's classical partial derivatives in $\mathbb{R}^d$. In particular, applying the chain rule and using characterization (2.22) in this

case yields

$$\begin{aligned}\left[\frac{\mathrm{d}}{\mathrm{d}t}\eta\big(\tilde{\mathbf{x}}_{\mathbf{0}}(t),t\big)\right]_{t=t_0} &= \Big[\nabla\hat{\eta}\big(\tilde{\mathbf{x}}_{\mathbf{0}}(t),t\big)\cdot\tilde{\mathbf{x}}_{\mathbf{0}}{}'(t)+\partial_t\hat{\eta}\big(\tilde{\mathbf{x}}_{\mathbf{0}}(t),t\big)\cdot 1\Big]_{t=t_0}\\ &= \big[\partial_t\hat{\eta}+\mathbf{v}_{\boldsymbol{\mathcal{M}}}\cdot\nabla\hat{\eta}\big]\big(\tilde{\mathbf{x}}_{\mathbf{0}}(t),t\big)\Big|_{t=t_0}\\ &= \big[\partial_t\hat{\eta}+\mathbf{v}_{\boldsymbol{\mathcal{M}}}\cdot\nabla\hat{\eta}\big]\big(\underline{\mathbf{x}_{\mathbf{0}}},t_0\big).\end{aligned}$$

Since $\big(\underline{\mathbf{x}_{\mathbf{0}}},t_0\big)$ can be chosen arbitrarily, we obtain the following theorem.

Theorem 2.6.1 (Representation of $\partial^\bullet\eta$ in terms of classical partial derivatives in $\mathbb{R}^d$)**.** *If η has a differentiable extension $\hat{\eta}$ to a space–time neighborhood $\mathcal{N}(\mathcal{M}_t)$ of $\mathcal{M}(t)$, the material derivative of η can be represented as*

$$\partial^\bullet\eta = \partial_t\hat{\eta}+\mathbf{v}_{\boldsymbol{\mathcal{M}}}\cdot\nabla\hat{\eta}\quad \textit{on}\quad \mathcal{M}_t.$$

The material derivative of scalar fields on evolving hypersurfaces has a key role in the following theorem which is an evolving hypersurface analogue to the Reynolds transport theorem for evolving bulk domains (see e.g. Belytschko et al., 2000, Equation (3.5.11) in Section 3.5.3).

Theorem 2.6.2 (Transport relation for evolving material hypersurfaces (see also Cermelli et al., 2005, Section 5.2))**.** *Let the material velocity $\mathbf{v}_{\boldsymbol{\mathcal{M}}}$ have a spatially differentiable extension to a space–time neighborhood $\mathcal{N}(\mathcal{M}_t)$ of $\mathcal{M}(t)$, such that $\nabla_{\mathcal{M}}\cdot\mathbf{v}_{\boldsymbol{\mathcal{M}}}$ is defined at each fixed time t. Then the identity*

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathcal{M}(t)}\eta\,\mathrm{d}\sigma = \int_{\mathcal{M}(t)}\partial^\bullet\eta+\eta\big(\nabla_{\mathcal{M}}\cdot\mathbf{v}_{\boldsymbol{\mathcal{M}}}\big)\,\mathrm{d}\sigma \tag{2.23}$$

holds at each fixed time t.

Proof. In Cermelli et al. (2005, Section 5.2), the theorem is proved as a special case of more general transport relations for evolving hypersurfaces. A direct proof which additionally requires that η has an extension to the space–time neighborhood $\mathcal{N}(\mathcal{M}_t)$ which is differentiable with respect to both space and time can be found in Dziuk and Elliott (2007a, Lemma 2.2 and Appendix A). □

We would like to point out that a surface differential operator of the form $\partial^\bullet\diamond+\diamond\big(\nabla_{\mathcal{M}}\cdot\mathbf{v}_{\boldsymbol{\mathcal{M}}}\big)$ is applied in the integrand on the right-hand side of identity (2.23). Note that this operator is also applied in continuity equations (1.5b), (1.5$\tilde{\text{b}}$) and (1.6b). In the latter equations, it generates those terms which are related to conservative material transport driven by the evolution of the hypersurface. Finding those terms in continuity equations of this kind is no coincidence. There is a close connection between Theorem 2.6.2 and continuity equations on evolving hypersurfaces which will be discussed in Section 3.1.1.

Due to the importance of the surface differential operator which has just been looked at, we close this section by deriving a representation of this operator in the level set framework. Our findings are summarized as a theorem in the following subsection.

2.6.1. Conservative material transport in the level set framework

Let exist a time-dependent scalar field $\Phi\colon \mathcal{N}(\mathcal{M}_t) \to \mathbb{R}$ which has $\mathcal{M}(t)$ as one of its level sets and is a space–time generalization of the static scalar field Φ from Section 2.3. Hence, let Φ fulfill time-dependent analogues to the Φ-related requirements in Section 2.3, e.g. being twice differentiable in space.

Theorem 2.6.3 (Representation of $\partial^\bullet\eta+\eta\big(\nabla_{\mathcal{M}}\cdot\mathbf{v}_{\boldsymbol{\mathcal{M}}}\big)$ in the level set framework)**.** *Let η have a differentiable extension $\hat{\eta}$ to a space–time neighborhood $\mathcal{N}(\mathcal{M}_t)$ of $\mathcal{M}(t)$ and the material velocity $\mathbf{v}_{\boldsymbol{\mathcal{M}}}$ have a spatially differentiable extension $\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}$ to $\mathcal{N}(\mathcal{M}_t)$ which describes the velocity of the level sets of Φ. Furthermore, assume that Φ has symmetric second derivatives with respect to space and symmetric mixed second derivatives with respect to space and time, e.g. $\Phi\in C^2\big(\mathcal{N}(\mathcal{M}_t)\big)$. Then*

$$\partial^\bullet\eta+\eta\big(\nabla_{\mathcal{M}}\cdot\mathbf{v}_{\boldsymbol{\mathcal{M}}}\big)=\frac{1}{|\nabla\Phi|}\Big(\partial_t\big[|\nabla\Phi|\,\hat{\eta}\big]+\nabla\cdot\big(|\nabla\Phi|\,\hat{\eta}\,\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\big)\Big)\quad\text{on}\quad\mathcal{M}_t.$$

Proof. Let $\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}$ be the canonical extension of $\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}$ which is associated with Φ, i.e. $\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}=|\nabla\Phi|^{-1}\nabla\Phi$. By successively applying Theorem 2.6.1 and the first part of Theorem 2.3.2 for $\boldsymbol{\xi}=\mathbf{v}_{\boldsymbol{\mathcal{M}}}$, and by subsequently using the linearity of the classical divergence operator in $\mathbb{R}^d$ and the product rule twice, we get

$$\begin{aligned}
&|\nabla\Phi|\Big[\partial^\bullet\eta+\eta\big(\nabla_{\mathcal{M}}\cdot\mathbf{v}_{\boldsymbol{\mathcal{M}}}\big)\Big]\\
&\quad=|\nabla\Phi|\big(\partial_t\hat{\eta}+\mathbf{v}_{\boldsymbol{\mathcal{M}}}\cdot\nabla\hat{\eta}\big)+|\nabla\Phi|\big(\nabla_{\mathcal{M}}\cdot\mathbf{v}_{\boldsymbol{\mathcal{M}}}\big)\eta\\
&\quad=|\nabla\Phi|\,\partial_t\hat{\eta}+|\nabla\Phi|\big(\mathbf{v}_{\boldsymbol{\mathcal{M}}}\cdot\nabla\hat{\eta}\big)+\nabla\cdot\Big(|\nabla\Phi|\big[\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}-\big(\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\cdot\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big]\Big)\eta\\
&\qquad\quad+\Big[|\nabla\Phi|(\mathbf{v}_{\boldsymbol{\mathcal{M}}}\cdot\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})\big(\nabla\cdot\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)\Big]\eta\\
&\quad=|\nabla\Phi|\,\partial_t\hat{\eta}+|\nabla\Phi|\big(\mathbf{v}_{\boldsymbol{\mathcal{M}}}\cdot\nabla\hat{\eta}\big)+\nabla\cdot\Big(|\nabla\Phi|\big[\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}-\big(\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\cdot\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big]\Big)\eta\\
&\qquad\quad+\nabla\cdot\Big(|\nabla\Phi|\big(\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\cdot\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\Big)\eta-\nabla\Big[|\nabla\Phi|\big(\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\cdot\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)\Big]\cdot(\eta\,\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})\\
&\quad=-\nabla\Big[|\nabla\Phi|\big(\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\cdot\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)\Big]\cdot(\eta\,\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})+|\nabla\Phi|\,\partial_t\hat{\eta}\\
&\qquad\quad+|\nabla\Phi|\big(\mathbf{v}_{\boldsymbol{\mathcal{M}}}\cdot\nabla\hat{\eta}\big)+\nabla\cdot\big(|\nabla\Phi|\,\hat{\eta}\,\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\big)\eta\\
&\quad=(\eta\,\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})\cdot\nabla\Big[-|\nabla\Phi|\big(\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\cdot\hat{\boldsymbol{\nu}}_{\boldsymbol{\mathcal{M}}}\big)\Big]+|\nabla\Phi|\,\partial_t\hat{\eta}+\nabla\cdot\big(|\nabla\Phi|\,\hat{\eta}\,\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\big)\\
&\quad=(\eta\,\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}})\cdot\nabla\big[-\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\cdot\nabla\Phi\big]+|\nabla\Phi|\,\partial_t\hat{\eta}+\nabla\cdot\big(|\nabla\Phi|\,\hat{\eta}\,\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}\big).
\end{aligned}$$

Since, by assumption, the extension $\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}}$ of $\mathbf{v}_{\boldsymbol{\mathcal{M}}}$ describes the velocity of the

level sets of Φ in $\mathcal{N}(\mathcal{M}_t)$, we have $\partial_t \Phi + \hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}} \cdot \nabla \Phi = 0$ in $\mathcal{N}(\mathcal{M}_t)$ (for details, see the theory in Section 3.3; note that this theory is not based on the theorem which we are currently proving). Therefore, the left-most term on the right-hand side can be rewritten as

$$(\eta\, \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}) \cdot \nabla\big[-\hat{\mathbf{v}}_{\boldsymbol{\mathcal{M}}} \cdot \nabla \Phi\big] = \Big(\nabla\big[\partial_t \Phi\big] \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}\Big)\, \eta = \partial_t |\nabla \Phi|\, \eta,$$

where, in the last step, we have used that

$$\nabla\big[\partial_t \Phi\big] \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}} = \partial_t |\nabla \Phi|.$$

The latter equality can be seen completely analogously to Remark 2.3.4, using that Φ has symmetric mixed second derivatives with respect to space and time by assumption. We conclude the proof by noting that

$$\partial_t |\nabla \Phi|\, \eta + |\nabla \Phi|\, \partial_t \hat{\eta} = \partial_t \big[|\nabla \Phi|\, \hat{\eta}\big]$$

due to the product rule. □

3. Further mathematical background

In Chapter 2, we looked at concepts from calculus that are necessary for formulating PDEs on hypersurfaces and dealing with those equations. In the course of this, a special focus has been laid on properties which can be exploited by level set extension based numerical schemes for systems with surface PDEs, such as the UDG schemes which will be introduced in Chapter 4 and Chapter 5. In this chapter, we continue with further important mathematical background regarding those UDG schemes. We discuss the fundamentals of continuity equations formulated on bulk domains and hypersurfaces, as well as other tools from pure and numerical mathematics which are important ingredients of our schemes.

We start in Section 3.1 by discussing mathematical equations known as *conservation laws*, covering their formulations for bulk domains and hypersurfaces. Those laws can be used to express important physical principles and lead to the notion of continuity equations. Conservation laws are particularly relevant when designing flux-based numerical methods for continuity equations, such as DG methods in general. In Section 3.2, we give an extensive introduction to fitted DG methods for elliptic and parabolic bulk equations, mainly focussing on those methods that are relevant within the scope of this thesis. Section 3.3 offers a detailed description of the level set framework, i.e. the framework for geometry description which is employed by the UDG schemes that we develop in this thesis. As part of this description, we carefully examine the key player in this framework, and state associated mathematical assumptions which are required for obtaining well-defined UDG schemes later on.

3.1. Conservation laws and continuity equations

Most of the bulk PDEs and surface PDEs in Chapter 1 and particularly the class of bulk–surface models which is considered in Section 1.2 are based on physical principles that can be expressed by continuity equations like (1.8a) and (1.8c), and by equivalent conservation laws. These physical principles are conservation of mass or conservation of electric charge, for example. In this thesis, we are specifically dealing with conserved quantities that live on hypersurfaces and conserved quantities living in bulk domains. Distinguishing between those two types of quantities, we will now discuss the notions of conservation laws and continuity equations in a mathematically rigorous way.

3.1.1. Conserved quantities on hypersurfaces

Let $\mathcal{M}(t)$ be an evolving, orientable, smooth $(d-1)$-dimensional hypersurface embedded in $\mathbb{R}^d$, which is observed during a specified time period $[0,T]$. At each time t where the hypersurface has a non-empty boundary (i.e. where $\mathcal{M}(t)$ is an open hypersurface), let this boundary $\partial\mathcal{M}(t)$ not be part of the set $\mathcal{M}(t)$, as usual in this thesis. Let $\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}(\cdot,t)\colon \mathcal{M}(t) \to \mathbb{R}^d$ denote a field of unit normal vectors to $\mathcal{M}(t)$ which specifies an orientation for $\mathcal{M}(t)$ and let $\mathbf{v}_{\boldsymbol{\mathcal{M}}}(\cdot,t)\colon \mathcal{M}(t) \to \mathbb{R}^d$ be a field which describes the material velocity of $\mathcal{M}(t)$. Moreover, let $M(t) \subseteq \mathcal{M}(t)$ denote an arbitrary $(d-1)$-dimensional portion of $\mathcal{M}(t)$ moving with the material velocity $\mathbf{v}_{\boldsymbol{\mathcal{M}}}$, whose boundary $\partial M(t)$ is either empty (if $M(t) = \mathcal{M}(t)$ and $\mathcal{M}(t)$ is a closed hypersurface) or a smooth $(d-2)$-dimensional manifold.

When saying that some scalar quantity which is reasonable on $\mathcal{M}(t)$ is a *conserved quantity* on $\mathcal{M}(t)$, we refer to the principle that, in every surface portion $M(t)$ of the type specified above, the amount of the quantity can only change by the amount which passes in or out through the boundary $\partial M(t)$ and by the amount being generated or removed inside $M(t)$. Denoting the concentration (i.e. the volume density) of the quantity with respect to $\mathcal{M}(t)$ by u_s, this can be expressed by the following equation, which we call the *general conservation law* for quantities on hypersurfaces (cf. Dziuk and Elliott, 2007a; Barreira et al., 2011):

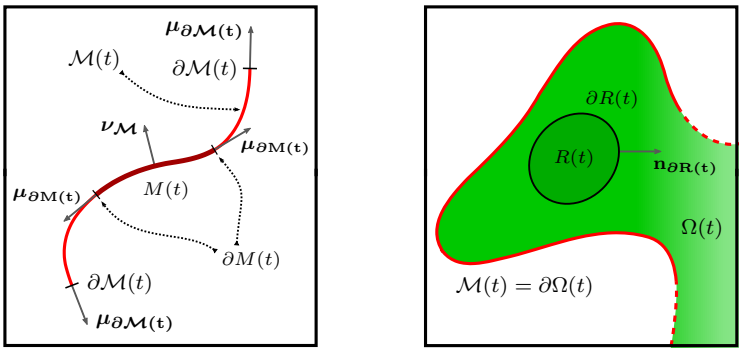
$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{M(t)} u_s \,\mathrm{d}\sigma = -\int_{\partial M(t)} \mathbf{q_s}\cdot\boldsymbol{\mu}_{\boldsymbol{\partial\mathbf{M}(\mathbf{t})}} \,\mathrm{d}\varsigma + \int_{M(t)} g_s \,\mathrm{d}\sigma \quad \forall M(t) \subseteq \mathcal{M}(t). \tag{3.1}$$

In this equation, the terms $\mathbf{q_s}$ and g_s are fields reasonable on $\mathcal{M}(t)$, with images $\mathbf{q_s}(\underline{\mathbf{x}},t) \in \mathbb{R}^d$ and $g_s(\underline{\mathbf{x}},t) \in \mathbb{R}$, which are both defined by constitutive laws. The vector field $\mathbf{q_s}$ describes the flux of the quantity and the scalar field g_s the density of its generation/removal. Note that $\mathbf{q_s}$ usually dependents on u_s and this may also be the case for g_s. Furthermore, $\mathrm{d}\sigma$ denotes the volume element with respect to $\mathcal{M}(t)$, $\mathrm{d}\varsigma$ denotes the surface element with respect to $\mathcal{M}(t)$, and $\boldsymbol{\mu}_{\boldsymbol{\partial\mathbf{M}(\mathbf{t})}}$ is the field of intrinsic outward-pointing unit normal vectors to $\partial M(t)$ which has already been considered in Section 2.4. See Figure 3.1a for an illustration of the geometrical setting.

Every surface portion $M(t)$ of the type specified above is a hypersurface itself, which inherits its properties from $\mathcal{M}(t)$. Hence, if we assume that the material velocity $\mathbf{v}_{\boldsymbol{\mathcal{M}}}$ has a spatially differentiable extension to a space–time neighborhood of $\mathcal{M}(t)$, the transport relation from Section 2.6 (Theorem 2.6.2) yields

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{M(t)} u_s \,\mathrm{d}\sigma = \int_{M(t)} \partial^\bullet u_s + u_s\big(\nabla_{\mathcal{M}}\cdot\mathbf{v}_{\boldsymbol{\mathcal{M}}}\big)\,\mathrm{d}\sigma.$$

Here $\partial^\bullet u_s$ is the material derivative of u_s with respect to $\mathbf{v}_{\boldsymbol{\mathcal{M}}}$, which has been discussed in Section 2.6. Moreover, if we assume that the flux $\mathbf{q_s}$ and the unit normal vector field $\boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}}$ have spatially differentiable extensions to a space–time



(a) Setting: Hypersurfaces. (b) Setting: Bulk domains.

Figure 3.1.: Geometrical entities that appear in the discussion of conservation laws for (a) quantities on hypersurfaces $\mathcal{M}(t)$ (see Section 3.1.1) and for (b) quantities in bulk domains $\Omega(t)$ (see Section 3.1.2).

neighborhood of $\mathcal{M}(t)$, and assume that $\mathbf{q_s}$ is continuously differentiable on $\mathcal{M}(t) \cup \partial\mathcal{M}(t)$ with respect to the spatial variable $\underline{\mathbf{x}}$, the divergence theorem from Section 2.4 (Theorem 2.4.1) yields

$$\int_{\partial M(t)} \mathbf{q_s} \cdot \boldsymbol{\mu_{\partial M(t)}} \, \mathrm{d}\varsigma = \int_{M(t)} \nabla_{\mathcal{M}} \cdot \mathbf{q_s} \, \mathrm{d}\sigma + \int_{M(t)} \big(\mathbf{q_s} \cdot \boldsymbol{\nu_{\mathcal{M}}}\big) H_{\mathcal{M}} \, \mathrm{d}\sigma.$$

Looking at equation (3.1), the above implication of the transport relation can be applied to the left-hand side of the equation, and the above implication of the divergence theorem can be applied to its right-hand side. Provided that the necessary assumptions regarding $\mathbf{v}_{\boldsymbol{\mathcal{M}}}$, $\mathbf{q_s}$ and $\boldsymbol{\nu_{\mathcal{M}}}$ hold, equation (3.1) can therefore be rewritten in the following way. For all surface portions $M(t) \subseteq \mathcal{M}(t)$ of the type specified above, we have the identity

$$\int_{M(t)} \partial^{\bullet} u_s + u_s\big(\nabla_{\mathcal{M}} \cdot \mathbf{v}_{\boldsymbol{\mathcal{M}}}\big) + \nabla_{\mathcal{M}} \cdot \mathbf{q_s} + \big(\mathbf{q_s} \cdot \boldsymbol{\nu_{\mathcal{M}}}\big) H_{\mathcal{M}} - g_s \, \mathrm{d}\sigma = 0.$$

If we assume, in addition, that all terms in the integrand are continuous on $\mathcal{M}(t)$ with respect to the spatial variable $\underline{\mathbf{x}}$, the latter equation, since it holds for every surface portion $M(t)$ of the type specified above, is equivalent to the surface PDE

$$\partial^{\bullet} u_s + u_s\big(\nabla_{\mathcal{M}} \cdot \mathbf{v}_{\boldsymbol{\mathcal{M}}}\big) + \nabla_{\mathcal{M}} \cdot \mathbf{q_s} + \big(\mathbf{q_s} \cdot \boldsymbol{\nu_{\mathcal{M}}}\big) H_{\mathcal{M}} = g_s \quad \text{on} \quad \mathcal{M}(t). \tag{3.2}$$

PDEs derived from conservation laws are commonly known as continuity equations or pointwise conservation laws. Calling equation (3.1) the general conservation law for quantities on hypersurfaces, equation (3.2) can be referred to as the associated *general continuity equation*. It is general in the sense that it

is usable to derive any specific continuity equation on the evolving hypersurface $\mathcal{M}(t)$, including the static geometry special case $\mathbf{v}_{\boldsymbol{\mathcal{M}}} \equiv \underline{\mathbf{0}}$, by particular choices of the flux $\mathbf{q_s}$ and the source/sink density g_s. It has to be supplemented with initial values $u_s(\cdot, 0)$ on $\mathcal{M}(0)$. Furthermore, an appropriate boundary condition is required where the boundary of $\mathcal{M}(t)$ is not empty (i.e. where $\mathcal{M}(t)$ is an open hypersurface) during the observation period $(0, T]$.

Remark 3.1.1 (Tangential flux)**.** *If we think of processes which solely happen inside of $\mathcal{M}(t)$, it is quite natural to have a flux $\mathbf{q_s}$ tangential to $\mathcal{M}(t)$, i.e., a flux with $\mathbf{q_s} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}} = 0$ on $\mathcal{M}(t)$. However, it should be noted that we do not need to assume such a tangential flux. In conservation law* (3.1)*, only the tangential component $\mathbf{q_s} \cdot \boldsymbol{\mu}_{\boldsymbol{\partial \mathbf{M}(\mathbf{t})}}$ of the flux is considered anyway. This results in the curvature term in continuity equation* (3.2)*, which automatically removes the effect of any normal component of $\mathbf{q_s}$ on $\mathcal{M}(t)$. Assuming a tangential flux would simply make this curvature term vanish. Furthermore, by employing equation* (2.13) *from Section 2.2 to rewrite the curvature term in terms of the surface divergence operator, and by subsequently using the first statement of Theorem 2.2.8, it can be shown easily that continuity equation* (3.2) *can be equivalently formulated as*

$$\partial^\bullet u_s + u_s\big(\nabla_{\mathcal{M}} \cdot \mathbf{v}_{\boldsymbol{\mathcal{M}}}\big) + \nabla_{\mathcal{M}} \cdot \mathcal{P}_{\mathcal{M}} \mathbf{q_s} = g_s \quad \text{on} \quad \mathcal{M}(t).$$

Note that the term $\mathcal{P}_{\mathcal{M}} \mathbf{q_s}$ corresponds to the tangential component of the flux.

3.1.2. Conserved quantities in bulk domains

Let $\Omega(t)$ be an evolving bulk domain in $\mathbb{R}^d$ which is bounded by an evolving, smooth $(d-1)$-dimensional hypersurface. Let $\Omega(t)$ be observed during a specified time period $[0, T]$ and let $\mathbf{v}(\cdot, t) \colon \Omega(t) \to \mathbb{R}^d$ denote a field which describes the material velocity of $\Omega(t)$. Moreover, let $R(t) \subseteq \Omega(t)$ denote an arbitrary, bounded d-dimensional portion of $\Omega(t)$ moving with the material velocity $\mathbf{v}$, whose boundary $\partial R(t)$ is a smooth $(d-1)$-dimensional hypersurface.

With considerations analogous to those in Section 3.1.1 for quantities on hypersurfaces, we obtain the notion of conserved scalar quantities in bulk domains like $\Omega(t)$. Denoting its concentration with respect to $\Omega(t)$ by u_b, conservation of a scalar quantity in $\Omega(t)$ can be expressed by

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{R(t)} u_b \, \mathrm{d}x = - \int_{\partial R(t)} \mathbf{q_b} \cdot \mathbf{n}_{\boldsymbol{\partial \mathbf{R}(\mathbf{t})}} \, \mathrm{d}\sigma + \int_{R(t)} g_b \, \mathrm{d}x \quad \forall R(t) \subseteq \Omega(t). \tag{3.3}$$

In this equation, which we call the *general conservation law* for quantities in bulk domains, the terms $\mathbf{q_b}$ and g_b denote fields in $\Omega(t)$ that describe the flux and the source/sink density of the quantity. Their images $\mathbf{q_b}(\underline{\mathbf{x}}, t) \in \mathbb{R}^d$ and $g_b(\underline{\mathbf{x}}, t) \in \mathbb{R}$ are both defined by constitutive laws, where the constitutive law for $\mathbf{q_b}$ usually involves the concentration u_b. This may also be the case for the constitutive law for g_b. The differential $\mathrm{d}x$ denotes the volume element

with respect to $\Omega(t)$, $\mathrm{d}\sigma$ denotes the surface element with respect to $\Omega(t)$ and $\mathbf{n}_{\partial\mathbf{R}(\mathbf{t})}$ is the field of outward-pointing unit normal vectors to $\partial R(t)$. See Figure 3.1b for an illustration of the geometrical setting.

Proceeding similar to Section 3.1.1, we can formulate equation (3.3) by the following equivalent bulk PDE:

$$\partial_t u_b + \nabla \cdot (u_b \mathbf{v}) + \nabla \cdot \mathbf{q_b} = g_b \quad \text{on} \quad \Omega(t). \tag{3.4}$$

Again, calling equation (3.3) the general conservation law for quantities in bulk domains, equation (3.4) can be referred to as the associated *general continuity equation*. It is usable to derive any specific continuity equation formulated on the evolving bulk domain $\Omega(t)$, including the static geometry special case $\mathbf{v} \equiv \underline{\mathbf{0}}$, by particular choices of the flux $\mathbf{q_b}$ and the source/sink density g_b. It has to be supplemented with initial values $u_b(\cdot, 0)$ on $\Omega(0)$, and an appropriate boundary condition is required on $\partial\Omega(t)$, this time without restriction of any kind.

The equivalence of general conservation law (3.3) and general continuity equation (3.4) follows under assumptions on the regularity of u_b, $\mathbf{v}$, $\mathbf{q_b}$ and g_b which are analogous to those of the corresponding terms in Section 3.1.1. It follows from the Reynolds transport theorem (see e.g. Belytschko et al., 2000, Equation (3.5.14) in Section 3.5.3) and the classical divergence theorem in $\mathbb{R}^d$. For every surface portion $R(t) \subseteq \Omega(t)$ of the type specified above, these theorems yield reformulations

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{R(t)} u_b \, \mathrm{d}x = \int_{R(t)} \partial_t u_b + \nabla \cdot (u_b \mathbf{v}) \, \mathrm{d}x$$

and

$$\int_{\partial R(t)} \mathbf{q_b} \cdot \mathbf{n}_{\partial\mathbf{R}(\mathbf{t})} \, \mathrm{d}\sigma = \int_{R(t)} \nabla \cdot \mathbf{q_b} \, \mathrm{d}x,$$

respectively, which can be applied in equation (3.3).

3.1.3. Additional remarks

In models that are based on continuity equations and shall be simulated using numerical methods, e.g. in many models for biological processes, the physical principles which are described by the continuity equations and their underlying equivalent conservation laws often are a fundamental hypothesis. For computational modeling in cell biology, for instance, especially conservation of mass is an important fundamental hypothesis (see e.g. Otsuji et al., 2007; Novak et al., 2007; Mori et al., 2008; Rubinstein et al., 2012).

Therefore, numerical schemes are preferable which incorporate a discrete analogue to conservation laws derived from general conservation laws (3.1) and (3.3), such that the numerical approximation of the solution reflects the considered physical principles in the best possible way. In the next section,

we consider DG methods for discretization of a certain class of bulk PDEs that are derivable from stationary analogues to equation (3.3), and for spatial discretization of the time-dependent counterpart of this class of bulk PDEs. As we will see for the latter class of time-dependent equations, DG methods take into account the underlying bulk conservation laws in a natural way.

3.2. Fitted DG methods for elliptic and parabolic bulk PDEs

Let Ω be a static bulk domain in $\mathbb{R}^d$ which is bounded by a static, smooth $(d-1)$-dimensional hypersurface. Assume that this hypersurface $\partial\Omega$ comprises two disjunct subsets $\partial\Omega_D$ and $\partial\Omega_N$, where $\partial\Omega_D = \emptyset$ (i.e. $\partial\Omega = \partial\Omega_N$) and $\partial\Omega_N = \emptyset$ (i.e. $\partial\Omega = \partial\Omega_D$) shall be valid special cases.

On top of this geometrical setting, we first consider the steady-state diffusion equation (1.2a), together with the combination of a Dirichlet boundary condition on $\partial\Omega_D$ and a Neumann boundary condition on $\partial\Omega_N$. This elliptic boundary value problem of second order can be stated in the following way. We wish to find $u_b \colon \Omega \to \mathbb{R}$ with

$$-\nabla \cdot (\mathcal{D}_b \nabla u_b) = g_b \quad \text{in} \quad \Omega, \tag{3.5a}$$

$$u_b = g_{b,D} \quad \text{on} \quad \partial\Omega_D, \tag{3.5b}$$

$$-\mathcal{D}_b \nabla u_b \cdot \mathbf{n}_{\boldsymbol{\partial\Omega_N}} = g_{b,N} \quad \text{on} \quad \partial\Omega_N, \tag{3.5c}$$

where the bulk diffusivity tensor $\mathcal{D}_b$, and g_b, $g_{b,D}$ and $g_{b,N}$ are given data.

3.2.1. Obtaining DG methods by choosing numerical fluxes

In Arnold et al. (2002), a framework has been introduced which allows for the derivation and analysis of a large class of DG methods for second order elliptic problems. For the sake of simplicity and easy explanation of the main ideas, the presentation was restricted to Poisson's equation (1.1a) with homogeneous Dirichlet boundary values, i.e. to the special case $\mathcal{D}_b := \mathcal{I}$ of equation (3.5a), in combination with the special case $g_{b,D} := 0$ of boundary condition (3.5b), only considering the setting $\partial\Omega = \partial\Omega_D$. For being able to deal with general model problems of the form (3.5), we now slightly generalize the framework's original presentation.

Weak formulation of the problem

The framework by Arnold et al. (2002) is based on some reformulation that is often called the problem's *mixed formulation*. Still looking for a scalar field $u_b \colon \Omega \to \mathbb{R}$ which solves model problem (3.5), we introduce an auxiliary vector field $\boldsymbol{\sigma}_\mathbf{b} := -\mathcal{D}_b \nabla u_b$ and write the problem as a system of first order PDEs

$$\boldsymbol{\sigma}_\mathbf{b} = -\mathcal{D}_b \nabla u_b \quad \text{in} \quad \Omega, \tag{3.6a}$$

$$\nabla \cdot \boldsymbol{\sigma}_\mathbf{b} = g_b \quad \text{in} \quad \Omega, \tag{3.6b}$$

together with boundary conditions

$$u_b = g_{b,D} \qquad \text{on} \quad \partial\Omega_D, \tag{3.6c}$$

$$\boldsymbol{\sigma_b} \cdot \mathbf{n}_{\boldsymbol{\partial\Omega_N}} = g_{b,N} \qquad \text{on} \quad \partial\Omega_N. \tag{3.6d}$$

To apply the framework, we derive a suitable *weak formulation* of model problem (3.5) from its mixed formulation (3.6). Using the equality

$$-\boldsymbol{\sigma_b} \cdot \boldsymbol{\psi_b} = \mathcal{D}_b \nabla u_b \cdot \boldsymbol{\psi_b} = \nabla u_b \cdot \mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi_b},$$

we formally obtain

$$-\int_R \boldsymbol{\sigma_b} \cdot \boldsymbol{\psi_b} \,\mathrm{d}x = -\int_R u_b \big(\nabla \cdot (\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi_b})\big) \,\mathrm{d}x + \int_{\partial R} u_b \big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi_b} \cdot \mathbf{n}_{\boldsymbol{\partial R}}\big) \,\mathrm{d}\sigma, \tag{3.7a}$$

$$-\int_R \boldsymbol{\sigma_b} \cdot \nabla \varphi_b \,\mathrm{d}x = \int_R g_b \, \varphi_b \,\mathrm{d}x - \int_{\partial R} \big(\boldsymbol{\sigma_b} \cdot \mathbf{n}_{\boldsymbol{\partial R}}\big) \varphi_b \,\mathrm{d}\sigma, \tag{3.7b}$$

by multiplying equations (3.6a) and (3.6b) by some vector-valued test function $\boldsymbol{\psi_b}$ and by some scalar test function φ_b of suitable regularity, respectively, and by subsequent application of the classical integration by parts formula for bulk domains. Here, the set $R \subseteq \Omega$ is an arbitrary, bounded d-dimensional portion of Ω, whose boundary ∂R is a smooth $(d-1)$-dimensional hypersurface, and $\mathbf{n}_{\boldsymbol{\partial R}}$ is the field of outward-pointing unit normal vectors to ∂R.

The resulting weak formulation of our model problem reads as follows. We look for some pair of functions $(u_b, \boldsymbol{\sigma_b}) \in H^1(\Omega) \times [H^1(\Omega)]^d$ which satisfy boundary conditions (3.6c) and (3.6d) in the sense of traces and furthermore have the property that, for every portion $R \subseteq \Omega$ of the type specified above, equations (3.7) hold for all test function pairs $(\varphi_b, \boldsymbol{\psi_b}) \in H^1(\Omega) \times [H^1(\Omega)]^d$.

Meshes and discrete approximation spaces

Given a bulk domain $\mathcal{D}$ in $\mathbb{R}^d$, a *mesh* $\mathcal{T}(\mathcal{D})$ shall be understood as a set of open, disjoint elements $K_0, \ldots, K_{M-1} \subset \mathbb{R}^d$ with

$$\mathrm{cl}\,(\mathcal{D}) = \bigcup\nolimits_{i=0,\ldots,M-1} \mathrm{cl}\,(K_i)\,.$$

We call $\mathcal{T}(\mathcal{D})$ a mesh *of* $\mathcal{D}$ and say that $\mathcal{T}(\mathcal{D})$ *resolves the boundary* $\partial\mathcal{D}$.

To construct finite-dimensional function spaces that are capable of approximating the solution on some discrete reconstruction Ω_h of the geometry Ω, we use a mesh of some bulk domain Ω_h which approximates Ω. This mesh is expected to comprise shape regular elements that are tetrahedra or hexahedra for $d = 3$, and triangles or quadrilaterals for $d = 2$. We denote the mesh by

$\mathcal{T}_h(\Omega_h)$, where h denotes its maximum element size

$$h := \max_{K \in \mathcal{T}_h(\Omega_h)} \operatorname{diam}(K).$$

Due to the properties of the discrete approximation spaces that we are about to associate with the mesh $\mathcal{T}_h(\Omega_h)$, we will consider the set of *internal faces* of $\mathcal{T}_h(\Omega_h)$, which can be defined as

$$\mathcal{E}_h^{\text{int}}(\Omega_h) := \\ \left\{ E = \partial K_E^+ \cap \partial K_E^- \;\middle|\; K_E^+, K_E^- \in \mathcal{T}_h(\Omega_h),\, K_E^+ \neq K_E^-,\, \operatorname{meas}_{\mathbb{R}^{d-1}}(E) > 0 \right\}.$$

The set $\mathcal{E}_h^{\text{int}}(\Omega_h)$ is often called the *internal skeleton* of the mesh. Each internal face $E \in \mathcal{E}_h^{\text{int}}(\Omega_h)$ is the intersection of the boundaries of two elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_h)$. Their boundaries are oriented by two fields $\mathbf{n}_{\boldsymbol{\partial \mathbf{K}_E^+}}, \mathbf{n}_{\boldsymbol{\partial \mathbf{K}_E^-}}$ of outward-pointing unit normal vectors which are opposing each other on E. To each internal face $E \in \mathcal{E}_h^{\text{int}}(\Omega_h)$, we can hence assign a dedicated field of unit vectors normal to E by assigning the names K_E^+, K_E^- to the adjacent elements in a fixed manner and by arbitrarily choosing $\mathbf{n}_{\mathbf{E}} := \mathbf{n}_{\boldsymbol{\partial \mathbf{K}_E^+}}\big|_E$.

In addition to the internal faces of $\mathcal{T}_h(\Omega_h)$, we will also deal with the set of faces that lie on the boundary $\partial\Omega_h$. This set is often referred to as the *external skeleton* of $\mathcal{T}_h(\Omega_h)$. It can be defined as

$$\mathcal{E}_h^{\text{ext}}(\Omega_h) := \left\{ E = \partial K_E^+ \cap \partial\Omega_h \;\middle|\; K_E^+ \in \mathcal{T}_h(\Omega_h),\, \operatorname{meas}_{\mathbb{R}^{d-1}}(E) > 0 \right\}.$$

In view of boundary conditions (3.6c) and (3.6d), we will be particularly interested in its two subsets

$$\mathcal{E}_{h,D}^{\text{ext}}(\Omega_h) := \left\{ E \in \mathcal{E}_h^{\text{ext}}(\Omega_h) \;\middle|\; E \subset \partial\Omega_{h,D} \right\},$$
$$\mathcal{E}_{h,N}^{\text{ext}}(\Omega_h) := \left\{ E \in \mathcal{E}_h^{\text{ext}}(\Omega_h) \;\middle|\; E \subset \partial\Omega_{h,N} \right\}.$$

Each *external face* $E \in \mathcal{E}_h^{\text{ext}}(\Omega_h)$ is part of the boundary of a single element $K_E^+ \in \mathcal{T}_h(\Omega_h)$. The boundary of this element is oriented by a field $\mathbf{n}_{\boldsymbol{\partial \mathbf{K}_E^+}}$ of outward-pointing unit normal vectors. To each external face $E \in \mathcal{E}_h^{\text{ext}}(\Omega_h)$, we can hence assign a dedicated field of unit vectors normal to E by choosing $\mathbf{n}_{\mathbf{E}} := \mathbf{n}_{\boldsymbol{\partial \mathbf{K}_E^+}}\big|_E$. It should be noted that this field $\mathbf{n}_{\mathbf{E}}$ matches the field $\mathbf{n}_{\boldsymbol{\partial \Omega_h}}|_E$.

As discrete approximation spaces, we specifically construct finite element spaces of piecewise polynomial functions over Ω_h given by

$$V_{b,h}(\Omega_h) := \left\{ v_{b,h} \in L^2(\Omega_h) \;\middle|\; v_{b,h}\big|_K \in \mathbb{P}(K)\ \forall K \in \mathcal{T}_h(\Omega_h) \right\},$$
$$\Sigma_{b,h}(\Omega_h) := \left\{ \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}} \in \left[L^2(\Omega_h)\right]^d \;\middle|\; \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}}\big|_K \in \left[\mathbb{P}(K)\right]^d\ \forall K \in \mathcal{T}_h(\Omega_h) \right\}.$$

Here, $\mathbb{P}(K)$ denotes some space of polynomial functions over a mesh element K. Popular choices of $\mathbb{P}(K)$ are the space $P_k(K)$ of polynomial functions

of total degree less than or equal to some $k \in \mathbb{N}$, and the space $Q_k(K)$ of polynomial functions with a degree less than or equal to k in each coordinate direction.

In general, functions in $V_{b,h}(\Omega_h)$ and functions in $\Sigma_{b,h}(\Omega_h)$ are discontinuous and do not take a unique value along the internal skeleton $\mathcal{E}_h^{\text{int}}(\Omega_h)$. Nevertheless, on each internal face $E \in \mathcal{E}_h^{\text{int}}(\Omega_h)$ with adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_h)$ as defined above, a function $v_{b,h} \in V_{b,h}(\Omega_h)$, for instance, has two well-defined traces $v_{b,h}|_{\partial K_E^+}$, $v_{b,h}|_{\partial K_E^-}$. Using these traces, we define the *jump* of a function $v_{b,h}$ on an internal face E as

$$[\![v_{b,h}]\!]\big|_E := v_{b,h}\big|_{\partial K_E^+} - v_{b,h}\big|_{\partial K_E^-}$$

and its *average* as

$$\{v_{b,h}\}\big|_E := \frac{1}{2}\Big(v_{b,h}\big|_{\partial K_E^+} + v_{b,h}\big|_{\partial K_E^-}\Big). \tag{3.8}$$

The definitions of these two operators $[\![\cdot]\!]$ and $\{ \cdot \}$ will not only be used for functions in $V_{b,h}(\Omega_h)$ and for functions in $\Sigma_{b,h}(\Omega_h)$. We will rather use it for all functions that have a reasonable definition on each internal face E of the mesh $\mathcal{T}_h(\Omega_h)$, with two branches $v_{b,h}\big|_{\partial K_E^+}$ and $v_{b,h}\big|_{\partial K_E^-}$ which are associated with the adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_h)$.

For functions in $V_{b,h}(\Omega_h)$, we furthermore define a piecewise variant of the (transposed) classical gradient operator in $\mathbb{R}^d$. Given a function $v_{b,h}$, we first set

$$\nabla_h v_{b,h}\big|_K := \nabla\Big[v_{b,h}\big|_K\Big]$$

for each $K \in \mathcal{T}_h(\Omega_h)$. Subsequently, we extend this definition to the faces in $\mathcal{E}_h^{\text{int}}(\Omega_h) \cup \mathcal{E}_h^{\text{ext}}(\Omega_h)$, i.e. to the whole bulk domain Ω_h and its boundary. More specifically, on each internal face $E \in \mathcal{E}_h^{\text{int}}(\Omega_h)$, we define two branches by evaluating the gradient in the two adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_h)$. Similarly, on each external face $E \in \mathcal{E}_h^{\text{ext}}(\Omega_h)$, we evaluate the gradient in the single adjacent element $K_E^+ \in \mathcal{T}_h(\Omega_h)$. We would like to emphasize that this definition does not permit single-valued evaluations on the internal skeleton $\mathcal{E}_h^{\text{int}}(\Omega_h)$. This is sufficient, though, as long as only the jump and average of terms in $\nabla_h v_{b,h}$ is evaluated on internal faces. The latter will be all we need.

Along the same lines, we also define a piecewise variant $\nabla_h \cdot$ of the classical divergence operator $\nabla \cdot$ in $\mathbb{R}^d$. This piecewise divergence operator can be applied to functions in $\Sigma_{b,h}(\Omega_h)$, for example.

It should be mentioned that the discrete approximation spaces $V_{b,h}(\Omega_h)$ and $\Sigma_{b,h}(\Omega_h)$ are *non-conforming spaces* with respect to the given problem. They do not conform to the given problem in the following sense. While we want to approximate some weak solution $(u_b, \boldsymbol{\sigma_b}) \in H^1(\Omega) \times [H^1(\Omega)]^d$ on the reconstructed geometry Ω_h, our discrete spaces do *not* have the property $V_{b,h}(\Omega_h) \times \Sigma_{b,h}(\Omega_h) \subset H^1(\Omega_h) \times [H^1(\Omega_h)]^d$. The latter property is rather

satisfied in a piecewise fashion. Defining the piecewise Sobolev spaces

$$H^k\big(\mathcal{T}_h(\Omega_h)\big) := \Big\{ v_b \in L^2(\Omega_h) \;\Big|\; v_b\big|_K \in H^k(K)\ \forall K \in \mathcal{T}_h(\Omega_h) \Big\}, \quad k \in \mathbb{N},$$

which are also known as broken Sobolev spaces, we particularly have

$$V_{b,h}(\Omega_h) \times \Sigma_{b,h}(\Omega_h) \subset H^k\big(\mathcal{T}_h(\Omega_h)\big) \times \big[H^k\big(\mathcal{T}_h(\Omega_h)\big)\big]^d$$

for any number k.

Discretization: Flux formulation

Starting from the weak formulation of the model equations, which is given by equations (3.6c), (3.6d) and (3.7), we discretize in three steps. First, we replace the geometry Ω by its discrete reconstruction Ω_h. As a second step, we restrict the set of admissible function pairs by considering only function pairs that are representable using the discrete function spaces $V_{b,h}(\Omega_h)\times\Sigma_{b,h}(\Omega_h)$, restricting the set of admissible portions $R \subseteq \Omega_h$ to the mesh elements $K \in \mathcal{T}_h(\Omega_h)$ at the same time. Finally, we replace the discrete solution variables $u_{b,h}$ and $\boldsymbol{\sigma}_{\mathbf{b,h}}$ on the boundary ∂K of each such portion by special numerical approximations $\hat{u}_{h,\partial K}(u_{b,h})$ and $\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}}(u_{b,h}, \boldsymbol{\sigma}_{\mathbf{b,h}})$. This yields schemes of the following kind.

Scheme 3.2.1 (Flux formulation)**.** *We seek to find a pair of discrete functions $(u_{b,h}, \boldsymbol{\sigma}_{\mathbf{b,h}}) \in V_{b,h}(\Omega_h) \times \Sigma_{b,h}(\Omega_h)$, such that, for every element $K \in \mathcal{T}_h(\Omega_h)$, we have*

$$\begin{aligned}
-\int_K \boldsymbol{\sigma}_{\mathbf{b,h}} \cdot \boldsymbol{\psi}_{\mathbf{b,h}} \,\mathrm{d}x &= -\int_K u_{b,h}\big(\nabla_h \cdot (\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b,h}})\big)\,\mathrm{d}x \\
&\quad + \int_{\partial K} \hat{u}_{h,\partial K}(u_{b,h})\,\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b,h}}\big|_{\partial K} \cdot \mathbf{n}_{\boldsymbol{\partial K}}\big)\,\mathrm{d}\sigma, \\
-\int_K \boldsymbol{\sigma}_{\mathbf{b,h}} \cdot \nabla_h \varphi_{b,h} \,\mathrm{d}x &= \int_K g_b\, \varphi_{b,h} \,\mathrm{d}x \\
&\quad - \int_{\partial K} \big(\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}}(u_{b,h}, \boldsymbol{\sigma}_{\mathbf{b,h}}) \cdot \mathbf{n}_{\boldsymbol{\partial K}}\big)\, \varphi_{b,h}\big|_{\partial K} \,\mathrm{d}\sigma,
\end{aligned}$$

for all $(\varphi_{b,h}, \boldsymbol{\psi}_{\mathbf{b,h}}) \in V_{b,h}(\Omega_h) \times \Sigma_{b,h}(\Omega_h)$.

In the above type of schemes, we assume that two functions

$$\begin{aligned}
\hat{u}_{h,\partial K} &\colon H^1\big(\mathcal{T}_h(\Omega_h)\big) \to L^2(\partial K), \\
\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}} &\colon H^2\big(\mathcal{T}_h(\Omega_h)\big) \times \big[H^1\big(\mathcal{T}_h(\Omega_h)\big)\big]^d \to \big[L^2(\partial K)\big]^d,
\end{aligned}$$

are given for each element $K \in \mathcal{T}_h(\Omega_h)$. We will refer to those two functions as *local numerical fluxes*. They yield approximations to $u_{b,h}$ and $\boldsymbol{\sigma}_{\mathbf{b,h}}$ on the boundary of K, respectively, and are usually defined uniformly throughout the

whole set of mesh elements. To complete the specification of each particular DG method, a collection of specific pairs of local numerical fluxes $\hat{u}_{h,\partial K}$ and $\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}}$ needs to be given, that are expressed in terms of $u_{b,h}$ and $\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}}$, and in terms of the problem's boundary conditions. Scheme 3.2.1 is known as the *flux formulation* of the considered class of DG methods for our model problem.

Discretization: Primal formulation

A typical finite element formulation can be obtained from Scheme 3.2.1 by eliminating the auxiliary vector field $\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}}$ and by combining all local numerical fluxes into global analogues. Starting with the latter, we define two functions

$$\begin{aligned}\hat{u}_h &\colon H^1\big(\mathcal{T}_h(\Omega_h)\big) \to \prod\nolimits_{K\in\mathcal{T}_h(\Omega_h)} L^2(\partial K),\\ \hat{\boldsymbol{\sigma}}_{\mathbf{h}} &\colon H^2\big(\mathcal{T}_h(\Omega_h)\big) \times \big[H^1\big(\mathcal{T}_h(\Omega_h)\big)\big]^d \to \prod\nolimits_{K\in\mathcal{T}_h(\Omega_h)} \big[L^2(\partial K)\big]^d,\end{aligned}$$

which we will refer to as *(global) numerical fluxes*. Both yield functions living on $\mathcal{E}_h^{\mathrm{int}}(\Omega_h) \cup \mathcal{E}_h^{\mathrm{ext}}(\Omega_h)$. Those functions are double-valued on $\mathcal{E}_h^{\mathrm{int}}(\Omega_h)$, and they are single-valued on $\mathcal{E}_h^{\mathrm{ext}}(\Omega_h)$. More specifically, given some argument $v_b \in H^1\big(\mathcal{T}_h(\Omega_h)\big)$, the function $\hat{u}_h(v_b)$, for instance, shall collect the two values

$$\hat{u}_{h,\partial K_E^+}(v_b)\Big|_E \quad \text{and} \quad \hat{u}_{h,\partial K_E^-}(v_b)\Big|_E$$

on each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_h)$, and the single value

$$\hat{u}_{h,\partial K_E^+}(v_b)\Big|_E$$

on each external face $E \in \mathcal{E}_h^{\mathrm{ext}}(\Omega_h)$. Based on the collection of local numerical fluxes $\{\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}}\}_{K\in\mathcal{T}_h(\Omega_h)}$, the numerical flux $\hat{\boldsymbol{\sigma}}_{\mathbf{h}}$ shall be defined analogously.

As the first step in eliminating the auxiliary vector field $\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}}$, we partially revert steps that have been performed to derive the weak formulation of our problem. Using the equality

$$\nabla u_{b,h} \cdot \mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}} = \mathcal{D}_b \nabla u_{b,h} \cdot \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}}$$

and noting that the functions which we are dealing with locally have suitable regularity, application of the classical integration by parts formula for bulk domains in the first equation of Scheme 3.2.1 yields

$$\begin{aligned}-\int_K \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}} \cdot \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}}\,\mathrm{d}x &= \int_K \big(\mathcal{D}_b \nabla_h u_{b,h}\big) \cdot \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}}\,\mathrm{d}x\\ &\quad + \int_{\partial K} \big(\hat{u}_{h,\partial K}(u_{b,h}) - u_{b,h}\big|_{\partial K}\big)\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}}\big|_{\partial K} \cdot \mathbf{n}_{\boldsymbol{\partial K}}\big)\,\mathrm{d}\sigma. \qquad (3.9)\end{aligned}$$

Please note that the latter equation would locally recover a discrete analogue

to equation (3.6a) if the local numerical flux value $\hat{u}_{h,\partial K}(u_{b,h})$ was taken to be exactly the trace of $u_{b,h}$ on ∂K for every mesh element K.

By summing over all mesh elements in equation (3.9) and in the second equation of Scheme 3.2.1, we subsequently obtain

$$\begin{aligned} -\sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)} \int_K \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}} \cdot \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}} \,\mathrm{d}x &= \sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)} \int_K \big(\mathcal{D}_b \nabla_h u_{b,h}\big) \cdot \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}} \,\mathrm{d}x \\ &\quad + \sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)} \int_{\partial K} \big(\hat{u}_{h,\partial K}(u_{b,h}) - u_{b,h}\big|_{\partial K}\big)\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}}\big|_{\partial K} \cdot \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}}\big) \,\mathrm{d}\sigma, \end{aligned} \tag{3.10a}$$

$$\begin{aligned} -\sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)} \int_K \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}} \cdot \nabla_h \varphi_{b,h} \,\mathrm{d}x &= \sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)} \int_K g_b \,\varphi_{b,h} \,\mathrm{d}x \\ &\quad - \sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)} \int_{\partial K} \big(\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial}\mathbf{K}}(u_{b,h}, \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}}) \cdot \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}}\big)\, \varphi_{b,h}\big|_{\partial K} \,\mathrm{d}\sigma. \end{aligned} \tag{3.10b}$$

The last sum in each equation can be rewritten in terms of the faces in the internal skeleton $\mathcal{E}_h^{\mathrm{int}}(\Omega_h)$ and in the external skeleton $\mathcal{E}_h^{\mathrm{ext}}(\Omega_h)$ by means of the jump operator $[\![\cdot]\!]$ and the global numerical fluxes $\hat{u}_h$ and $\hat{\boldsymbol{\sigma}}_{\mathbf{h}}$. Considering equation (3.10a), for instance, we have

$$\begin{aligned} &\sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)} \int_{\partial K} \big(\hat{u}_{h,\partial K}(u_{b,h}) - u_{b,h}\big|_{\partial K}\big)\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}}\big|_{\partial K} \cdot \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}}\big) \,\mathrm{d}\sigma \\ &= \quad \sum\nolimits_{E\in\mathcal{E}_h^{\mathrm{int}}(\Omega_h)} \int_E \big(\hat{u}_{h,\partial K_E^+}(u_{b,h}) - u_{b,h}\big|_{\partial K_E^+}\big)\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}}\big|_{\partial K_E^+} \cdot \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}_\mathbf{E}^+}\big) \,\mathrm{d}\sigma \\ &\qquad\qquad + \int_E \big(\hat{u}_{h,\partial K_E^-}(u_{b,h}) - u_{b,h}\big|_{\partial K_E^-}\big)\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}}\big|_{\partial K_E^-} \cdot \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}_\mathbf{E}^-}\big) \,\mathrm{d}\sigma \\ &\quad + \sum\nolimits_{E\in\mathcal{E}_h^{\mathrm{ext}}(\Omega_h)} \int_E \big(\hat{u}_{h,\partial K_E^+}(u_{b,h}) - u_{b,h}\big|_{\partial K_E^+}\big)\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}}\big|_{\partial K_E^+} \cdot \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}_\mathbf{E}^+}\big) \,\mathrm{d}\sigma \\ &= \quad \sum\nolimits_{E\in\mathcal{E}_h^{\mathrm{int}}(\Omega_h)} \int_E \big[\!\big[\big(\hat{u}_h(u_{b,h}) - u_{b,h}\big)\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}} \cdot \mathbf{n}_{\mathbf{E}}\big)\big]\!\big] \,\mathrm{d}\sigma \\ &\quad + \sum\nolimits_{E\in\mathcal{E}_h^{\mathrm{ext}}(\Omega_h)} \int_E \big(\hat{u}_h(u_{b,h}) - u_{b,h}\big)\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi}_{\mathbf{b},\mathbf{h}} \cdot \mathbf{n}_{\mathbf{E}}\big) \,\mathrm{d}\sigma, \end{aligned} \tag{3.11}$$

where we make use of the facts that $\mathbf{n}_{\mathbf{E}} := \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}_\mathbf{E}^+} = -\mathbf{n}_{\boldsymbol{\partial}\mathbf{K}_\mathbf{E}^-}$ holds on the internal skeleton and that our terms are single-valued on the external skeleton.

Straightforward computation shows the following product rule for the jump operator.

Remark 3.2.2 (Product rule for $[\![\cdot]\!]$)**.** *Given two functions v_b and $\tilde{v}_b$ that are reasonable arguments of the jump operator $[\![\cdot]\!]$, the identity*

$$[\![v_b \cdot \tilde{v}_b]\!] = [\![v_b]\!]\{\tilde{v}_b\} + \{v_b\}[\![\tilde{v}_b]\!]$$

holds on each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_h)$.

Applying this rule to the above identity (3.11) subsequently yields the following reformulation of equation (3.10a):

$$\begin{aligned}
-\sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)}\int_K \boldsymbol{\sigma}_{\mathbf{b,h}}\cdot\boldsymbol{\psi}_{\mathbf{b,h}}\,\mathrm{d}x &= \sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)}\int_K \big(\mathcal{D}_b\nabla_h u_{b,h}\big)\cdot\boldsymbol{\psi}_{\mathbf{b,h}}\,\mathrm{d}x \\
&\quad+\sum\nolimits_{E\in\mathcal{E}_h^{\mathrm{int}}(\Omega_h)}\int_E \big[\![\hat{u}_h(u_{b,h})-u_{b,h}\big]\!\big\{\big(\mathcal{D}_b^{\mathrm{tr}}\boldsymbol{\psi}_{\mathbf{b,h}}\big)\cdot\mathbf{n}_{\mathbf{E}}\big\}\,\mathrm{d}\sigma \\
&\qquad+\int_E \big\{\hat{u}_h(u_{b,h})-u_{b,h}\big\}\big[\!\big[\big(\mathcal{D}_b^{\mathrm{tr}}\boldsymbol{\psi}_{\mathbf{b,h}}\big)\cdot\mathbf{n}_{\mathbf{E}}\big]\!\big]\,\mathrm{d}\sigma \\
&\quad+\sum\nolimits_{E\in\mathcal{E}_h^{\mathrm{ext}}(\Omega_h)}\int_E \big(\hat{u}_h(u_{b,h})-u_{b,h}\big)\big[\big(\mathcal{D}_b^{\mathrm{tr}}\boldsymbol{\psi}_{\mathbf{b,h}}\big)\cdot\mathbf{n}_{\mathbf{E}}\big]\,\mathrm{d}\sigma. \qquad (3.12a)
\end{aligned}$$

By performing the same steps for equation (3.10b), we obtain

$$\begin{aligned}
-\sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)}\int_K \boldsymbol{\sigma}_{\mathbf{b,h}}\cdot\nabla_h\varphi_{b,h}\,\mathrm{d}x &= \sum\nolimits_{K\in\mathcal{T}_h(\Omega_h)}\int_K g_b\,\varphi_{b,h}\,\mathrm{d}x \\
&\quad-\sum\nolimits_{E\in\mathcal{E}_h^{\mathrm{int}}(\Omega_h)}\int_E \big\{\hat{\boldsymbol{\sigma}}_{\mathbf{h}}(u_{b,h},\boldsymbol{\sigma}_{\mathbf{b,h}})\cdot\mathbf{n}_{\mathbf{E}}\big\}\big[\!\big[\varphi_{b,h}\big]\!\big]\,\mathrm{d}\sigma \\
&\qquad+\int_E \big[\!\big[\hat{\boldsymbol{\sigma}}_{\mathbf{h}}(u_{b,h},\boldsymbol{\sigma}_{\mathbf{b,h}})\cdot\mathbf{n}_{\mathbf{E}}\big]\!\big]\{\varphi_{b,h}\}\,\mathrm{d}\sigma \\
&\quad-\sum\nolimits_{E\in\mathcal{E}_h^{\mathrm{ext}}(\Omega_h)}\int_E \big(\hat{\boldsymbol{\sigma}}_{\mathbf{h}}(u_{b,h},\boldsymbol{\sigma}_{\mathbf{b,h}})\cdot\mathbf{n}_{\mathbf{E}}\big)\,\varphi_{b,h}\,\mathrm{d}\sigma. \qquad (3.12b)
\end{aligned}$$

Now, taking equations (3.12) as a basis, the auxiliary vector field $\boldsymbol{\sigma}_{\mathbf{b,h}}$ can be eliminated in two steps.

On the one hand, noting that $\nabla_h u_{b,h}\in\Sigma_{b,h}(\Omega_h)$ and rewriting equation (3.12a) by means of operators which lift functions on $\mathcal{E}_h^{\mathrm{int}}(\Omega_h)$ and functions on $\mathcal{E}_h^{\mathrm{ext}}(\Omega_h)$ in a suitable manner to functions in the discrete space $\Sigma_{b,h}(\Omega_h)$ yields a representation of the auxiliary vector field $\boldsymbol{\sigma}_{\mathbf{b,h}}$ in terms of $u_{b,h}$, cf. Arnold et al. (2002, equation (3.9)):

$$\begin{aligned}
&\boldsymbol{\sigma}_{\mathbf{b,h}}(u_{b,h}) = -\mathcal{D}_b\nabla_h u_{b,h} \\
&\quad+l_1\Big(\big[\!\big[\hat{u}_h(u_{b,h})-u_{b,h}\big]\!\big]\Big)+l_2\Big(\big\{\hat{u}_h(u_{b,h})-u_{b,h}\big\}\Big)+l_3\Big(\hat{u}_h(u_{b,h})-u_{b,h}\Big).
\end{aligned}$$

This representation can be used to evaluate the value $\hat{\boldsymbol{\sigma}}_{\mathbf{h}}(u_{b,h},\boldsymbol{\sigma}_{\mathbf{b,h}})$ of the numerical flux $\hat{\boldsymbol{\sigma}}_{\mathbf{h}}$ if the definitions of the underlying local numerical fluxes make use of the argument $\boldsymbol{\sigma}_{\mathbf{b,h}}$. Since the local numerical fluxes which we are considering in this thesis solely make use of the argument $u_{b,h}$, we do not go into details here.

On the other hand, taking into account that $\nabla_h\varphi_{b,h}\in\Sigma_{b,h}(\Omega_h)$ for all $\varphi_{b,h}\in V_{b,h}(\Omega_h)$, we can take $\boldsymbol{\psi}_{\mathbf{b,h}}:=\nabla_h\varphi_{b,h}$ in equation (3.12a). By doing so and by subsequently combining the resulting identity and equation (3.12b), we arrive at the following formulation which is known as the *primal formulation* of the considered class of DG methods for our model problem.

Scheme 3.2.3 (Primal formulation)**.** *Find a discrete function $u_{b,h} \in V_{b,h}(\Omega_h)$, such that*

$$a_b(u_{b,h}, \varphi_{b,h}) = \int_{\Omega_h} g_b \, \varphi_{b,h} \, \mathrm{d}x$$

for all $\varphi_{b,h} \in V_{b,h}(\Omega_h)$. Here, $a_b \colon V_{b,h}(\Omega_h) \times V_{b,h}(\Omega_h) \to \mathbb{R}$ is a bilinear form that depends on the choice of the numerical fluxes $\hat{u}_h$ and $\hat{\boldsymbol{\sigma}}_{\mathbf{h}}$ (which are given by the choice of the local numerical fluxes $\hat{u}_{h,\partial K}$ and $\hat{\boldsymbol{\sigma}}_{\mathbf{h},\partial\mathbf{K}}$ for every mesh element $K \in \mathcal{T}_h(\Omega_h)$). It is defined by

$$a_b(u_{b,h}, \varphi_{b,h}) := a^{(\hat{u}_h, \hat{\boldsymbol{\sigma}}_{\mathbf{h}})}\big(\Omega_h,\, \mathcal{D}_b,\, u_{b,h},\, \varphi_{b,h}\big),$$

with

$$\begin{aligned}
a^{(\hat{u}_h, \hat{\boldsymbol{\sigma}}_{\mathbf{h}})}\big(\mathcal{D}, \mathcal{D}, u_h, \varphi_h\big) &:= \sum\nolimits_{K \in \mathcal{T}_h(\mathcal{D})} \int_K \big(\mathcal{D}\nabla_h u_h\big) \cdot \nabla_h \varphi_h \, \mathrm{d}x \\
&+ \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})} \int_E [\![\hat{u}_h - u_h]\!] \big\{ \big(\mathcal{D}^{\mathrm{tr}} \nabla_h \varphi_h\big) \cdot \mathbf{n}_{\mathbf{E}} \big\} + \big\{ \hat{\boldsymbol{\sigma}}_{\mathbf{h}} \cdot \mathbf{n}_{\mathbf{E}} \big\} [\![\varphi_h]\!] \, \mathrm{d}\sigma \\
&+ \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})} \int_E \{\hat{u}_h - u_h\} \big[\!\big[\big(\mathcal{D}^{\mathrm{tr}} \nabla_h \varphi_h\big) \cdot \mathbf{n}_{\mathbf{E}} \big]\!\big] + \big[\!\big[\hat{\boldsymbol{\sigma}}_{\mathbf{h}} \cdot \mathbf{n}_{\mathbf{E}} \big]\!\big] \{\varphi_h\} \, \mathrm{d}\sigma \\
&+ \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{ext}}(\mathcal{D})} \int_E (\hat{u}_h - u_h) \big[\big(\mathcal{D}^{\mathrm{tr}} \nabla_h \varphi_h\big) \cdot \mathbf{n}_{\mathbf{E}} \big] + \big(\hat{\boldsymbol{\sigma}}_{\mathbf{h}} \cdot \mathbf{n}_{\mathbf{E}} \big) \, \varphi_h \, \mathrm{d}\sigma.
\end{aligned}$$

Here, we write $\hat{u}_h = \hat{u}_h(u_h)$ and $\hat{\boldsymbol{\sigma}}_{\mathbf{h}} = \hat{\boldsymbol{\sigma}}_{\mathbf{h}}\big(u_h, \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}}(u_h)\big)$ in the integrands to shorten notation. A representation of $\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}}$ in terms of $u_{b,h}$ is available, but it will not be required within the scope of this thesis.

3.2.2. The classical SIPG formulation and related approaches

Next, we consider specific choices of numerical fluxes and the associated DG methods. Motivated by the work of Rivière and Bastian (2004), we choose local numerical fluxes that depend on two scalar parameters $\epsilon, \gamma \in \mathbb{R}$. Uniformly throughout the whole set of mesh elements $K \in \mathcal{T}_h(\Omega_h)$, we particularly define

$$\hat{u}_{h,\partial K}(u_h)\big|_E := \begin{cases} \{u_h\} + \frac{1+\epsilon}{2} (\mathbf{n}_{\partial\mathbf{K}} \cdot \mathbf{n}_{\mathbf{E}}) [\![u_h]\!] & E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_h), \\ u_h + \epsilon(u_h - g_{b,D}) & E \in \mathcal{E}_{h,D}^{\mathrm{ext}}(\Omega_h), \\ u_h & E \in \mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h), \end{cases}$$

and

$$\hat{\boldsymbol{\sigma}}_{\mathbf{h},\partial\mathbf{K}}(u_h, \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}})\big|_E := \begin{cases} \{-\mathcal{D}_b \nabla_h u_h\} + \frac{\gamma}{h_E} [\![u_h]\!] \, \mathbf{n}_{\mathbf{E}} & E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_h), \\ -\mathcal{D}_b \nabla_h u_h + \frac{\gamma}{h_E} (u_h - g_{b,D}) \, \mathbf{n}_{\mathbf{E}} & E \in \mathcal{E}_{h,D}^{\mathrm{ext}}(\Omega_h), \\ g_{b,N} \, \mathbf{n}_{\mathbf{E}} & E \in \mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h), \end{cases}$$

where we consider arbitrary faces $E \in \mathcal{E}_h^{\text{int}}(\Omega_h) \cup \mathcal{E}_h^{\text{ext}}(\Omega_h)$ with $E \subset \partial K$, and incorporate boundary conditions (3.6c) and (3.6d) by different cases.

On each internal face $E \in \mathcal{E}_h^{\text{int}}(\Omega_h)$, the numerical fluxes $\hat{u}_h$ and $\hat{\boldsymbol{\sigma}}_{\mathbf{h}}$ which result from this choice satisfy

$$\begin{aligned}
[\![\hat{u}_h(u_h) - u_h]\!] &= [\![\hat{u}_h(u_h)]\!] - [\![u_h]\!] = (1+\epsilon)[\![u_h]\!] - [\![u_h]\!] = \epsilon[\![u_h]\!] \\
\{\hat{u}_h(u_h) - u_h\} &= \{\hat{u}_h(u_h)\} - \{u_h\} = \{u_h\} - \{u_h\} = 0 \\
\{\hat{\boldsymbol{\sigma}}_{\mathbf{h}}(u_h, \boldsymbol{\sigma}_{\mathbf{b,h}}) \cdot \mathbf{n}_{\mathbf{E}}\} &= -\{(\mathcal{D}_b \nabla_h u_h) \cdot \mathbf{n}_{\mathbf{E}}\} + \frac{\gamma}{h_E}[\![u_h]\!] \\
[\![\hat{\boldsymbol{\sigma}}_{\mathbf{h}}(u_h, \boldsymbol{\sigma}_{\mathbf{b,h}}) \cdot \mathbf{n}_{\mathbf{E}}]\!] &= 0.
\end{aligned}$$

Therefore, Scheme 3.2.3 takes the following form.

Scheme 3.2.4 (SIPG/NIPG/OBB-DG/IIPG formulation)**.** *Find a discrete function $u_{b,h} \in V_{b,h}(\Omega_h)$, such that*

$$a_b(u_{b,h}, \varphi_{b,h}) = -\sum\nolimits_{E \in \mathcal{E}_{h,N}^{\text{ext}}(\Omega_h)} \int_E g_{b,N}\, \varphi_{b,h}\, \mathrm{d}\sigma + \int_{\Omega_h} g_b\, \varphi_{b,h}\, \mathrm{d}x \tag{3.13}$$

for all $\varphi_{b,h} \in V_{b,h}(\Omega_h)$, where $a_b \colon V_{b,h}(\Omega_h) \times V_{b,h}(\Omega_h) \to \mathbb{R}$ is a bilinear form defined by

$$a_b(u_{b,h}, \varphi_{b,h}) := a^{(\epsilon,\gamma)}\big(\Omega_h,\, \mathcal{D}_b,\, u_{b,h},\, \varphi_{b,h}\big),$$

with

$$\begin{aligned}
a^{(\epsilon,\gamma)}\big(\mathcal{D}, \mathcal{D}, u_h, \varphi_h\big) &:= \sum\nolimits_{K \in \mathcal{T}_h(\mathcal{D})} \int_K \big(\mathcal{D}\nabla_h u_h\big) \cdot \nabla_h \varphi_h \, \mathrm{d}x \\
&+ \sum\nolimits_{E \in \mathcal{E}_h^{\text{int}}(\mathcal{D})} \int_E \epsilon\big\{\big(\mathcal{D}^{\text{tr}} \nabla_h \varphi_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big\}[\![u_h]\!] - \big\{\big(\mathcal{D}\nabla_h u_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big\}[\![\varphi_h]\!] \, \mathrm{d}\sigma \\
&+ \sum\nolimits_{E \in \mathcal{E}_h^{\text{int}}(\mathcal{D})} \frac{\gamma}{h_E} \int_E [\![u_h]\!][\![\varphi_h]\!] \, \mathrm{d}\sigma \\
&+ \sum\nolimits_{E \in \mathcal{E}_{h,D}^{\text{ext}}(\mathcal{D})} \int_E \epsilon\big[\big(\mathcal{D}^{\text{tr}} \nabla_h \varphi_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big](u_h - g_{b,D}) - \big[\big(\mathcal{D}\nabla_h u_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big]\varphi_h \, \mathrm{d}\sigma \\
&+ \sum\nolimits_{E \in \mathcal{E}_{h,D}^{\text{ext}}(\mathcal{D})} \frac{\gamma}{h_E} \int_E (u_h - g_{b,D})\, \varphi_h \, \mathrm{d}\sigma.
\end{aligned}$$

Specific DG methods for model problem (3.5) can be obtained by choosing the parameters ϵ and γ. In this thesis, we refer to such methods as different *DG formulations*, some of which are well-known in the literature. Table 3.1 lists parameter values which yield the classical *symmetric interior penalty Galerkin (SIPG) formulation* (Wheeler, 1978) that is motivated by the work of Douglas and Dupont (1976), the non-symmetric interior penalty Galerkin (NIPG) formulation (Rivière et al., 1999, 2001), the Oden–Babuška–Baumann DG (OBB-DG) formulation (Oden et al., 1998) and the incomplete interior penalty Galerkin (IIPG) formulation (see e.g. Dawson et al., 2004).

Formulation	Symmetry parameter	Penalty parameter
SIPG	$\epsilon := -1$	$\gamma > 0$
NIPG	$\epsilon := 1$	$\gamma > 0$
OBB-DG	$\epsilon := 1$	$\gamma := 0$
IIPG	$\epsilon := 0$	$\gamma > 0$

Table 3.1.: Values of the parameters of Scheme 3.2.4 that correspond to DG formulations known in the literature.

The terms in the bilinear form of the DG formulations given by Scheme 3.2.4 serve different purposes. For the *consistency* of each formulation, the terms of the form

$$\int_K \big(\mathcal{D}\nabla_h u_h\big) \cdot \nabla_h \varphi_h \, \mathrm{d}x,$$

and also the terms of the form

$$\int_E -\big\{\big(\mathcal{D}\nabla_h u_h\big) \cdot \mathbf{n_E}\big\}[\![\varphi_h]\!] \, \mathrm{d}\sigma \quad \text{and} \quad \int_E -\big[\big(\mathcal{D}\nabla_h u_h\big) \cdot \mathbf{n_E}\big]\varphi_h \, \mathrm{d}\sigma$$

are required. Consistency is defined to mean that also a classical solution u_b to model problem (3.5) satisfies

$$a_b(u_b, \varphi_{b,h}) = -\sum\nolimits_{E \in \mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h)} \int_E g_{b,N}\, \varphi_{b,h} \, \mathrm{d}\sigma + \int_{\Omega_h} g_b\, \varphi_{b,h} \, \mathrm{d}x$$

for all $\varphi_{b,h} \in V_{b,h}(\Omega_h)$. In view of equation (3.13), having this property is equivalent to having a property known as Galerkin orthogonality:

$$a_b(u_b - u_{b,h}, \varphi_{b,h}) = 0 \quad \forall \varphi_{b,h} \in V_{b,h}(\Omega_h).$$

The latter identity is very useful in error analysis.

The terms with the parameter ϵ in front are capable of controling the symmetry of the matrix which describes the system of linear equations associated with the discrete problem. They are hence known as *symmetry terms* and, accordingly, the parameter ϵ can be called *symmetry parameter*. A formulation with $\epsilon = -1$ results in a symmetric system matrix if and only if the diffusivity tensor $\mathcal{D}_b$ is symmetric. Formulations with $\epsilon \neq -1$ do not result in symmetric system matrices. However, in formulations with $\epsilon = 1$, such as the NIPG formulation and the OBB-DG formulation, the symmetry terms play a special role. Here, those terms contribute to the stability of the formulation in some sense.

The terms with the parameter γ in front contribute to the stability of the formulation (i.e. to the coercivity of its bilinear form) if $\gamma > 0$. Therefore,

they can be referred to as *stability terms*. They are also commonly known as *(interior/exterior) penalty terms* since they penalize jumps of the solution $u_{b,h}$ on the internal skeleton $\mathcal{E}_h^{\text{int}}(\Omega_h)$ and differences between $u_{b,h}$ and the Dirichlet boundary values $g_{b,D}$ on the external skeleton $\mathcal{E}_h^{\text{ext}}(\Omega_h)$, respectively, when considering parameters in the regime $\gamma > 0$. Accordingly, the parameter γ can be called *penalty parameter*.

The penalization effect of the stability terms can be seen most easily in the special case $\epsilon = -1$, $\mathcal{D}_b^{\text{tr}} = \mathcal{D}_b$, with either $\partial\Omega_D = \emptyset$ or $g_{b,D} = 0$. In this special case, the bilinear form $a_b \colon V_{b,h}(\Omega_h) \times V_{b,h}(\Omega_h) \to \mathbb{R}$ in equation (3.13) is symmetric. As a consequence, Scheme 3.2.4 can be interpreted as the problem of finding a global minimizer $u_{b,h} \in V_{b,h}(\Omega_h)$ of the energy functional

$$\mathfrak{E} \colon V_{b,h}(\Omega_h) \to \mathbb{R}, \quad \mathfrak{E}(v_{b,h}) := \frac{1}{2}\, a_b(v_{b,h}, v_{b,h}) - c_b(v_{b,h}),$$

where $c_b \colon V_{b,h}(\Omega_h) \to \mathbb{R}$ denotes the linear form that is defined by the right-hand side of equation (3.13). It is easy to see that, in the parameter regime $\gamma > 0$, the stability terms in the bilinear form have non-negative contributions to the energy functional $\mathfrak{E}$ in this variational setting, and that those contributions penalize jumps in the manner discussed above.

In this thesis, we will particularly deal with the classical SIPG formulation and with a slight generalization which is introduced next.

3.2.3. The SWIPG formulation

For the classical SIPG formulation, there exists a minimum penalty parameter γ, independent of the mesh size h, which makes the method converge to the solution of the given problem as $h \to 0$. However, this minimum penalty parameter depends on the diffusivity tensor $\mathcal{D}_b$ in a global sense, more precisely on the diffusivity in normal direction on the entire internal skeleton of the mesh. Moreover, the diffusivity can exhibit heterogeneities, such as anisotropies or even discontinuities across internal faces. In this case, it would be beneficial to choose the penalty parameter individually on each face, based on the local properties of the diffusivity tensor. This way, the error in the numerical solution locally would stay as small as possible.

A modification of the classical SIPG formulation which follows this idea has been proposed in Ern et al. (2009), where it is applied to advection–diffusion equations. The advection terms are treated by upwind stabilization (see e.g. Brezzi et al., 2004), which corresponds to using the classical upwind numerical flux as a replacement of the advective flux on the internal faces of $\mathcal{T}_h(\Omega_h)$. Neglecting advection terms and sticking with our model problem (3.5), this modified SIPG formulation, which we will refer to as the *symmetric weighted interior penalty Galerkin (SWIPG) formulation* in this thesis, can be described the following way.

One central step of the modification is to introduce a double-valued, scalar weighting function ω on the internal skeleton $\mathcal{E}_h^{\text{int}}(\Omega_h)$. This weighting function

is then employed to generalize the classical average operator $\{\cdot\}$ which is defined in equation (3.8), and it is used to define a diffusivity-dependent scaling function $\gamma_{\mathcal{D}}$ on $\mathcal{E}_h^{\mathrm{int}}(\Omega_h) \cup \mathcal{E}_{h,D}^{\mathrm{ext}}(\Omega_h)$. The latter is incorporated into the penalty parameter γ as an additional factor.

As a weighting function ω, we generally consider some function which has two branches $\omega|_{\partial K_E^+}, \omega|_{\partial K_E^-} \in \mathbb{R}^{\geq 0}$ on each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_h)$. Those branches shall represent weights that are associated with the two adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_h)$ and have the property $\omega|_{\partial K_E^+} + \omega|_{\partial K_E^-} = 1$. We would like to emphasize that this point of view does not permit single-valued evaluations of ω. This is sufficient, though, since we will always target the two weights $\omega|_{\partial K_E^+}$ and $\omega|_{\partial K_E^-}$ explicitly.

Using the weighting function ω, we define a weighted average operator $\{\cdot\}_\omega$ for functions $v_{b,h}$ which have a reasonable definition on each internal face E of the mesh $\mathcal{T}_h(\Omega_h)$, with two branches $v_{b,h}\big|_{\partial K_E^+}$ and $v_{b,h}\big|_{\partial K_E^-}$ that are associated with the adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_h)$. Given such a function $v_{b,h}$ and some internal face E, we specifically set

$$\big\{v_{b,h}\big\}_\omega\big|_E := \omega|_{\partial K_E^+} \cdot v_{b,h}\big|_{\partial K_E^+} + \omega|_{\partial K_E^-} \cdot v_{b,h}\big|_{\partial K_E^-}. \tag{3.14}$$

Note that the operator $\{\cdot\}_\omega$ which results from the latter definition equals the classical average operator $\{\cdot\}$ for the choice $\omega|_{\partial K_E^+} := \omega|_{\partial K_E^-} := \frac{1}{2}$.

The diffusivity-dependent scaling function $\gamma_{\mathcal{D}}$, is based on a scalar function $\mathcal{d}_{\mathcal{D}}$ which describes the diffusivity in normal direction on $\mathcal{E}_h^{\mathrm{int}}(\Omega_h) \cup \mathcal{E}_{h,D}^{\mathrm{ext}}(\Omega_h)$. The latter is double-valued on $\mathcal{E}_h^{\mathrm{int}}(\Omega_h)$ and single-valued on $\mathcal{E}_{h,D}^{\mathrm{ext}}(\Omega_h)$. More specifically, on each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_h)$, we define the two branches

$$\mathcal{d}_{\mathcal{D}}\big|_{\partial K_E^+} := (\mathbf{n_E})^{\mathrm{tr}}\, \mathcal{D}|_{\partial K_E^+}\, \mathbf{n_E} \quad \text{and} \quad \mathcal{d}_{\mathcal{D}}\big|_{\partial K_E^-} := (\mathbf{n_E})^{\mathrm{tr}}\, \mathcal{D}|_{\partial K_E^-}\, \mathbf{n_E}$$

that are associated with the two adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_h)$. Similarly, on each external face $E \in \mathcal{E}_{h,D}^{\mathrm{ext}}(\Omega_h)$, we define the single branch

$$\mathcal{d}_{\mathcal{D}} := (\mathbf{n_E})^{\mathrm{tr}}\, \mathcal{D}\, \mathbf{n_E},$$

performing evaluations in the single adjacent element $K_E^+ \in \mathcal{T}_h(\Omega_h)$. On each face $E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_h) \cup \mathcal{E}_{h,D}^{\mathrm{ext}}(\Omega_h)$, the scaling function $\gamma_{\mathcal{D}}$ is subsequently defined by

$$\gamma_{\mathcal{D}}\big|_E := \begin{cases} \omega|_{\partial K_E^+}^2 \cdot \mathcal{d}_{\mathcal{D}}\big|_{\partial K_E^+} + \omega|_{\partial K_E^-}^2 \cdot \mathcal{d}_{\mathcal{D}}\big|_{\partial K_E^-} & E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_h), \\ \mathcal{d}_{\mathcal{D}} & E \in \mathcal{E}_{h,D}^{\mathrm{ext}}(\Omega_h). \end{cases}$$

Please note that we intentionally leave $\mathcal{d}_{\mathcal{D}}$ and $\gamma_{\mathcal{D}}$ undefined on $\mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h)$.

By replacing the classical average operator $\{\cdot\}$ which is defined in equation (3.8) entirely (i.e. in Scheme 3.2.3 and in the local numerical fluxes $\hat{u}_{h,\partial K}$ and $\hat{\boldsymbol{\sigma}}_{\mathbf{h},\partial\mathbf{K}}$ that yield Scheme 3.2.4) by the weighted average operator $\{\cdot\}_\omega$ that

is defined in equation (3.14), by entirely replacing the penalty parameter γ by its diffusivity-scaled analogue $\gamma \cdot \gamma_{\mathcal{D}}$ with $\gamma \in \mathbb{R}^{>0}$, and by fixing the symmetry parameter $\epsilon := -1$, we obtain the following scheme.

Scheme 3.2.5 (SWIPG formulation)**.** *Find a discrete function $u_{b,h} \in V_{b,h}(\Omega_h)$, such that*

$$a_b(u_{b,h}, \varphi_{b,h}) = -\sum\nolimits_{E \in \mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h)} \int_E g_{b,N}\, \varphi_{b,h} \,\mathrm{d}\sigma + \int_{\Omega_h} g_b\, \varphi_{b,h} \,\mathrm{d}x \tag{3.15}$$

for all $\varphi_{b,h} \in V_{b,h}(\Omega_h)$, where $a_b \colon V_{b,h}(\Omega_h) \times V_{b,h}(\Omega_h) \to \mathbb{R}$ is a bilinear form defined by

$$a_b(u_{b,h}, \varphi_{b,h}) := a^{\mathrm{swipg}}\big(\Omega_h,\, \mathcal{D}_b,\, u_{b,h},\, \varphi_{b,h},\, \gamma\big),$$

with

$$\begin{aligned}
a^{\mathrm{swipg}}\big(\mathcal{D}, \mathcal{D}, u_h, \varphi_h, \gamma\big) &:= \sum\nolimits_{K \in \mathcal{T}_h(\mathcal{D})} \int_K \big(\mathcal{D}\nabla_h u_h\big) \cdot \nabla_h \varphi_h \,\mathrm{d}x \\
&- \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})} \int_E \big\{\big(\mathcal{D}^{\mathrm{tr}} \nabla_h \varphi_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big\}_\omega \,[\![u_h]\!] + \big\{\big(\mathcal{D}\nabla_h u_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big\}_\omega \,[\![\varphi_h]\!] \,\mathrm{d}\sigma \\
&+ \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})} \frac{\gamma}{h_E} \int_E \gamma_{\mathcal{D}}\, [\![u_h]\!][\![\varphi_h]\!] \,\mathrm{d}\sigma \\
&- \sum\nolimits_{E \in \mathcal{E}_{h,D}^{\mathrm{ext}}(\mathcal{D})} \int_E \big[\big(\mathcal{D}^{\mathrm{tr}} \nabla_h \varphi_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big](u_h - g_{b,D}) + \big[\big(\mathcal{D}\nabla_h u_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big]\varphi_h \,\mathrm{d}\sigma \\
&+ \sum\nolimits_{E \in \mathcal{E}_{h,D}^{\mathrm{ext}}(\mathcal{D})} \frac{\gamma}{h_E} \int_E \gamma_{\mathcal{D}}\, (u_h - g_{b,D})\, \varphi_h \,\mathrm{d}\sigma.
\end{aligned}$$

Regarding the weighting function ω, Ern et al. (2009) specifically set

$$\omega|_{\partial K_E^+} := \frac{\mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^-}}{\mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^+} + \mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^-}} \quad \text{and} \quad \omega|_{\partial K_E^-} := \frac{\mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^+}}{\mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^+} + \mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^-}}.$$

On the internal skeleton, the scaling function $\gamma_{\mathcal{D}}$ then takes the form

$$\begin{aligned}
\gamma_{\mathcal{D}}\big|_E &= \omega|_{\partial K_E^+}\, \mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^+} \\
&= \omega|_{\partial K_E^-}\, \mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^-} = \frac{\mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^+}\, \mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^-}}{\mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^+} + \mathrm{d}_{\mathcal{D}}\big|_{\partial K_E^-}} \qquad \forall E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_h),
\end{aligned}$$

which means that it corresponds to *half the harmonic average* of the normal component of the diffusivity tensor across internal faces. Ern et al. (2009) recommend using this choice of weights whenever the diffusivity exhibits heterogeneities and they show that their SWIPG formulation can have superior properties in this case.

3.2.4. Spatial discretization of parabolic equations

Model problem

Next, let us consider the time-dependent counterpart of model problem (3.5), restricting our considerations to the case $\partial\Omega_D = \emptyset$. The latter case will be the relevant one in the remainder of this thesis. Given some initial values $u_b(\cdot, 0)$, we seek $u_b \colon \Omega \times [0,T] \to \mathbb{R}$ with

$$\partial_t u_b - \nabla \cdot (\mathcal{D}_b \nabla u_b) = g_b \qquad \text{in} \quad \Omega \times (0,T], \tag{3.16a}$$

$$-\mathcal{D}_b \nabla u_b \cdot \mathbf{n}_{\boldsymbol{\partial\Omega}} = g_{b,N} \quad \text{on} \quad \partial\Omega \times (0,T]. \tag{3.16b}$$

This problem of parabolic nature develops from the theory which has been discussed in Section 3.1.2, in particular using the choices $\Omega(t) := \Omega$ and $\mathbf{v}(\cdot, t) := \underline{\mathbf{0}}$. More precisely, by choosing the diffusive flux $\mathbf{q_b} := -\mathcal{D}_b \nabla u_b$ according to Fick's first law of diffusion, we obtain continuity equation (3.16a) for a bulk quantity with concentration u_b. Furthermore, seeking a particular solution with the property that the considered quantity satisfies the outflux $\mathbf{q_b} \cdot \mathbf{n}_{\boldsymbol{\partial\Omega}} = g_{b,N}$ on the boundary of Ω corresponds to supplementing continuity equation (3.16a) with boundary condition (3.16b).

The above derivation via the theory from Section 3.1.2 reveals that continuity equation (3.16a) represents the underlying equivalent conservation law

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \int_R u_b \, \mathrm{d}x &= -\int_{\partial R} \mathbf{q_b} \cdot \mathbf{n}_{\boldsymbol{\partial R}} \, \mathrm{d}\sigma \qquad + \int_R g_b \, \mathrm{d}x \\ &= \int_{\partial R} \mathcal{D}_b \nabla u_b \cdot \mathbf{n}_{\boldsymbol{\partial R}} \, \mathrm{d}\sigma + \int_R g_b \, \mathrm{d}x, \end{aligned} \tag{3.17}$$

which holds for arbitrary portions $R \subseteq \Omega$ (cf. equation (3.3)). Since $R := \Omega$ is an admissible choice, this conservation law and boundary condition (3.16b) imply that a solution u_b to model problem (3.16) satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_\Omega u_b \, \mathrm{d}x = -\int_{\partial\Omega} g_{b,N} \, \mathrm{d}\sigma + \int_\Omega g_b \, \mathrm{d}x. \tag{3.18}$$

In this thesis, properties of the latter kind will be called *global conservation properties*. Accordingly, conservation laws like (3.17) will be also referred to as *local conservation properties*. Due to property (3.18), the amount

$$m(t) := \int_\Omega u_b \, \mathrm{d}x$$

of the considered quantity is an invariant with respect to time if the model parameters are chosen accordingly. This holds true, e.g., for models with $g_b \equiv 0$ and $g_{b,N} \equiv 0$. When modeling masses, for example, the value $m(t)$ describes the system's mass at each fixed time t.

Weak formulation

As with elliptic model problem (3.5), parabolic model problem (3.16) can be written as a system of first order PDEs. We have

$$\boldsymbol{\sigma_b} = -\mathcal{D}_b \nabla u_b \quad \text{in} \quad \Omega \times (0,T], \tag{3.19a}$$

$$\partial_t u_b + \nabla \cdot \boldsymbol{\sigma_b} = g_b \quad \text{in} \quad \Omega \times (0,T], \tag{3.19b}$$

together with the boundary condition

$$\boldsymbol{\sigma_b} \cdot \mathbf{n}_{\boldsymbol{\partial\Omega}} = g_{b,N} \quad \text{on} \quad \partial\Omega \times (0,T]. \tag{3.19c}$$

On this basis, the following weak formulation of problem (3.16) can be derived completely analogous to our way of proceeding in Section 3.2.1. Given initial values $u_b(\cdot,0) \in H^1(\Omega)$, we look for some pair of functions $u_b \colon \Omega \times [0,T] \to \mathbb{R}$ and $\boldsymbol{\sigma_b} \colon \Omega \times [0,T] \to \mathbb{R}^d$ with $u_b(\cdot,t) \in H^1(\Omega)$ and $\boldsymbol{\sigma_b}(\cdot,t) \in [H^1(\Omega)]^d$, which satisfy boundary condition (3.19c) in the sense of traces and furthermore have the property that, for every portion $R \subseteq \Omega$ of the type specified in Section 3.2.1, the equations

$$-\int_R \boldsymbol{\sigma_b} \cdot \boldsymbol{\psi_b} \,\mathrm{d}x = -\int_R u_b\big(\nabla \cdot (\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi_b})\big) \,\mathrm{d}x + \int_{\partial R} u_b\big(\mathcal{D}_b^{\mathrm{tr}} \boldsymbol{\psi_b} \cdot \mathbf{n}_{\boldsymbol{\partial R}}\big) \,\mathrm{d}\sigma,$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_R u_b\, \varphi_b \,\mathrm{d}x\right) - \int_R \boldsymbol{\sigma_b} \cdot \nabla \varphi_b \,\mathrm{d}x = \int_R g_b\, \varphi_b \,\mathrm{d}x - \int_{\partial R} \big(\boldsymbol{\sigma_b} \cdot \mathbf{n}_{\boldsymbol{\partial R}}\big) \varphi_b \,\mathrm{d}\sigma,$$

hold for all test function pairs $(\varphi_b, \boldsymbol{\psi_b}) \in H^1(\Omega) \times [H^1(\Omega)]^d$, and for each $t \in (0,T]$.

Discretization in space

Starting from this weak formulation of problem (3.16), the method of lines (see e.g. Schiesser, 1991) allows us to discretize in space in a way that is completely analogous to the approach for elliptic equations in Section 3.2.1, Section 3.2.2 and Section 3.2.3. This yields the following semidiscretization.

Semidiscretization 3.2.6 (Spatial discretization of model problem (3.16) using the SIPG/NIPG/OBB-DG/IIPG formulation or the SWIPG formulation)**.** *Given some suitable approximation $u_{b,h}(\cdot,0) \in V_{b,h}(\Omega_h)$ of the initial values, we seek a semidiscrete function $u_{b,h} \colon \Omega_h \times [0,T] \to \mathbb{R}$ with $u_{b,h}(\cdot,t) \in V_{b,h}(\Omega_h)$,*

such that for each $t \in (0, T]$

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{\Omega_h} u_{b,h}\,\varphi_{b,h}\,\mathrm{d}x\right) \;+\; a_b(u_{b,h}, \varphi_{b,h}) \\ &\quad = -\sum\nolimits_{E\in\mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h)} \int_E g_{b,N}\,\varphi_{b,h}\,\mathrm{d}\sigma + \int_{\Omega_h} g_b\,\varphi_{b,h}\,\mathrm{d}x \quad \forall \varphi_{b,h} \in V_{b,h}(\Omega_h).\end{aligned} \tag{3.20}$$

Here, $a_b\colon V_{b,h}(\Omega_h) \times V_{b,h}(\Omega_h) \to \mathbb{R}$ *is either the bilinear form that is defined in Scheme 3.2.4 or the bilinear form from Scheme 3.2.5. Please note that, for both choices, the terms on the subset* $\mathcal{E}_{h,D}^{\mathrm{ext}}(\Omega_h)$ *of the external skeleton vanish and hence do not contribute to* $a_b(u_{b,h}, \varphi_{b,h})$*, given that we restrict our considerations to the case* $\partial\Omega_D = \emptyset$*.*

3.2.5. Semidiscrete conservation properties

According to what we discussed in Section 3.2.4, exact solutions to problem (3.16) satisfy global conservation property (3.18) and local conservation property (3.17). We now want to show that Semidiscretization 3.2.6 is constructed in such a way that it recovers semidiscrete analogues to those properties.

Global conservation properties

We have the following result, which is typical not only when employing DG methods for spatial discretization of our parabolic model problem (3.16), but also when FEMs are used that are based on a continuous Galerkin approach.

Theorem 3.2.7 (Semidiscrete solutions – global conservation property)**.** *Each semidiscrete solution* $u_{b,h}\colon \Omega_h \times [0, T] \to \mathbb{R}$ *with* $u_{b,h}(\cdot, t) \in V_{b,h}(\Omega_h)$ *which is obtained using Semidiscretization 3.2.6 satisfies*

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_h} u_{b,h}\,\mathrm{d}x = -\sum\nolimits_{E\in\mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h)} \int_E g_{b,N}\,\mathrm{d}\sigma + \int_{\Omega_h} g_b\,\mathrm{d}x. \tag{3.21}$$

This identity is a semidiscrete analogue to global conservation property (3.18)*. Noting that* $\partial\Omega_{h,N} = \partial\Omega_h$*, we have*

$$\sum\nolimits_{E\in\mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h)} \int_E g_{b,N}\,\mathrm{d}\sigma = \int_{\partial\Omega_h} g_{b,N}\,\mathrm{d}\sigma.$$

Proof. Property (3.21) can be derived in a straightforward way. In particular, the characteristic function $\mathbb{1}_{\Omega_h}$ of Ω_h is contained in the discrete function spaces $V_{b,h}(\Omega_h)$ and hence an admissible test function $\varphi_{b,h}$ in equation (3.20). Moreover, we have $a_b\big(u_{b,h}, \mathbb{1}_{\Omega_h}\big) = 0$ for the bilinear form that is defined in Scheme 3.2.4 and for the bilinear form from Scheme 3.2.5. Therefore, testing with $\varphi_{b,h} = \mathbb{1}_{\Omega_h}$ in equation (3.20) yields property (3.21). □

Local conservation properties

Due to the discontinuity of the discrete function spaces $V_{b,h}(\Omega_h)$ which we are dealing with, also the characteristic functions of smaller portions $R_h \subseteq \Omega_h$ are admissible test functions $\varphi_{b,h}$ in equation (3.20). In fact, the smallest portions that our spaces allow for are the mesh elements $K \in \mathcal{T}_h(\Omega_h)$. Choosing the characteristic function $\mathbb{1}_K$ of an arbitrary mesh element K as a test function in Semidiscretization 3.2.6 yields

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_K u_{b,h}\,\mathrm{d}x = \\ -\left(a_b(u_{b,h},\mathbb{1}_K) + \sum\nolimits_{E\in\mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h),\,E\subset\partial K}\int_E g_{b,N}\,\mathrm{d}\sigma\right) + \int_K g_b\,\mathrm{d}x.\end{aligned}$$

Considering the bilinear form that is defined in Scheme 3.2.4, as well as the corresponding collection of local numerical fluxes $\{\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}}\}_{K\in\mathcal{T}_h(\Omega_h)}$ introduced in Section 3.2.2, the first term on the right-hand side of the latter equation can be rewritten as

$$\begin{aligned}&a_b(u_{b,h},\mathbb{1}_K) + \sum\nolimits_{E\in\mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h),\,E\subset\partial K}\int_E g_{b,N}\,\mathrm{d}\sigma\\
&\quad= \sum\nolimits_{E\in\mathcal{E}_{h}^{\mathrm{int}}(\Omega_h),\,E\subset\partial K}\int_E\Big(-\big\{\big(\mathcal{D}_b\nabla_h u_{b,h}\big)\cdot\mathbf{n}_{\mathbf{E}}\big\} + \frac{\gamma}{h_E}[\![u_{b,h}]\!]\Big)[\![\mathbb{1}_K]\!]\,\mathrm{d}\sigma\\
&\qquad+ \sum\nolimits_{E\in\mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h),\,E\subset\partial K}\int_E g_{b,N}\,\mathrm{d}\sigma\\
&\quad= \sum\nolimits_{E\in\mathcal{E}_{h}^{\mathrm{int}}(\Omega_h),\,E\subset\partial K}\int_E\big(\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}}(u_{b,h},\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}})\cdot\mathbf{n}_{\mathbf{E}}\big)[\![\mathbb{1}_K]\!]\,\mathrm{d}\sigma\\
&\qquad+ \sum\nolimits_{E\in\mathcal{E}_{h,N}^{\mathrm{ext}}(\Omega_h),\,E\subset\partial K}\int_E \hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}}(u_{b,h},\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}})\cdot\mathbf{n}_{\mathbf{E}}\,\mathrm{d}\sigma\\
&\quad= \int_{\partial K}\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}}(u_{b,h},\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}})\cdot\mathbf{n}_{\boldsymbol{\partial K}}\,\mathrm{d}\sigma,\end{aligned}$$

where we use that $\partial\Omega_{h,N} = \partial\Omega_h$. The same result is obtained by analogous proceeding for the bilinear form from Scheme 3.2.5 and its corresponding collection of local numerical fluxes. We hence have proven the following theorem.

Theorem 3.2.8 (Semidiscrete solutions – local conservation property)**.** *Each semidiscrete solution $u_{b,h}\colon \Omega_h\times[0,T]\to\mathbb{R}$ with $u_{b,h}(\cdot,t)\in V_{b,h}(\Omega_h)$ which is obtained using Semidiscretization 3.2.6 satisfies*

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_K u_{b,h}\,\mathrm{d}x = -\int_{\partial K}\hat{\boldsymbol{\sigma}}_{\mathbf{h},\boldsymbol{\partial K}}(u_{b,h},\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}})\cdot\mathbf{n}_{\boldsymbol{\partial K}}\,\mathrm{d}\sigma + \int_K g_b\,\mathrm{d}x \tag{3.22}$$

for all mesh elements $K\in\mathcal{T}_h(\Omega_h)$.

Recalling that the local numerical flux value $\hat{\boldsymbol{\sigma}}_{\mathbf{h},\partial\mathbf{K}}(u_{b,h}, \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}})$ in equation (3.22) is an approximation to the auxiliary vector field $\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}}$ on the boundary of K, and that $\boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}}$ approximates the physical flux $\mathbf{q}_\mathbf{b}$ that is employed in equation (3.17), property (3.22) can be seen as a semidiscrete analogue to local conservation property (3.17). In view of the local numerical fluxes which we are considering in this thesis, it is easy to see that the value $\hat{\boldsymbol{\sigma}}_{\mathbf{h},\partial\mathbf{K}}(u_{b,h}, \boldsymbol{\sigma}_{\mathbf{b},\mathbf{h}})$ converges to $\mathbf{q}_\mathbf{b}$ if the diffusivity tensor $\mathcal{D}_b$ is continuous across internal faces and the DG method converges.

3.3. Implicit geometry description using the level set framework

Next, we consider the geometrical setting of the class of bulk–surface models that has been introduced in Section 1.2. Let $\Omega(t)$ be an evolving bulk domain in $\mathbb{R}^d$ which is bounded by an evolving, potentially complex-shaped, smooth $(d-1)$-dimensional hypersurface $\Gamma(t)$. Let both be observed during a specified time period $[0,T]$. Moreover, let $\boldsymbol{\nu}(\cdot,t)\colon \Gamma(t) \to \mathbb{R}^d$ denote the field of outward-pointing unit normal vectors to $\Gamma(t)$ and let $\mathbf{v}(\cdot,t)\colon \Omega(t)\cup\Gamma(t) \to \mathbb{R}^d$ be a field which describes the material velocity of $\Omega(t)\cup\Gamma(t)$.

Rather than using an explicit representation of $\Omega(t)$ and $\Gamma(t)$ and tracking their evolution from a Lagrangian point of view (e.g. by means of a family of time-dependent local parametrizations), the UDG schemes that we develop in this thesis are based on a framework for geometry description which describes all geometrical entities from an Eulerian point of view. This allows for addressing all geometry-related challenges which have been discussed in Chapter 1, especially those caused by geometrical evolution.

3.3.1. The level set framework

The specific Eulerian framework that we employ is known as the *level set framework*. Without naming the framework this way, its basic concept has been first introduced in Osher and Sethian (1988). In this seminal paper, the concept is applied to evolving hypersurfaces only, but it can be extended to evolving bulk domains in a straightforward manner.

The evolving bulk domain and the evolving hypersurface making up its boundary are embedded in an Eulerian frame of reference, which is represented by some larger, static bulk domain Ω_Φ in $\mathbb{R}^d$ that contains both objects during the whole observation period, i.e. $\Omega(t)\cup\Gamma(t) \subset \Omega_\Phi$ at each fixed time t. On top of this embedding, all information on the geometry and its motion is captured in an implicit way. In the level set framework, the latter feature is implemented by means of a differentiable scalar function Φ that is defined on the space–time domain $\Omega_\Phi \times [0,T]$, or rather by means of its *level sets* defined by

$$\Gamma_l(t) := \Phi(\cdot,t)\big|_{\Omega_\Phi}^{-1}(l) = \big\{\underline{\mathbf{x}} \in \Omega_\Phi \;\big|\; \Phi(\underline{\mathbf{x}},t) = l\big\}, \quad l \in \mathbb{R}, \quad t \in [0,T], \quad (3.23)$$

and by means of a bulk PDE which is associated with Φ.

The level set function, geometry description and the level set equation

In particular, starting with the role of the differentiable scalar function Φ, it is assumed that the hypersurface $\Gamma(t)$ can be identified with one of its level sets. It is furthermore assumed that $\Phi|_{\Omega(t)}$ and $\Phi|_{\bar{\Omega}^c(t)}$ uniformly take values smaller and greater than the function value corresponding to this designated level set, respectively, or the other way round. Here, $\bar{\Omega}^c(t) := \Omega_\Phi \setminus \big(\Omega(t) \cup \Gamma(t)\big)$ denotes the complement of $\Omega(t) \cup \Gamma(t)$ in Ω_Φ at each fixed time t. A prominent choice, which we also make in this thesis, is to assume that $\Gamma(t)$ can be identified with the zero level set $\Gamma_0(t)$ of Φ and that $\Omega(t)$ corresponds to the negative function values of Φ. In terms of level sets only, this yields representations

$$\Gamma(t) = \Gamma_0(t), \quad \Omega(t) = \bigcup\nolimits_{l<0} \Gamma_l(t) \quad \text{and} \quad \Omega_\Phi = \bigcup\nolimits_{l\in\mathbb{R}} \Gamma_l(t), \quad t \in [0,T], \tag{3.24}$$

where the representations of $\Omega(t)$ and Ω_Φ result from the definition in equation (3.23). See Figure 3.2 for illustrations.

In the following, we assume that such a level set description of $\Omega(t) \cup \Gamma(t)$ exists. Points on $\Gamma_0(t)$ move with the material velocity $\mathbf{v_s} := \mathbf{v}|_{\Gamma(t)}$ of $\Gamma(t)$ since $\Gamma(t)$ and $\Gamma_0(t)$ coincide. Moreover, the scalar function Φ is constant on $\Gamma_0(t)$. Therefore, the function Φ solves the surface PDE

$$\partial^\bullet \Phi = 0 \quad \text{on} \quad \Gamma_0(t),$$

where $\partial^\bullet \Phi$ is the material derivative of Φ with respect to $\mathbf{v_s}$, cf. Section 2.6. By using the differentiability assumption on Φ and applying Theorem 2.6.1, this equation can be rewritten as

$$\partial_t \Phi + \mathbf{v_s} \cdot \nabla \Phi = 0 \quad \text{on} \quad \Gamma_0(t).$$

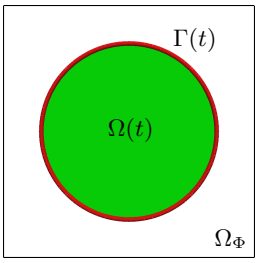
At the same time, representations (3.24) and their underlying assumptions offer a high degree of flexibility in choosing Φ in $\Omega_\Phi \setminus \Gamma(t)$. It is hence reasonable to choose Φ in a way that also the motion of level sets in $\Omega_\Phi \setminus \Gamma(t)$ is characterized by a vanishing material derivative with respect to some given velocity field. Employing a velocity field $\mathbf{v_s^\Phi}$ for this purpose, that is a continuous extension of $\mathbf{v_s}$ to $\Omega_\Phi \times [0,T]$, results in PDEs

$$\partial_t \Phi + \mathbf{v_s^\Phi} \cdot \nabla \Phi = 0 \quad \text{on} \quad \Gamma_l(t),$$

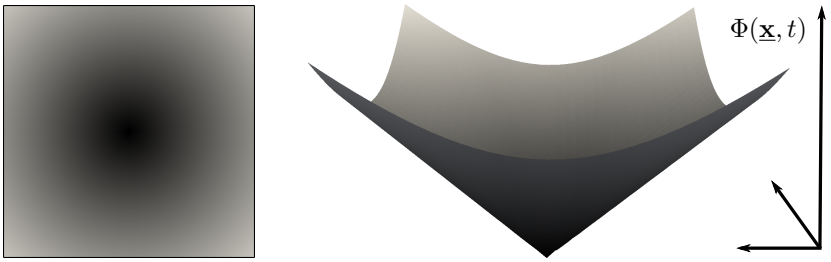
which hold for all level sets $\Gamma_l(t)$ of Φ. According to equation (3.24), the level sets make up a partition of Ω_Φ with $\Omega_\Phi = \bigcup_{l\in\mathbb{R}} \Gamma_l(t)$ at each fixed time t. Therefore, a bulk PDE is obtained which describes the evolution of all level sets simultaneously:

$$\partial_t \Phi + \mathbf{v_s^\Phi} \cdot \nabla \Phi = 0 \quad \text{in} \quad \Omega_\Phi \times (0,T]. \tag{3.25}$$

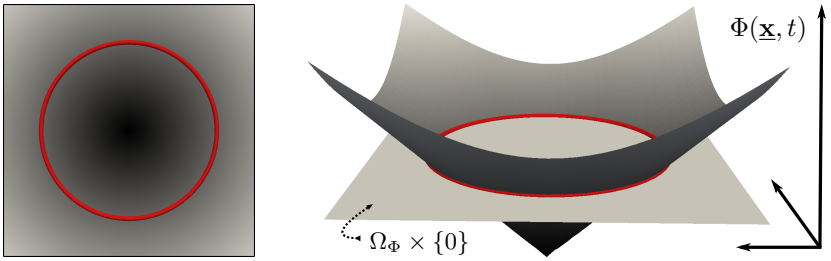
In the context of the level set framework, special names are given to the



(a) Example geometry: A circular bulk domain $\Omega(t)$ in $\mathbb{R}^2$ and its boundary $\Gamma(t)$ at a fixed time t, embedded in some larger, static bulk domain $\Omega_\Phi \subset \mathbb{R}^2$.



(b) A scalar function Φ suitable for describing the example geometry depicted in (a). Left: Function values of Φ visualized by shades of gray (the larger $\Phi(\underline{\mathbf{x}}, t)$, the brighter the gray). Right: Graph of Φ visualized using the same color scheme.



(c) The associated level set description of $\Gamma(t) = \Gamma_0(t)$.

Figure 3.2.: Level set description of some circular example geometry. See Figure 3.3 for detailed information on the pair (Ω_Φ, Φ) and the level sets of Φ.

function Φ and the associated bulk PDE (3.25). More precisely, Φ is known by the name *level set function* since the entire focus is on its level sets and their evolution. For the same reason, equation (3.25) is often referred to as the *level set equation*. Note that this equation is a non-conservative bulk advection equation which corresponds to the prototype (1.7) that has been introduced in Section 1.1.

Practical properties of the level set function and of the level set equation

By its design, the level set function has practical properties. At each point, the gradient of any differentiable scalar function is a vector pointing in the direction of the steepest slope. The level set function Φ, in particular, is constant on each individual level set defined in equation (3.23). Particularly on $\Gamma(t) = \Gamma_0(t)$, it furthermore performs a change of sign in normal direction to the level set. The gradient $\nabla\Phi$ on $\Gamma(t)$ hence points in normal direction to $\Gamma(t)$. More precisely, it is a field of outward-pointing normal vectors to $\Gamma(t)$ since Φ takes negative values in $\Omega(t)$. We therefore have

$$\nabla\Phi \neq 0 \quad \text{on} \quad \Gamma(t),$$

and, using normalization of $\nabla\Phi$, the field $\boldsymbol{\nu}$ of outward-pointing unit normal vectors to $\Gamma(t)$ can be represented as

$$\boldsymbol{\nu} = \left.\frac{\nabla\Phi}{|\nabla\Phi|}\right|_{\Gamma(t)}.$$

For arbitrary level sets of Φ, we have the following generalization.

Theorem 3.3.1 (Properties of arbitrary level sets $\Gamma_l(t)$)**.** *Let t be an arbitrary but fixed point in time and let the level set function Φ (which we assumed to be differentiable in space and time) be a C^k-function with respect to its spatial domain, where $k \in \mathbb{N} \cup \{\infty\}$ is a positive number. Then an arbitrary but fixed level set $\Gamma_l(t) \neq \emptyset$ of Φ has the following property:*

$$\Gamma_l(t) \text{ is a } C^k\text{-hypersurface} \quad \Leftrightarrow \quad \nabla\Phi \neq 0 \text{ on } \Gamma_l(t).$$

Furthermore, if $\Gamma_l(t)$ is a C^k-hypersurface, a field of unit normal vectors to $\Gamma_l(t)$ which provides an orientation for $\Gamma_l(t)$ is given by

$$\left.\frac{\nabla\Phi}{|\nabla\Phi|}\right|_{\Gamma_l(t)}.$$

Given a level set function which is a C^k-function with respect to space, any non-empty level set hence is an orientable C^k-hypersurface if and only if Φ has a non-vanishing gradient on the level set.

Proof. If $\Gamma_l(t)$ is a C^k-hypersurface, it can be seen similar to the above considerations for $\Gamma(t) = \Gamma_0(t)$ that the gradient $\nabla\Phi$ on $\Gamma_l(t)$ points in normal direction to $\Gamma_l(t)$ and hence can not vanish. This proves direction "$\Rightarrow$" of the equivalence. The opposite direction "$\Leftarrow$" directly follows from the definition of C^k-hypersurfaces, see Definition C.1.2. It is strongly related to the implicit function theorem which, together with the assumption of a non-vanishing gradient of Φ, yields local parametrizations of the set $\Gamma_l(t)$ over subsets of $(d-1)$-dimensional Euclidean space.

The second part of the theorem can be seen as follows. Let $\Gamma_l(t)$ be a C^k-hypersurface. Since the gradient $\nabla\Phi$ on $\Gamma_l(t)$ points in normal direction to $\Gamma_l(t)$ and does not vanish, normalization of $\nabla\Phi$ yields a field of unit normal vectors to $\Gamma_l(t)$. This field is continuous and hence provides an orientation for $\Gamma_l(t)$ in the sense of Definition C.1.3. The continuity of the field follows from the fact that Φ is a C^k-function with respect to its spatial domain, such that $\nabla\Phi$ is continuous, particularly on $\Gamma_l(t)$. □

The connection between the level set function Φ and level set equation (3.25) which we have derived above has the following practical implications. Given a suitable extension $\mathbf{v}_\mathbf{s}^\mathbf{\Phi}$ of $\mathbf{v_s}$, an unknown level set function Φ can be obtained by solving equation (3.25) if initial values $\Phi(\cdot, 0)$ are provided. Conversely, considering a given level set function Φ that is at least a C^1-function with respect to space, and an arbitrary but fixed level set $\Gamma_l(t) \neq \emptyset$ with $\nabla\Phi \neq 0$ on $\Gamma_l(t)$, the component of $\mathbf{v}_\mathbf{s}^\mathbf{\Phi}$ normal to $\Gamma_l(t)$ can be obtained using the following equality which can be easily derived from equation (3.25):

$$\mathbf{v}_\mathbf{s}^\mathbf{\Phi} \cdot \frac{\nabla\Phi}{|\nabla\Phi|} = \frac{-\partial_t\Phi}{|\nabla\Phi|} \quad \text{on} \quad \Gamma_l(t).$$

However, note that extracting the component of $\mathbf{v}_\mathbf{s}^\mathbf{\Phi}$ tangential to $\Gamma_l(t)$ is not possible, since function values of Φ do not encode movement of individual points on a level set $\Gamma_l(t)$. Motion which is purely tangential to each level set $\Gamma_l(t)$, for instance, induces a level set function Φ which is constant in time. The latter fact again is reflected by equation (3.25), which reduces to $\partial_t\Phi \equiv 0$ in this case.

Signed distance functions

A frequent choice of level set functions are functions from a class whose members are known by the name *signed distance functions*. On top of providing geometry descriptions, these functions offer information on the distance of each point in the embedding domain Ω_Φ to the closest point on the hypersurface $\Gamma(t)$. More specifically, signed distance functions are level set functions of the form

$$\Phi(\underline{\mathbf{x}}, t) = \begin{cases} 0 & \underline{\mathbf{x}} \in \Gamma(t), \\ -\operatorname{dist}\big(\underline{\mathbf{x}}, \Gamma(t)\big) & \underline{\mathbf{x}} \in \Omega(t), \\ \operatorname{dist}\big(\underline{\mathbf{x}}, \Gamma(t)\big) & \underline{\mathbf{x}} \in \bar{\Omega}^\mathrm{c}(t). \end{cases}$$

Here,

$$\operatorname{dist}\big(\underline{\mathbf{x}}, \Gamma(t)\big) := \inf_{\underline{\mathbf{y}} \in \Gamma(t)} |\underline{\mathbf{x}} - \underline{\mathbf{y}}|$$

denotes the Euclidean distance of a point $\underline{\mathbf{x}} \in \Omega_\Phi$ to the closest point on $\Gamma(t)$, and $\bar{\Omega}^\mathrm{c}(t)$ again is the complement of $\Omega(t) \cup \Gamma(t)$ in Ω_Φ at each fixed time t.

Whether or not a level set function is a signed distance function is characterized by the extra condition

$$|\nabla \Phi| \equiv 1,$$

which holds at least in some space–time neighborhood of $\Gamma(t)$ in case of a signed distance function (cf. Osher and Fedkiw, 2003, Chapter 2). From a theoretical point of view, this simplifies formulations, as we have seen in Theorem 2.3.2.

The approaches presented in this thesis do not require this property. They hence dispense with the need for constructing a signed distance function Φ or constructing initial values $\Phi(\cdot, 0)$ with the signed distance property and keeping this property while solving equation (3.25). For keeping the signed distance property, an appropriate extension $\mathbf{v}_\mathbf{s}^\mathbf{\Phi}$ of $\mathbf{v_s}$ is necessary, or at least some periodic redistancing mechanism needs to be employed. Redistancing is only feasible if it is not crucial that equation (3.25) and the signed distance property hold exactly throughout the calculation.

Although not being required, signed distance functions can be beneficial also for the approaches presented in this thesis. In particular, it is known that signed distance functions can result in methods which avoid stiffness and yield more accurate numerical solutions.

Final remarks

The level set framework allows for an elegant treatment of complex geometrical morphologies with potential topology changes in a fully implicit way. Any geometrical complexity is encoded in terms of function values of the level set function Φ and related properties which are easy to handle.

As we will see in Chapter 4, discrete counterparts to the level set function Φ can be defined by constructing some discrete approximation space which lives on a fixed mesh of Ω_Φ, and by subsequently considering approximations of Φ in this space. If Φ is known a priori, it can be projected into the approximation space at each fixed time. Otherwise, numerical methods can be applied which discretize level set equation (3.25) and look for discrete solutions in the approximation space. To obtain discrete inital values which are required in the latter case, standard techniques for computing numerical approximations of signed distance functions can be employed, such as the fast marching method (see e.g. Osher and Fedkiw, 2003, Chapter 7). In view of this discrete setting, it is convenient to choose an embedding domain Ω_Φ which allows to use a simple Cartesian mesh.

Alternative, more exhaustive introductions to the level set framework are presented, e.g., in Osher and Fedkiw (2003) and in Sethian (1999). Both books discuss, inter alia, numerical methods for dealing with level set equation (3.25), employing computational techniques for hyperbolic bulk PDEs. Within the scope of this thesis, we will always assume the level set function Φ to be known and will treat level set equation (3.25) solely from a theoretical point

of view. In practical applications, however, equation (3.25) frequently needs to be considered as an additional model equation and the UDG schemes which will be introduced in Chapter 4 and Chapter 5 need to be supplemented with an appropriate solver dealing with this equation.

3.3.2. Individual assumptions and definitions in this thesis

In this thesis, we assume that the geometry $\Omega(t) \cup \Gamma(t)$ can be described via representations (3.24) using a level set function

$$\Phi\colon\ \mathrm{cl}\,(\Omega_\Phi) \times [0,T] \to \mathbb{R}$$

whose spatial domain is the closure of some bulk domain Ω_Φ in $\mathbb{R}^d$. It should be noted in this context that, although we require the function Φ to be defined on $\mathrm{cl}\,(\Omega_\Phi) \times [0,T]$ rather than on $\Omega_\Phi \times [0,T]$, everything which has been presented in Section 3.3.1 applies without changes. In equation (3.23), the level sets $\Gamma_l(t)$ of Φ have been defined solely considering the restriction $\Phi(\cdot,t)\big|_{\Omega_\Phi}$.

We assume that Φ is twice continuously differentiable with respect to space and require that it satisfies the non-degeneracy condition

$$\nabla\Phi \neq 0 \quad \text{in} \quad \mathrm{cl}\,(\Omega_\Phi) \times [0,T].$$

According to Theorem 3.3.1, the level sets of Φ have important properties in this case. At each fixed time t, every non-empty level set $\Gamma_l(t)$ is a C^2-hypersurface. Moreover, the field of unit normal vectors to $\Gamma_l(t)$ given by

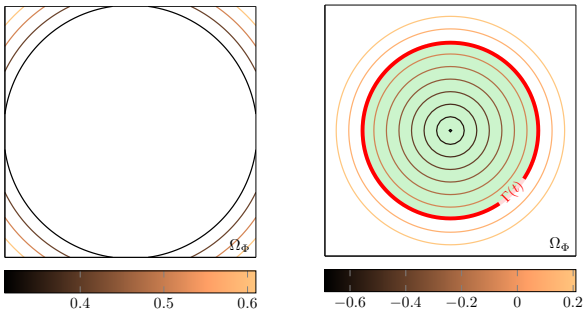
$$\left.\frac{\nabla\Phi}{|\nabla\Phi|}\right|_{\Gamma_l(t)} \tag{3.26}$$

provides an orientation for $\Gamma_l(t)$. Being a C^2-hypersurface, each such level set can be considered as sufficiently smooth for applying the theory from Chapter 2 and the theory from Section 3.1.1, with $\mathcal{M} = \Gamma_l(t)$ or $\mathcal{M} \subset \Gamma_l(t)$. Please refer to our considerations in Appendix C.2 in this respect. In the remainder of this thesis, we will treat the level sets of interest and subsets of those level sets as smooth $(d-1)$-dimensional hypersurfaces embedded in $\mathbb{R}^d$, which are oriented by fields like the one given in equation (3.26).

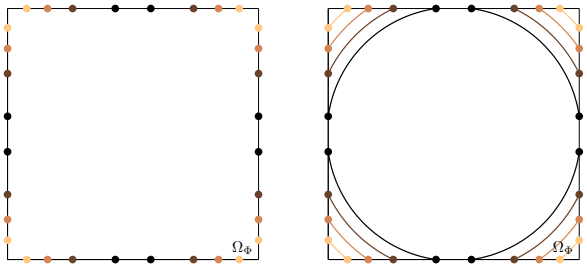
Open level sets, closed level sets, boundaries and closures

Whether or not a level set is an open hypersurface or a closed hypersurface can be characterized by sets which we will define next. For each $t \in [0,T]$ and each non-empty level set $\Gamma_l(t)$, we define a *boundary* $\partial\Gamma_l(t)$ and a *closure* $\mathrm{cl}\left(\Gamma_l(t)\right)$ by

$$\partial\Gamma_l(t) := \left\{\underline{\mathbf{x}} \in \partial\Omega_\Phi \;\middle|\; \Phi(\underline{\mathbf{x}},t) = l\right\} \subset \partial\Omega_\Phi, \quad \mathrm{cl}\left(\Gamma_l(t)\right) := \Gamma_l(t) \cup \partial\Gamma_l(t).$$



(a) Some open level sets of Φ (left) and some of its closed level sets (right). $\Gamma(t) = \Gamma_0(t)$ is depicted in red, the associated bulk domain $\Omega(t)$ is depicted in green. Level sets $\Gamma_l(t)$ with $l \neq 0$ are colored according to the value l and its associated color from the two color bars at the bottom.



(b) Boundaries $\partial\Gamma_l(t)$ (left) and closures $\mathrm{cl}\left(\Gamma_l(t)\right)$ (right) of the open level sets $\Gamma_l(t)$ shown in the left picture of (a). The coloring is the same as in the left picture of (a).

Figure 3.3.: Different types of level sets, together with sets which characterize these types. To illustrate the geometrical setting, we use a level set function Φ which is constant in time, with $\Omega_\Phi := (-1, 1)^2$ (depicted in black) and $\Phi(\underline{\mathbf{x}}, t) := |\underline{\mathbf{x}}| - 0.70$. This pair (Ω_Φ, Φ) is the same as the one depicted in Figure 3.2b.

If and only if $\partial\Gamma_l(t) \neq \emptyset$, a point $\underline{\mathbf{x}} \in \partial\Omega_\Phi$ exists with $\Phi(\underline{\mathbf{x}}, t) = l$. In this case, the level set $\Gamma_l(t)$ is an open hypersurface and we call $\Gamma_l(t)$ an *open level set*. Otherwise, if $\partial\Gamma_l(t) = \emptyset$, we call $\Gamma_l(t)$ a *closed level set* since it is a closed hypersurface. See Figure 3.3 for illustrations.

With these definitions, we have $\mathrm{cl}\left(\Gamma_l(t)\right) = \Gamma_l(t)$ exactly for closed level sets. In addition, $\Gamma_0(t) = \Gamma(t)$ is a closed level set at each fixed time t according to our assumption that $\Gamma(t) \subset \Omega_\Phi$. Note that this is consistent with the fact that $\Gamma(t)$ is a closed hypersurface, given that it is the boundary of an evolving bulk domain $\Omega(t)$.

4. Unfitted DG schemes for coupled bulk–surface PDEs on complex static geometries

In this chapter, we construct and analyze a new type of UDG schemes for bulk–surface models on static geometries. These schemes are specially designed for models comprising continuity equations, together with geometries of arbitrarily complex shape that are represented using the level set framework. The level set description of the geometry is exploited to obtain extensions of the models' surface PDEs. These extensions are additional bulk PDEs which can be treated numerically in a similar way as the native bulk PDEs of the model. Employing the UDG method together with host DG formulations that are adapted to the specific needs of the bulk PDEs provides the general benefits of DG approaches. Moreover, and equally important, it allows us to obtain efficient numerical schemes, even though the model's native bulk domains and the bulk domains which are associated with the level set extensions of the surface PDEs potentially exhibit complex geometrical shapes. We show how schemes can be constructed that are numerically robust and recover discrete analogues to the original conservation properties that are embedded in models comprising continuity equations. Numerical investigation of our schemes indicates good convergence properties, as well as well-posedness and well-conditionedness of the corresponding discrete problems.

We begin in Section 4.1 by deriving those classes of bulk–surface models that are considered for presenting our numerical schemes, and by discussing their associated conservation properties. Section 4.2 focusses on the numerical schemes themselves. Step by step, we develop level set extensions of the models' surface PDEs, aiming for extensions with beneficial properties, and we describe the actual UDG discretization in space which is applied on top of the extension process. Subsequently, we present two approaches which remedy deficiencies resulting from the nature of extension-based discretizations. The first approach allows for recovering original conservation properties, while the other one increases numerical robustness. Finally, we discuss discretization in time and introduce fully discrete schemes for time-dependent bulk–surface models. The proposed schemes are investigated numerically in Section 4.3 using specific bulk–surface models with a known analytical solution and practical applications which involve equations of the same type. In Section 4.4, we summarize our findings and discuss future perspectives.

The content of this chapter is joint work with Christian Engwer (WWU Münster). In large part, it can also be found in Engwer and Westerheide (2018). Preliminary work includes Engwer and Westerheide (2014).

4.1. Classes of static geometry model problems

To present and investigate the numerical schemes that we develop in this chapter, classes of bulk–surface models are considered which are similar to the static geometry special case $\mathbf{v} \equiv \underline{\mathbf{0}}$ of the class of bulk–surface models from Section 1.2. In the course of this, a given static geometry is assumed which is represented by means of the level set framework that has been described in Section 3.3. In this light, let us briefly recall the entities of the setting from Section 1.2 and the entities of the level set framework, while dropping the time-dependency in the notations to some extent. The following entities and notations are used throughout this chapter.

Let Ω be a static bulk domain in $\mathbb{R}^d$ that is bounded by a potentially complex-shaped, smooth $(d-1)$-dimensional hypersurface Γ. Let both be represented by a level set function $\Phi\colon \operatorname{cl}(\Omega_\Phi) \to \mathbb{R}$ which is defined on the closure of some larger bulk domain $\Omega_\Phi \subset \mathbb{R}^d$ that contains $\Omega \cup \Gamma$. Moreover, let $\boldsymbol{\nu}\colon \Gamma \to \mathbb{R}^d$ denote the field of outward-pointing unit normal vectors to Γ. Analogous to the time-dependent definitions from Section 3.3, let Γ_l, $\partial\Gamma_l$ and $\operatorname{cl}(\Gamma_l)$ with $l \in \mathbb{R}$ be the level sets of Φ, their boundaries and their closures, respectively, and assume $\Gamma = \Gamma_0$. Furthermore, let u_b and u_s denote the concentrations of a scalar bulk quantity in Ω and a surface-bound scalar quantity on Γ, let $\mathcal{D}_b$ and $\mathcal{D}_s$ be bulk and surface diffusivity tensors, and let $f_b(u_b)$, $f_s(u_s)$, $f_{b,s}(u_b, u_s)$ and $f_{s,b}(u_b, u_s)$ be source/sink densities in Ω or on Γ, respectively.

4.1.1. A class of parabolic model problems

Two specific classes of model problems are considered in this chapter. The first one is the class of parabolic model problems that exactly matches the static geometry special case $\mathbf{v} \equiv \underline{\mathbf{0}}$ of the class of bulk–surface models which has been introduced in Section 1.2. Given some initial values $u_b(\cdot, 0)$ and $u_s(\cdot, 0)$, we seek $u_b\colon \Omega \times [0,T] \to \mathbb{R}$ and $u_s\colon \Gamma \times [0,T] \to \mathbb{R}$ with

$$\partial_t u_b - \nabla \cdot (\mathcal{D}_b \nabla u_b) = f_b(u_b) \qquad \text{in } \Omega \times (0,T], \tag{4.1a}$$

$$-\mathcal{D}_b \nabla u_b \cdot \boldsymbol{\nu} = -f_{b,s}(u_b, u_s) \qquad \text{on } \Gamma \times (0,T], \tag{4.1b}$$

$$\partial_t u_s - \nabla_\Gamma \cdot (\mathcal{D}_s \nabla_\Gamma u_s) = f_{s,b}(u_b, u_s) + f_s(u_s) \qquad \text{on } \Gamma \times (0,T]. \tag{4.1c}$$

This class of model problems and its components already have been discussed in Section 1.2. Please refer to the explanations on the corresponding formulation for evolving geometries, which is represented by equations (1.8). However, we did not provide a mathematically rigorous derivation of any of the two formulations so far. We catch up on this now.

Model equations of the form (4.1) develop from the theory which has been discussed in Section 3.1, particularly from the theory in Section 3.1.1 choosing $\mathcal{M}(t) := \Gamma$ and $\mathbf{v}_{\boldsymbol{\mathcal{M}}}(\cdot, t) := \underline{\mathbf{0}}$, and from the theory in Section 3.1.2 using the choices $\Omega(t) := \Omega$ and $\mathbf{v}(\cdot, t) := \underline{\mathbf{0}}$. More precisely, by choosing the tangential diffusive surface flux $\mathbf{q_s} := -\mathcal{D}_s \nabla_\Gamma u_s$ according to Fick's first law of diffusion, and by choosing the source/sink density $g_s := f_{s,b}(u_b, u_s) + f_s(u_s)$, we get continuity equation (4.1c) for the surface-bound quantity with concentration u_s. Similarly, choosing the diffusive flux $\mathbf{q_b} := -\mathcal{D}_b \nabla u_b$ and the source/sink density $g_b := f_b(u_b)$ for the bulk quantity with concentration u_b leads to continuity equation (4.1a). Last but not least, seeking a particular solution whose bulk quantity satisfies the outflux $\mathbf{q_b} \cdot \boldsymbol{\nu} = -f_{b,s}(u_b, u_s)$ on the boundary of Ω corresponds to supplementing continuity equation (4.1a) with boundary condition (4.1b).

The above derivation via the theory from Section 3.1 reveals that continuity equations (4.1a) and (4.1c) represent underlying equivalent conservation laws

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_R u_b \, \mathrm{d}x = \int_{\partial R} \mathcal{D}_b \nabla u_b \cdot \mathbf{n}_{\boldsymbol{\partial R}} \, \mathrm{d}\sigma \quad + \int_R f_b(u_b) \, \mathrm{d}x, \tag{4.2a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_M u_s \, \mathrm{d}\sigma = \int_{\partial M} \mathcal{D}_s \nabla_\Gamma u_s \cdot \boldsymbol{\mu}_{\boldsymbol{\partial M}} \, \mathrm{d}\varsigma + \int_M f_{s,b}(u_b, u_s) + f_s(u_s) \, \mathrm{d}\sigma, \tag{4.2b}$$

which hold for arbitrary portions $R \subseteq \Omega$ and $M \subseteq \Gamma$ (cf. equation (3.3) and equation (3.1), respectively). Since $R := \Omega$ and $M := \Gamma$ are admissible choices and since $\partial\Gamma = \emptyset$, these conservation laws and boundary condition (4.1b) imply that solutions (u_b, u_s) of model problems from class (4.1) satisfy global conservation properties

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_\Omega u_b \, \mathrm{d}x = \int_\Gamma f_{b,s}(u_b, u_s) \, \mathrm{d}\sigma + \int_\Omega f_b(u_b) \, \mathrm{d}x, \tag{4.3a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_\Gamma u_s \, \mathrm{d}\sigma = \int_\Gamma f_{s,b}(u_b, u_s) \, \mathrm{d}\sigma + \int_\Gamma f_s(u_s) \, \mathrm{d}\sigma. \tag{4.3b}$$

Therefore, the derivative of the total amount

$$m(t) := \int_\Omega u_b \, \mathrm{d}x + \int_\Gamma u_s \, \mathrm{d}\sigma$$

of the system's quantities fulfills

$$\frac{\mathrm{d}}{\mathrm{d}t} m(t) = \int_\Gamma f_{b,s}(u_b, u_s) + f_{s,b}(u_b, u_s) \, \mathrm{d}\sigma + \int_\Omega f_b(u_b) \, \mathrm{d}x + \int_\Gamma f_s(u_s) \, \mathrm{d}\sigma. \tag{4.4}$$

If the model parameters are chosen accordingly, the value $m(t)$ hence is an invariant with respect to time. This holds true, e.g., for models with $f_b \equiv 0$, $f_s \equiv 0$ and $f_{b,s} = -f_{s,b}$. When modeling masses, for example, the value $m(t)$ describes the system's total mass at each fixed time t.

4.1.2. A class of elliptic model problems

The second class of model problems which we consider in this chapter is the elliptic steady-state counterpart of the class of parabolic model problems given above. We wish to find $u_b\colon \Omega \to \mathbb{R}$ and $u_s\colon \Gamma \to \mathbb{R}$ with

$$-\nabla \cdot (\mathcal{D}_b \nabla u_b) = f_b(u_b) \qquad \text{in } \Omega, \tag{4.5a}$$
$$-\mathcal{D}_b \nabla u_b \cdot \boldsymbol{\nu} = -f_{b,s}(u_b, u_s) \qquad \text{on } \Gamma, \tag{4.5b}$$
$$-\nabla_\Gamma \cdot (\mathcal{D}_s \nabla_\Gamma u_s) = f_{s,b}(u_b, u_s) + f_s(u_s) \qquad \text{on } \Gamma. \tag{4.5c}$$

This class of model problems results from stationary analogues to the general conservation laws (3.3) and (3.1) that have been presented in Section 3.1. With the choices made for deriving system (4.1), these analogues lead to stationary conservation laws

$$0 = \int_{\partial R} \mathcal{D}_b \nabla u_b \cdot \mathbf{n}_{\partial \mathbf{R}} \, \mathrm{d}\sigma \quad + \int_R f_b(u_b) \, \mathrm{d}x, \tag{4.6a}$$
$$0 = \int_{\partial M} \mathcal{D}_s \nabla_\Gamma u_s \cdot \boldsymbol{\mu}_{\partial \mathbf{M}} \, \mathrm{d}\varsigma + \int_M f_{s,b}(u_b, u_s) + f_s(u_s) \, \mathrm{d}\sigma, \tag{4.6b}$$

which hold for arbitrary portions $R \subseteq \Omega$ and $M \subseteq \Gamma$. Note that those laws are stationary analogues to conservation laws (4.2a) and (4.2b), respectively. Given suitable regularity assumptions, it can be shown that stationary conservation law (4.6a) and continuity equation (4.5a) are equivalent, as well as stationary conservation law (4.6b) and continuity equation (4.5c). The procedure is similar to the one described in Section 3.1.

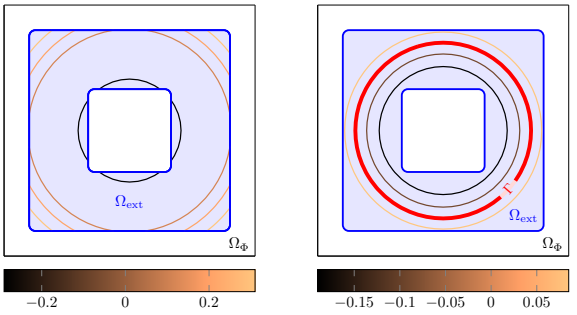
Using the same arguments as in Section 4.1.1, it follows from stationary conservation laws (4.6a) and (4.6b) and from boundary condition (4.5b) that solutions (u_b, u_s) of model problems from class (4.5) satisfy global conservation properties

$$0 = \int_\Gamma f_{b,s}(u_b, u_s) \, \mathrm{d}\sigma + \int_\Omega f_b(u_b) \, \mathrm{d}x, \tag{4.7a}$$
$$0 = \int_\Gamma f_{s,b}(u_b, u_s) \, \mathrm{d}\sigma + \int_\Gamma f_s(u_s) \, \mathrm{d}\sigma. \tag{4.7b}$$

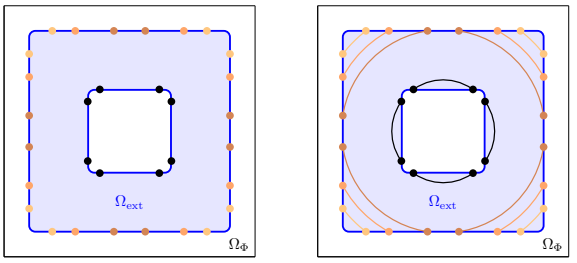
4.2. The approaches and corresponding schemes

4.2.1. An extension process for surface equations

Our schemes are based on extending the models' surface equations to some d-dimensional neighborhood of the considered $(d-1)$-dimensional hypersurface Γ. We particularly perform an extension to a bulk domain Ω_{ext} in $\mathbb{R}^d$ with $\Gamma \subset \Omega_{\text{ext}} \subseteq \Omega_\Phi$. In the course of this extension process, we need the surface differential operators from Chapter 2 for fields on arbitrary but fixed elements



(a) Some open hypersurfaces in the set $\mathcal{S}(\Phi, \Omega_{\text{ext}})$ (left) and some closed hypersurfaces in $\mathcal{S}(\Phi, \Omega_{\text{ext}})$ (right). $\Gamma = \Gamma_0 = \mathcal{M}_0$ is depicted in red. Hypersurfaces $\mathcal{M}_l$ with $l \neq 0$ are colored according to the value l and its associated color from the two color bars at the bottom.



(b) Boundaries $\partial\mathcal{M}_l$ (left) and closures $\text{cl}(\mathcal{M}_l)$ (right) of the open hypersurfaces $\mathcal{M}_l \in \mathcal{S}(\Phi, \Omega_{\text{ext}})$ shown in the left picture of (a). The coloring is the same as in the left picture of (a).

Figure 4.1.: Hypersurfaces $\mathcal{M}_l \in \mathcal{S}(\Phi, \Omega_{\text{ext}})$ employed in the extension process from Section 4.2.1. To illustrate the geometrical setting, we use the level set function Φ with $\Omega_\Phi := (-1, 1)^2$ (depicted in black) and $\Phi(\underline{\mathbf{x}}) := |\underline{\mathbf{x}}| - 0.70$, together with the surface extension domain $\Omega_{\text{ext}} := (-0.8, 0.8)^2 \setminus [-0.33, 0.33]^2$ (depicted in blue).

in the set

$$\mathcal{S}(\Phi, \Omega_{\text{ext}}) := \left\{ \mathcal{M}_l := \Gamma_l \cap \Omega_{\text{ext}} \;\middle|\; l \in \mathbb{R},\, \Gamma_l \cap \Omega_{\text{ext}} \neq \emptyset \right\}.$$

The latter comprises all non-empty intersections between level sets of Φ and the surface extension domain Ω_{ext}. Most notably, it contains $\Gamma = \Gamma_0 = \mathcal{M}_0$. See Figure 4.1 for illustrations.

As described in Section 3.3.2, elements in $\mathcal{S}(\Phi, \Omega_{\text{ext}})$ can be considered as orientable, smooth $(d-1)$-dimensional hypersurfaces embedded in $\mathbb{R}^d$. Therefore, the theory from Chapter 2 and the theory from Section 3.1.1 can be applied for each $\mathcal{M}_l \in \mathcal{S}(\Phi, \Omega_{\text{ext}})$, using the choices $\mathcal{M}(t) := \mathcal{M}_l$ and

$\mathbf{v}_{\boldsymbol{\mathcal{M}}}(\cdot,t) := \underline{\mathbf{0}}$. Each element $\mathcal{M}_l$ is not necessarily connected and either an open hypersurface that does not contain its non-empty boundary or a closed hypersurface. Here, the property of being an open hypersurface, the property of being a closed hypersurface, boundaries $\partial\mathcal{M}_l \subset \partial\Omega_{\text{ext}}$, and closures $\text{cl}(\mathcal{M}_l)$ can be characterized similar to the definitions in Section 3.3.2, with the surface extension domain Ω_{ext} taking the role of the level set domain Ω_Φ.

Extending surface differential operators

Extended differential operators for fields on Ω_{ext}, which pointwise act like the required surface differential operators, can be constructed using the level set function Φ. More precisely, they can be constructed by means of the canonical level set extension of the field $\boldsymbol{\nu}$ of outward-pointing unit normal vectors to Γ. Based on the level set function Φ, the latter extension is defined as

$$\boldsymbol{\nu}^{\boldsymbol{\Phi}}\colon \text{cl}(\Omega_\Phi) \to \mathbb{R}^d, \quad \boldsymbol{\nu}^{\boldsymbol{\Phi}} := \frac{\nabla\Phi}{|\nabla\Phi|}.$$

Strictly speaking, the field $\boldsymbol{\nu}^{\boldsymbol{\Phi}}$ represents (and extends) fields of unit normal vectors to arbitrary level sets of Φ. For each level set Γ_l, the restriction $\boldsymbol{\nu}^{\boldsymbol{\Phi}}|_{\text{cl}(\Gamma_l)}$ is a field of unit normal vectors to $\text{cl}(\Gamma_l)$ which provides an orientation for $\text{cl}(\Gamma_l)$, cf. equation (3.26). This allows to define an extended operator

$$\mathcal{P}^\Phi := \mathcal{I} - \boldsymbol{\nu}^{\boldsymbol{\Phi}} (\boldsymbol{\nu}^{\boldsymbol{\Phi}})^{\text{tr}}$$

analogous to (2.1), which projects the space of vector fields on an arbitrary subset $R \subseteq \text{cl}(\Omega_\Phi)$ onto the subspace of pointwise tangential vector fields on R. The latter shall be understood as the subspace of all vector fields $\boldsymbol{\zeta}\colon R \to \mathbb{R}^d$ which pointwise map into the tangent spaces of $\{\text{cl}(\Gamma_l)\}_{l\in\mathbb{R}}$, so that $\boldsymbol{\zeta}\cdot\boldsymbol{\nu}^{\boldsymbol{\Phi}} = 0$ at every point $\underline{\mathbf{x}} \in R$.

Employing $\mathcal{P}^\Phi$ instead of $\mathcal{P}_\mathcal{M}$ in definitions analogous to those in Section 2.1, we obtain extended differential operators ∇^Φ, grad^Φ and div^Φ for differentiable scalar fields η and differentiable vector fields $\boldsymbol{\xi}$ that live on $\text{cl}(\Omega_\Phi)$ or on an arbitrary d-dimensional subdomain of Ω_Φ, such as the surface extension domain Ω_{ext}. We particularly define

$$\nabla^\Phi := \mathcal{P}^\Phi \circ \nabla \quad \text{and}$$
$$\text{grad}^\Phi \eta := \big(\mathcal{P}^\Phi \nabla\eta\big)^{\text{tr}} = \big(\nabla^\Phi\eta\big)^{\text{tr}}, \quad \text{div}^\Phi \boldsymbol{\xi} := \sum_{i=1}^{d} \big(\nabla^\Phi \xi_i\big)\cdot\underline{\mathbf{e_i}} = \nabla^\Phi\cdot\boldsymbol{\xi}.$$

These extended operators pointwise act like surface differential operators. In particular, considering differentiable fields η and $\boldsymbol{\xi}$ on Ω_{ext} and an arbitrary hypersurface $\mathcal{M}_l \in \mathcal{S}(\Phi,\Omega_{\text{ext}})$, the fields

$$\nabla^\Phi\eta\big|_{\mathcal{M}_l} = \nabla_{\mathcal{M}_l}\eta \quad \text{and} \quad \big(\nabla^\Phi\cdot\boldsymbol{\xi}\big)\big|_{\mathcal{M}_l} = \nabla_{\mathcal{M}_l}\cdot\boldsymbol{\xi}$$

are the (transposed) surface gradient of η and the surface divergence of $\boldsymbol{\xi}$ on $\mathcal{M}_l$, which only depend on the values of η and $\boldsymbol{\xi}$ restricted to $\mathcal{M}_l$, as shown in Chapter 2. Moreover, the field $-\nabla^\Phi \cdot \boldsymbol{\nu}^\Phi$ pointwise matches the total curvature of $\{\mathrm{cl}\,(\Gamma_l)\}_{l\in\mathbb{R}}$. We hence also obtain an extended total curvature

$$H^\Phi := -\nabla^\Phi \cdot \boldsymbol{\nu}^\Phi,$$

where we use that the level set function Φ, which defines the field $\boldsymbol{\nu}^\Phi$, is assumed to be twice differentiable on $\mathrm{cl}\,(\Omega_\Phi)$ in this thesis. Please refer to our precise assumptions on Φ stated in Section 3.3.2.

The theory which has been presented in Section 2.3 yields reformulations of div^Φ and H^Φ in terms of the classical divergence operator in the ambient Cartesian space $\mathbb{R}^d$. This is the subject of the following corollary.

Corollary 4.2.1 (Splitted representation of div^Φ, representation of H^Φ)**.** *Let $\boldsymbol{\xi}$ be a differentiable vector field on Ω_{ext}. Noting that the level set function Φ has symmetric second derivatives on Ω_{ext}, since we assume $\Phi \in C^2\big(\Omega_{\mathrm{ext}}\big)$ in this thesis (cf. Section 3.3.2), Theorem 2.3.2 and Theorem 2.3.3 imply*

$$\nabla^\Phi \cdot \boldsymbol{\xi} \;=\; |\nabla\Phi|^{-1}\; \nabla \cdot \big(|\nabla\Phi| \mathcal{P}^\Phi \boldsymbol{\xi}\big) \;-\; (\boldsymbol{\xi} \cdot \boldsymbol{\nu}^\Phi) H^\Phi$$

and

$$H^\Phi = -\nabla \cdot \boldsymbol{\nu}^\Phi \quad \textit{on} \quad \Omega_{\mathrm{ext}}.$$

Extending surface equations

Using the extended differential operators that we have just defined, we extend surface equations like (4.1c) and (4.5c) to the surface extension domain Ω_{ext}. We do this by formulating given surface equations and their underlying conservation laws on all hypersurfaces in the set $\mathcal{S}(\Phi, \Omega_{\mathrm{ext}})$. Instead of the solution variable u_s, we then look for an extended solution variable u_s^{ext} on Ω_{ext} which simultaneously satisfies the corresponding set of equations.

For this purpose, we need suitable extensions of the data functions on Γ. In particular, for the surface diffusivity tensor $\mathcal{D}_s$, an extension $\mathcal{D}_s^{\mathrm{ext}}$ to $\mathrm{cl}\,(\Omega_{\mathrm{ext}})$ is required, whereas extensions $f_{s,b}^{\mathrm{ext}}(u_b, u_s^{\mathrm{ext}})$ and $f_s^{\mathrm{ext}}(u_s^{\mathrm{ext}})$ to Ω_{ext} are sufficient for $f_{s,b}(u_b, u_s)$ and $f_s(u_s)$. Further requirements on these extensions will be discussed at a later stage. For the time being, we assume suitable extensions to be given.

To describe the extension process in detail, let us first consider parabolic equation (4.1c) as an example, and consider elliptic surface equations later on. For each hypersurface $\mathcal{M}_l \in \mathcal{S}(\Phi, \Omega_{\mathrm{ext}})$, we apply the theory from Section 3.1.1 using the choices $\mathcal{M}(t) := \mathcal{M}_l$ and $\mathbf{v}_{\boldsymbol{\mathcal{M}}}(\cdot, t) := \underline{\mathbf{0}}$. Meanwhile, we choose surface fluxes $\mathbf{q}_{\mathbf{s},\boldsymbol{\mathcal{M}}_\mathbf{l}} := -\mathcal{D}_s^{\mathrm{ext}} \nabla_{\mathcal{M}_l} u_s^{\mathrm{ext}}$, together with source/sink densities $g_{s,\mathcal{M}_l} := f_{s,b}^{\mathrm{ext}}(u_b, u_s^{\mathrm{ext}}) + f_s^{\mathrm{ext}}(u_s^{\mathrm{ext}})$ on $\mathcal{M}_l \times (0, T]$. This yields a system of

continuity equations

$$\partial_t u_s^{\mathrm{ext}} + \nabla_{\mathcal{M}_l} \cdot \mathbf{q}_{\mathbf{s},\boldsymbol{\mathcal{M}}_\mathbf{l}} + (\mathbf{q}_{\mathbf{s},\boldsymbol{\mathcal{M}}_\mathbf{l}} \cdot \boldsymbol{\nu}_{\boldsymbol{\mathcal{M}}_\mathbf{l}}) H_{\mathcal{M}_l} = g_{s,\mathcal{M}_l} \quad \text{on} \quad \mathcal{M}_l \times (0,T]$$

and underlying equivalent conservation laws for surface-bound quantities with concentrations $u_s^{\mathrm{ext}}|_{\mathcal{M}_l \times (0,T]}$.

A solution of this system is a function u_s^{ext} on $\Omega_{\mathrm{ext}} \times (0,T]$, since the hypersurfaces $\mathcal{S}(\Phi, \Omega_{\mathrm{ext}})$ make up a partition of Ω_{ext} with $\Omega_{\mathrm{ext}} = \bigcup_{\mathcal{M}_l \in \mathcal{S}(\Phi,\Omega_{\mathrm{ext}})} \mathcal{M}_l$. Assuming that such a solution u_s^{ext} exists, and assuming that this solution and the vector field $\mathbf{q}_\mathbf{s}^{\mathbf{ext}} := -\mathcal{D}_s^{\mathrm{ext}} \nabla^\Phi u_s^{\mathrm{ext}} = -\mathcal{D}_s^{\mathrm{ext}} \mathcal{P}^\Phi \nabla u_s^{\mathrm{ext}}$ are differentiable in space, the system is equivalent to a single bulk PDE

$$\partial_t u_s^{\mathrm{ext}} + \nabla^\Phi \cdot \mathbf{q}_\mathbf{s}^{\mathbf{ext}} + (\mathbf{q}_\mathbf{s}^{\mathbf{ext}} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}}) H^\Phi = g_s^{\mathrm{ext}} \quad \text{in} \quad \Omega_{\mathrm{ext}} \times (0,T]. \tag{4.8}$$

Here, we define $g_s^{\mathrm{ext}} := f_{s,b}^{\mathrm{ext}}(u_b, u_s^{\mathrm{ext}}) + f_s^{\mathrm{ext}}(u_s^{\mathrm{ext}})$. The vector field $\mathbf{q}_\mathbf{s}^{\mathbf{ext}}$ and the field g_s^{ext} can be seen as an extended flux and an extended source/sink density on $\Omega_{\mathrm{ext}} \times (0,T]$, respectively.

Given that Φ has symmetric second derivatives on Ω_{ext} according to our assumptions (cf. Section 3.3.2), the surface divergence term in equation (4.8) can be reformulated. More specifically, applying Corollary 4.2.1 results in

$$\partial_t \big(|\nabla\Phi| u_s^{\mathrm{ext}} \big) + \nabla \cdot \big(|\nabla\Phi| \mathcal{P}^\Phi \mathbf{q}_\mathbf{s}^{\mathbf{ext}} \big) = |\nabla\Phi| g_s^{\mathrm{ext}} \quad \text{in} \quad \Omega_{\mathrm{ext}} \times (0,T].$$

By defining modified extended data functions

$$\tilde{\mathcal{D}}_s^{\mathrm{ext}} := |\nabla\Phi| \mathcal{P}^\Phi \mathcal{D}_s^{\mathrm{ext}} \mathcal{P}^\Phi, \tag{4.9a}$$

$$\tilde{f}_{s,b}^{\mathrm{ext}}(u_b, u_s^{\mathrm{ext}}) := |\nabla\Phi| f_{s,b}^{\mathrm{ext}}(u_b, u_s^{\mathrm{ext}}), \tag{4.9b}$$

$$\tilde{f}_s^{\mathrm{ext}}(u_s^{\mathrm{ext}}) := |\nabla\Phi| f_s^{\mathrm{ext}}(u_s^{\mathrm{ext}}), \tag{4.9c}$$

we subsequently obtain a bulk PDE of the form

$$\begin{aligned} &\partial_t \big(|\nabla\Phi| u_s^{\mathrm{ext}} \big) \\ &\quad - \nabla \cdot (\tilde{\mathcal{D}}_s^{\mathrm{ext}} \nabla u_s^{\mathrm{ext}}) = \tilde{f}_{s,b}^{\mathrm{ext}}(u_b, u_s^{\mathrm{ext}}) + \tilde{f}_s^{\mathrm{ext}}(u_s^{\mathrm{ext}}) \quad \text{in} \quad \Omega_{\mathrm{ext}} \times (0,T]. \end{aligned} \tag{4.10}$$

Remark 4.2.2 (Alternative derivation of equation (4.10))**.** *It is also possible to employ the equivalent reformulation of continuity equations which is given in Remark 3.1.1. Doing so yields a system of continuity equations*

$$\partial_t u_s^{\mathrm{ext}} + \nabla_{\mathcal{M}_l} \cdot \mathcal{P}_{\mathcal{M}_l} \mathbf{q}_{\mathbf{s},\boldsymbol{\mathcal{M}}_\mathbf{l}} = g_{s,\mathcal{M}_l} \quad \textit{on} \quad \mathcal{M}_l \times (0,T],$$

which can be dealt with in a similar manner to derive a single bulk PDE

$$\partial_t u_s^{\mathrm{ext}} + \nabla^\Phi \cdot \mathcal{P}^\Phi \mathbf{q}_\mathbf{s}^{\mathbf{ext}} = g_s^{\mathrm{ext}} \quad \textit{in} \quad \Omega_{\mathrm{ext}} \times (0,T].$$

In the end, by applying Corollary 4.2.1, we obtain the same formulation (4.10)*.*

Main property of solutions and requirements on extended data functions

The conserved quantity on each single hypersurface is independent of the conserved quantities on every other hypersurface. For each $\mathcal{M}_l \in \mathcal{S}(\Phi, \Omega_{\text{ext}})$, the underlying conservation law only takes into account the tangential part of the extended flux $\mathbf{q_s^{ext}}$ according to Remark 3.1.1. An exact solution u_s^{ext} of equation (4.10) hence is an extension of a solution of surface equation (4.1c).

For the same reason, we are furthermore free to choose an extension $\mathcal{D}_s^{\text{ext}}$ which does not map the tangent space of a hypersurface $\mathcal{M}_l \in \mathcal{S}(\Phi, \Omega_{\text{ext}})$ with $l \neq 0$ into itself at every point. In this light, see also the leftmost projection in the definition of modification $\tilde{\mathcal{D}}_s^{\text{ext}}$, which is given by equation (4.9a). This projection ultimately results from the aforementioned property of the conservation laws that feed into our extension process.

Extensions $\mathcal{D}_s^{\text{ext}}$, $f_{s,b}^{\text{ext}}(u_b, u_s^{\text{ext}})$ and $f_s^{\text{ext}}(u_s^{\text{ext}})$ can thus be chosen arbitrarily without affecting the restriction $u_s^{\text{ext}}|_{\Gamma\times(0,T]}$ of an exact solution of (4.10), as long as they agree with $\mathcal{D}_s$, $f_{s,b}(u_b, u_s^{\text{ext}}|_\Gamma)$ and $f_s(u_s^{\text{ext}}|_\Gamma)$ on $\Gamma \times (0, T]$ and yield an extended system which is uniquely solvable.

Closing the system: Extended initial values and artificial boundary conditions

However, to obtain a well-posed problem with a unique solution, equation (4.10) needs to be supplemented with extended initial values on Ω_{ext}. Furthermore, an artificial boundary condition on $\partial\Omega_{\text{ext}}$ is required, even though Γ is a closed hypersurface and thus has an empty boundary itself. Again, both the extension of the original initial values of equation (4.1) and the boundary condition's data functions can be chosen arbitrarily without influencing $u_s^{\text{ext}}|_{\Gamma\times(0,T]}$, as long as they have sufficient regularity to render the extended system uniquely solvable.

As artificial boundary condition for equation (4.10), we choose its natural boundary condition, i.e., the no-flux boundary condition on $\partial\Omega_{\text{ext}}$. The effect of this boundary condition and the precise reason why a boundary condition is required at all is the subject of the following remark.

Remark 4.2.3 (Effect of the no-flux boundary condition on $\partial\Omega_{\text{ext}}$)**.** *Since the boundaries of the open hypersurfaces in $\mathcal{S}(\Phi, \Omega_{\text{ext}})$ form a partition of $\partial\Omega_{\text{ext}}$ and since we assume that $\nabla\Phi \neq 0$ on $\partial\Omega_{\text{ext}} \subset \text{cl}\,(\Omega_\Phi)$ (cf. Section 3.3.2), we have*

$$\begin{aligned} & \mathbf{q_{s,\mathcal{M}_l}} \cdot \boldsymbol{\mu}_{\boldsymbol{\partial\mathcal{M}_l}} = 0 \quad \textit{on} \quad \partial\mathcal{M}_l \times (0,T] \\ & \textit{for all } \mathcal{M}_l \in \mathcal{S}(\Phi,\Omega_{\text{ext}}) \textit{ with } \partial\mathcal{M}_l \neq \emptyset \\ \Leftrightarrow \quad & \mathbf{q_s^{ext}} \cdot \mathcal{P}^\Phi \mathbf{n}_{\boldsymbol{\partial\Omega_{ext}}} = 0 \quad \textit{on} \quad \partial\Omega_{\text{ext}} \times (0,T] \\ \Leftrightarrow \; & |\nabla\Phi| \mathcal{P}^\Phi \mathbf{q_s^{ext}} \cdot \mathbf{n}_{\boldsymbol{\partial\Omega_{ext}}} = 0 \quad \textit{on} \quad \partial\Omega_{\text{ext}} \times (0,T]. \end{aligned}$$

Accordingly, imposing the no-flux boundary condition on the boundary $\partial\Omega_{\text{ext}}$ corresponds to simultaneously supplementing the continuity equation on every open hypersurface $\mathcal{M}_l$ in the set $\mathcal{S}(\Phi, \Omega_{\text{ext}})$ with a no-flux boundary condition.

Note that the continuity equation on each of the considered open hypersurfaces $\mathcal{M}_l$ requires a boundary condition, as indicated in Section 3.1.1, and that this boundary condition solely influences the restriction $u_s^{\text{ext}}|_{\mathcal{M}_l\times(0,T]}$ of an exact solution of equation (4.10)*. Note furthermore that the no-flux boundary condition on $\partial\Omega_{\text{ext}}$ does not influence the restriction $u_s^{\text{ext}}|_{\Gamma\times(0,T]}$ of an exact solution, given that Γ is a closed hypersurface in $\mathcal{S}(\Phi,\Omega_{\text{ext}})$.*

By supplementing equation (4.10) with the chosen boundary condition, we arrive at the following bulk boundary value problem which replaces surface equation (4.1c) in our approach. To simplify the notation, we henceforth drop the superscript $\diamond^{\text{ext}}$ of the extended solution variable where there is no risk of confusion. Given extended initial values $u_s(\cdot,0)$ on Ω_{ext}, we wish to find $u_s\colon \Omega_{\text{ext}}\times[0,T]\to\mathbb{R}$ with

$$\begin{aligned}
&\partial_t\big(|\nabla\Phi|u_s\big)\\
&\quad -\nabla\cdot(\tilde{\mathcal{D}}_s^{\text{ext}}\nabla u_s) = \tilde{f}_{s,b}^{\text{ext}}(u_b,u_s)+\tilde{f}_s^{\text{ext}}(u_s) \quad \text{in} \quad \Omega_{\text{ext}}\times(0,T], && \text{(4.11a)}\\
&-\tilde{\mathcal{D}}_s^{\text{ext}}\nabla u_s\cdot\mathbf{n}_{\boldsymbol{\partial\Omega_{\text{ext}}}} = 0 \qquad\qquad\qquad\quad \text{on} \quad \partial\Omega_{\text{ext}}\times(0,T], && \text{(4.11b)}
\end{aligned}$$

where $u_b\colon \Omega\times[0,T]\to\mathbb{R}$ is obtained by coupling this new problem with equations (4.1a, 4.1b), and by initial values $u_b(\cdot,0)$ given on Ω.

The core of problem (4.11) is an equation of a well-known type, namely a reaction–diffusion equation on a bulk domain in $\mathbb{R}^d$. It is special due to its first term, which is a weighted mass density, and in the sense that it is degenerated. Particularly, the projection operator $\mathcal{P}^\Phi$ and thus also the diffusivity tensor $\tilde{\mathcal{D}}_s^{\text{ext}}$ defined in equation (4.9a) have a zero eigenvalue in normal direction $\boldsymbol{\nu}^{\boldsymbol{\Phi}}$ to the hypersurfaces in $\mathcal{S}(\Phi,\Omega_{\text{ext}})$. However, problem (4.11) can be treated numerically by a broad class of methods for bulk reaction–diffusion equations with inhomogeneous, anisotropic diffusivity tensors. The special mass density can be easily incorporated into standard time-stepping schemes.

Elliptic surface equations

Elliptic surface equations like (4.5c) can be treated analogously, with identical considerations regarding extended solutions, data extensions and boundary conditions. The extension process which we described above then yields bulk equations like (4.11), but without the need for initial values and without the first term in equation (4.11a):

$$\begin{aligned}
&-\nabla\cdot(\tilde{\mathcal{D}}_s^{\text{ext}}\nabla u_s) = \tilde{f}_{s,b}^{\text{ext}}(u_b,u_s)+\tilde{f}_s^{\text{ext}}(u_s) \quad \text{in} \quad \Omega_{\text{ext}}, && \text{(4.12a)}\\
&-\tilde{\mathcal{D}}_s^{\text{ext}}\nabla u_s\cdot\mathbf{n}_{\boldsymbol{\partial\Omega_{\text{ext}}}} = 0 \qquad\qquad \text{on} \quad \partial\Omega_{\text{ext}}. && \text{(4.12b)}
\end{aligned}$$

In our approach, these two equations replace surface equation (4.5c). They are coupled to equations (4.5a, 4.5b) and we look for a pair (u_b,u_s) of bulk solutions $u_b\colon\Omega\to\mathbb{R}$ and $u_s\colon\Omega_{\text{ext}}\to\mathbb{R}$ that do not depend on time.

	Function definition
$\Phi_1(\underline{\mathbf{x}})$	$\lvert\underline{\mathbf{x}}\rvert - 0.70$
$\Phi_2(\underline{\mathbf{x}})$	$\lvert\underline{\mathbf{x}}\rvert + 0.15\,\lvert\underline{\mathbf{x}}\rvert \sin\big(2.0\arctan2(\mathrm{x}_1,\mathrm{x}_0)\big) - 0.70$
$\Phi_3(\underline{\mathbf{x}})$	$\lvert\underline{\mathbf{x}}\rvert + 0.15\,\lvert\underline{\mathbf{x}}\rvert \sin\big(8.0\arctan2(\mathrm{x}_1,\mathrm{x}_0)\big) - 0.60$
$\Phi_4(\underline{\mathbf{x}})$	$\sqrt{\mathrm{x}_0^2 + 4.0\,\mathrm{x}_1^2} + 0.30\sqrt{\mathrm{x}_0^2 + 4.0\,\mathrm{x}_1^2}\,\sin\big(8.0\arctan2(\mathrm{x}_1,\mathrm{x}_0)\big) - 0.50$
$\Phi_5(\underline{\mathbf{x}})$	$\lvert\underline{\mathbf{x}}\rvert + 0.17\,\lvert\underline{\mathbf{x}}\rvert \arctan2(-\mathrm{x}_1,\mathrm{x}_0)\sin\big(2.0\arctan2(-\mathrm{x}_1,\mathrm{x}_0)\big) - 0.50$
$\Phi_6(\underline{\mathbf{x}})$	$\lvert\underline{\mathbf{x}}\rvert + 0.17\,\lvert\underline{\mathbf{x}}\rvert \arctan2(-\mathrm{x}_1,\mathrm{x}_0)\sin\big(4.0\arctan2(-\mathrm{x}_1,\mathrm{x}_0)\big) - 0.45$

Table 4.1.: Level set functions that are employed in Figure 4.2, defined in two-dimensional Cartesian coordinates $\underline{\mathbf{x}} = (\mathrm{x}_0, \mathrm{x}_1)$. The function $\arctan2(\mathrm{x}_1, \mathrm{x}_0)$ yields the azimuthal angle ϕ of polar coordinates (r, ϕ) and will be defined later on in Remark 4.3.1.

Choosing the surface extension domain in a suitable manner

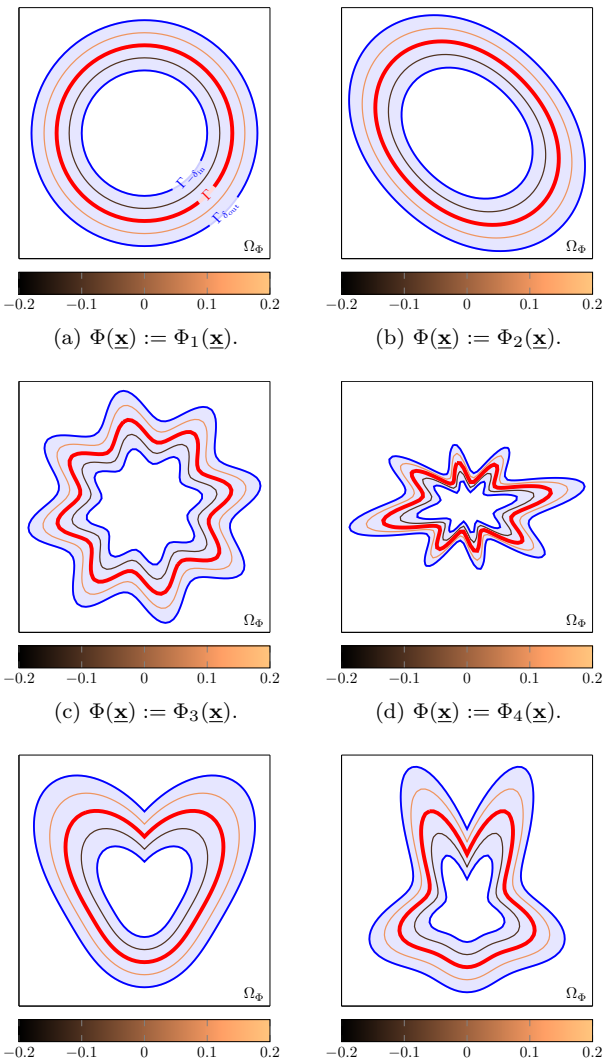
Even though an exact solution of problem (4.11) or problem (4.12) represents conserved quantities which live on different hypersurfaces in the set $\mathcal{S}(\Phi, \Omega_{\text{ext}})$ and do not interfere with each other, spatial discretization of the problem using numerical methods will inherently lead to a coupling between those quantities. The support of each basis function of a discrete bulk solution space will overlap a collection of different hypersurfaces.

Therefore, we can not expect that the conservation properties embedded in the original surface equations properly carry over in a discrete sense. Moreover, despite the above considerations, the artificial boundary conditions (4.11b) and (4.12b) can nonetheless have an influence on that part of a discrete solution which we are interested in, namely its restriction to Γ. It is indeed well-known that artificial boundary conditions of extended surface equations can lead to artifacts which spread from the boundary $\partial\Omega_{\text{ext}}$ to Γ, resulting in low order convergence of the numerical method (see e.g. Nemitz et al. (2009, particularly Section 2.3) for natural boundary conditions like (4.11b) or (4.12b), and Greer et al. (2006, particularly Figure 2 in Section 7.1) for Dirichlet boundary conditions).

For this reason, we choose a special surface extension domain Ω_{ext} which has been proposed by Deckelnick et al. (2010), namely a narrow band Ω_δ around Γ that consists of entire, non-empty, closed level sets of Φ:

$$\begin{aligned}\Omega_{\text{ext}} := \Omega_\delta := \bigcup\nolimits_{-\delta_{\text{in}} < l < \delta_{\text{out}}} \Gamma_l, \quad \delta := \delta_{\text{in}} + \delta_{\text{out}},\\ \delta_{\text{in}}, \delta_{\text{out}} \in \mathbb{R}^{>0} \text{ such that } \Gamma_l \text{ non-empty and closed for } -\delta_{\text{in}} < l < \delta_{\text{out}}.\end{aligned} \tag{4.13}$$

Examples are depicted in Figure 4.2.



(a) $\Phi(\underline{\mathbf{x}}) := \Phi_1(\underline{\mathbf{x}})$. (b) $\Phi(\underline{\mathbf{x}}) := \Phi_2(\underline{\mathbf{x}})$.

(c) $\Phi(\underline{\mathbf{x}}) := \Phi_3(\underline{\mathbf{x}})$. (d) $\Phi(\underline{\mathbf{x}}) := \Phi_4(\underline{\mathbf{x}})$.

(e) $\Phi(\underline{\mathbf{x}}) := \Phi_5(\underline{\mathbf{y}})$, with $\underline{\mathbf{y}} := (\mathrm{x}_0, \mathrm{x}_1 + 0.15)^{\mathrm{tr}}$. (f) $\Phi(\underline{\mathbf{x}}) := \Phi_6(\underline{\mathbf{y}})$, with $\underline{\mathbf{y}} := (\mathrm{x}_0, \mathrm{x}_1 + 0.24)^{\mathrm{tr}}$.

Figure 4.2.: Narrow band Ω_δ which we use as surface extension domain Ω_{ext} in our schemes. To illustrate the geometrical setting, we use different level set functions Φ (defined by means of Table 4.1) with $\Omega_\Phi := (-1,1)^2$ (depicted in black), together with narrow band parameters $\delta_{\mathrm{in}} := \delta_{\mathrm{out}} := 0.2$. The narrow band $\Omega_\delta = \Omega_{\mathrm{ext}}$ and its boundary $\partial\Omega_\delta = \partial\Omega_{\mathrm{ext}} = \left\{\Gamma_{-\delta_{\mathrm{in}}}, \Gamma_{\delta_{\mathrm{out}}}\right\}$ are depicted in blue. The hypersurface $\Gamma = \Gamma_0 = \mathcal{M}_0$ is depicted in red. Hypersurfaces $\Gamma_l = \mathcal{M}_l \in \mathcal{S}(\Phi, \Omega_{\mathrm{ext}})$ with $l \neq 0$ are colored according to the value l and its associated color from the color bars at the bottom.

Using this specific choice of Ω_{ext}, discretization still leads to a coupling within the set of conserved quantities, but the artificial boundary conditions (4.11b) and (4.12b) do not cause any artifacts in a discrete solution. According to Remark 4.2.3, they do not influence an exact solution $u_s = u_s^{\text{ext}}$ at all, given that there is no open hypersurface in the set $\mathcal{S}(\Phi, \Omega_{\text{ext}})$. In fact, conditions (4.11b) and (4.12b) are no longer required. They are fulfilled automatically since the normal component of $\tilde{\mathcal{D}}_s^{\text{ext}} \nabla u_s$ on the left-hand side of both equations vanishes on the narrow band boundary $\partial\Omega_\delta$. More specifically, the latter boundary is of the form $\partial\Omega_\delta = \left\{\Gamma_{-\delta_{\text{in}}}, \Gamma_{\delta_{\text{out}}}\right\}$, such that $\mathbf{n}_{\boldsymbol{\partial\Omega_\delta}} = \boldsymbol{\nu}^{\boldsymbol{\Phi}}$ on $\partial\Omega_\delta$. Furthermore, the diffusivity tensor $\tilde{\mathcal{D}}_s^{\text{ext}}$ defined in equation (4.9a) employs the projection operator $\mathcal{P}^\Phi$. This operator is self-adjoint and it satisfies $\mathcal{P}^\Phi \boldsymbol{\nu}^{\boldsymbol{\Phi}} = \underline{\mathbf{0}}$ on $\partial\Omega_\delta \subset \text{cl}\,(\Omega_\Phi)$. As long as these two equalities hold, we will not only obtain an intact exact solution, but also discrete solutions that are not distorted by some boundary condition. Of course, we need to assure that both equalities remain valid if the exact geometry is replaced by some discrete reconstruction.

A further advantage of choosing $\Omega_{\text{ext}} := \Omega_\delta$ is the following. As $\delta \to 0$, the extended equation tends back toward the surface equation we started from. For $\delta \to 0$, we can hence expect that even a discrete solution of, say, problem (4.11) adopts properties of an exact solution of (4.1c), e.g. its conservation properties.

On the downside, the narrow band Ω_δ usually is a complex-shaped bulk domain if Γ is complex-shaped, especially for small values of δ.

Final remarks on the extension process

To sum up, by using the extension process, we end up with a system of PDEs comprising well-known bulk equations instead of having to deal with a system that includes a native surface equation. But we need a numerical method which is very flexible with regard to the computational geometry. It must provide an easy and efficient way to handle problems on complex-shaped bulk domains that are implicitly described by a level set function. In addition, to allow for enforcing convergence of the surface part of the discrete solution toward the surface part of the solution of the original model problem, it needs to enable us to scale the narrow band parameters δ_{in} and δ_{out} with some discretization parameter that goes to zero.

Besides geometrical issues, the numerical method must be able to cope with inhomogeneous, anisotropic diffusivity tensors. Furthermore, we want to obtain discrete solutions which reflect the underlying conservation properties in the best possible way. Our specific focus lies on the conservation properties induced by the conservation laws that are associated with the original model problem. Hence, a numerical method is required which is either conservative in terms of discrete analogues to those conservation laws itself or at least allows for suitable modifications.

4.2.2. Unfitted discontinuous Galerkin

In this thesis, we consider a method that is specially designed for such problems. The UDG method (Bastian and Engwer, 2009; Engwer, 2009), that was sketched in Section 1.3.2, is a general approach to solve PDEs on complicated bulk domains, such as the narrow band Ω_δ which we use as a surface extension domain and the domain Ω from the bulk part of the problem. It combines the concepts of unfitted finite element methods (Barrett and Elliott, 1987; Hansbo and Hansbo, 2002) with the ideas of DG discretizations that were discussed in Section 3.2. In particular, employing host DG formulations suitable for the problem at hand, PDEs are discretized using discrete function spaces that are based on geometrically unfitted bulk meshes. As described in Section 1.3, geometrically unfitted meshes do not need to resolve bulk domain boundaries like $\partial\Omega_\delta$ and $\partial\Omega = \Gamma$ or discrete reconstructions of those boundaries.

In the UDG approach, discretization of the computational geometry is based on a discrete reconstruction of the bulk domains under consideration and their boundaries. To perform numerical integration over this discrete geometry, the approach applies the general concept of using local triangulations, which we discuss later on in this section. For both the discrete geometry reconstruction and the local triangulations, a discrete analogue to the continuous level set description that we have considered so far can be employed. To implement this discrete analogue, the approach uses a fixed, simple *geometry mesh* of a domain which covers the bulk domains of interest.

A coarser *fundamental mesh* of the same domain is employed for spatial discretization. For each PDE, a discrete function space is constructed by defining shape functions on an appropriate subset of this mesh and restricting their support to the reconstruction of the bulk domain which is associated with the PDE.

Since the fundamental mesh can be chosen independently of the geometry mesh, the approach allows to choose the size of discrete function spaces independently of geometric properties. Using the DG machinery allows for a fully element-local construction of the discrete function spaces and facilitates using higher order shape functions. Moreover, as shown in Section 3.2, it is especially attractive for continuity equations which model conserved bulk quantities. In addition to recovering discrete analogues to global bulk conservation properties, discrete analogues to the underlying bulk conservation laws can be easily incorporated locally at the level of fundamental mesh elements, such that fluxes over element boundaries can be accurately described. As we will show, this is also true for the surface conservation laws and global surface conservation properties which we have considered in Section 4.1.

Basic meshes, discretization of the computational geometry

We generalize the concept of meshes which has been introduced in Section 3.2.1. Given a bulk domain $\mathcal{D}$ in $\mathbb{R}^d$, a mesh $\mathcal{T}(\mathcal{D})$ shall be understood as a set of

open, disjoint elements $K_0, \dots, K_{M-1} \subset \mathbb{R}^d$ with

$$\operatorname{cl}(\mathcal{D}) \subseteq \bigcup\nolimits_{i=0,\dots,M-1} \operatorname{cl}(K_i)\,. \tag{4.14}$$

If $\operatorname{cl}(\mathcal{D})$ is a proper subset, we say that $\mathcal{T}(\mathcal{D})$ is a mesh which *covers* $\mathcal{D}$. Otherwise, in case of equality in (4.14), we keep using the notions from Section 3.2.1, i.e. we call $\mathcal{T}(\mathcal{D})$ a mesh *of* $\mathcal{D}$ and say that $\mathcal{T}(\mathcal{D})$ *resolves the boundary* $\partial\mathcal{D}$.

In this thesis, a fundamental mesh and a geometry mesh are used which are meshes of the level set domain Ω_Φ. They are expected to comprise shape regular elements that are tetrahedra or hexahedra for $d = 3$, and triangles or quadrilaterals for $d = 2$. We denote the fundamental mesh by $\mathcal{T}_h(\Omega_\Phi)$ and the geometry mesh by $\mathcal{T}_\hbar(\Omega_\Phi)$, where h and $\hbar$ denote their individual maximum element sizes, also known as *mesh widths*, which are given by

$$h := \max_{K \in \mathcal{T}_h(\Omega_\Phi)} \operatorname{diam}(K) \quad \text{and} \quad \hbar := \max_{K \in \mathcal{T}_\hbar(\Omega_\Phi)} \operatorname{diam}(K).$$

After choosing a specific geometry mesh $\mathcal{T}_\hbar(\Omega_\Phi)$, we start by discretizing the computational geometry. Let $X_\hbar(\Omega_\Phi)$ be the space of piecewise linear (for simplices), bilinear (for quadrilaterals) or trilinear (for hexahedra) continuous functions over $\mathcal{T}_\hbar(\Omega_\Phi)$ and let $I_\hbar$ denote an operator which performs interpolation of functions in $C^0(\Omega_\Phi)$ into $X_\hbar(\Omega_\Phi)$. To obtain a discrete geometry reconstruction, we define some discrete level set function $\Phi_\hbar$ as a function over $\mathcal{T}_\hbar(\Omega_\Phi)$ by setting $\Phi_\hbar := I_\hbar \Phi$. Subsequently, we consider its level sets and appropriate collections of those level sets. More precisely, we define sets

$$\Gamma_{l,\hbar} := \Phi_\hbar\big|_{\Omega_\Phi}^{-1}(l) = \big\{\underline{\mathbf{x}} \in \Omega_\Phi \;\big|\; \Phi_\hbar(\underline{\mathbf{x}}) = l\big\}, \quad l \in \mathbb{R},$$
$$\Gamma_\hbar := \Gamma_{0,\hbar}, \quad \Omega_\hbar := \bigcup\nolimits_{l<0} \Gamma_{l,\hbar} \quad \text{and} \quad \Omega_{\delta,\hbar} := \bigcup\nolimits_{-\delta_{\text{in}}<l<\delta_{\text{out}}} \Gamma_{l,\hbar}.$$

Being level sets of the discrete level set function $\Phi_\hbar$, the sets $\Gamma_{l,\hbar}$ are discrete analogues to the level sets Γ_l of the continuous level set function Φ, cf. the definition of Γ_l in equation (3.23). The sets $\Omega_\hbar$ and $\Gamma_\hbar$ are discrete counterparts to the given bulk domain Ω and the hypersurface Γ making up its boundary, cf. representations (3.24). Similarly, the set $\Omega_{\delta,\hbar}$ is a discrete analogue to the narrow band Ω_δ which has been introduced in equation (4.13). See Figure 4.3 for illustrations.

To make sure that the discrete narrow band $\Omega_{\delta,\hbar}$ provides all benefits of its continuous counterpart Ω_δ, we require the narrow band parameters δ_{in} and δ_{out} to be chosen small enough that all discrete level sets $\Gamma_{l,\hbar}$ which contribute to $\Omega_{\delta,\hbar}$ are non-empty and closed.

Complementary to the discretized computational geometry, we need a discrete analogue to the field $\nabla\Phi$ which is used in the extended surface equations. For this purpose, we assign a piecewise variant of the (transposed) classical gradient operator in $\mathbb{R}^d$ to the space $X_\hbar(\Omega_\Phi)$ that contains the discrete level set function $\Phi_\hbar$. Applied to $\Phi_\hbar$, this piecewise variant can be defined by setting

$\nabla_{\hbar}\Phi_{\hbar}\big|_K := \nabla\Big[\Phi_{\hbar}\big|_K\Big]$ for each $K \in \mathcal{T}_{\hbar}(\Omega_\Phi)$, and by extending this definition to the faces of geometry mesh elements, i.e. to the whole level set domain Ω_Φ and its boundary. On each face of an element, we do the latter by evaluating the gradient in an arbitrarily chosen adjacent element.

Meshes and discrete approximation spaces for the surface part of the problem

Based on the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ and the discretized computational geometry, we define a mesh of bulk domain $\Omega_{\delta,\hbar}$ by first specifying a mesh that covers $\Omega_{\delta,\hbar}$ as

$$\hat{\mathcal{T}}_h(\Omega_{\delta,\hbar}) := \big\{\hat{K} \in \mathcal{T}_h(\Omega_\Phi) \;\big|\; \mathrm{meas}_{\mathbb{R}^d}\big(\hat{K} \cap \Omega_{\delta,\hbar}\big) > 0\big\},$$

and then considering its restriction to $\Omega_{\delta,\hbar}$ given by

$$\mathcal{T}_h(\Omega_{\delta,\hbar}) := \big\{K = \hat{K} \cap \Omega_{\delta,\hbar} \;\big|\; \hat{K} \in \hat{\mathcal{T}}_h(\Omega_{\delta,\hbar})\big\}.$$

While $\hat{\mathcal{T}}_h(\Omega_{\delta,\hbar})$ is a subset of the fundamental mesh and hence consists of shape regular elements, we note that the $K \in \mathcal{T}_h(\Omega_{\delta,\hbar})$ are arbitrarily-shaped elements and call those elements *cut cells*. In general, they are not either shape regular or even convex. We call $\hat{\mathcal{T}}_h(\Omega_{\delta,\hbar})$ an *active mesh* for the bulk domain $\Omega_{\delta,\hbar}$ and say that $\mathcal{T}_h(\Omega_{\delta,\hbar})$ is a *cut cell mesh* of $\Omega_{\delta,\hbar}$. Cut cell meshes can be seen as the intersection of an active mesh and its associated bulk domain. See Figure 4.4 for illustrations.

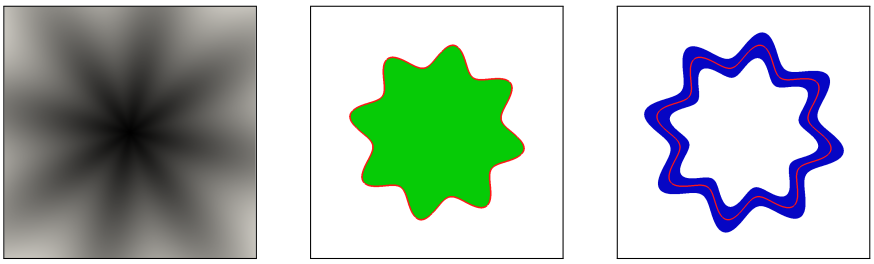
In addition to these meshes, we will also consider the set of internal faces of the cut cell mesh $\mathcal{T}_h(\Omega_{\delta,\hbar})$, which can be defined as

$$\mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar}) := \big\{E = \partial K_E^+ \cap \partial K_E^- \;\big|\; K_E^+, K_E^- \in \mathcal{T}_h(\Omega_{\delta,\hbar}),\, K_E^+ \neq K_E^-,\, \mathrm{meas}_{\mathbb{R}^{d-1}}(E) > 0\big\}.$$

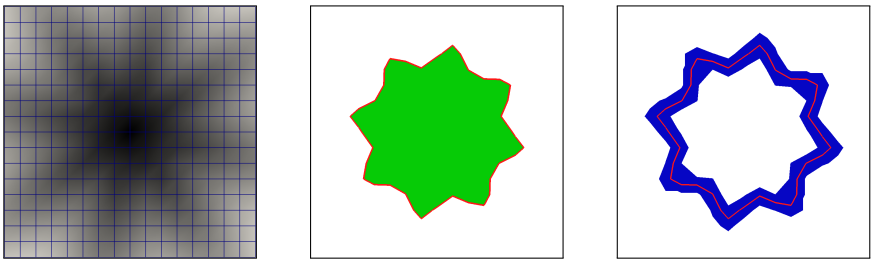
Referring to Section 3.2.1, we recall that the set $\mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar})$ is often called the internal skeleton of the mesh. Each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar})$ is the intersection of the boundaries of two elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_{\delta,\hbar})$. Their boundaries are oriented by two fields $\mathbf{n}_{\boldsymbol{\partial}\mathbf{K}_\mathbf{E}^+}, \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}_\mathbf{E}^-}$ of outward-pointing unit normal vectors which are opposing each other on E. To each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar})$, we can hence assign a dedicated field of unit vectors normal to E by assigning the names K_E^+, K_E^- to the adjacent elements in a fixed manner and by arbitrarily choosing $\mathbf{n}_\mathbf{E} := \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}_\mathbf{E}^+}\big|_E$. See Figure 4.5b for an illustration.

Using standard DG shape functions on the active mesh $\hat{\mathcal{T}}_h(\Omega_{\delta,\hbar})$, with their support restricted to the cut cells in $\mathcal{T}_h(\Omega_{\delta,\hbar})$, we construct discrete finite element spaces of piecewise polynomial functions over $\Omega_{\delta,\hbar}$ given by

$$V_{s,h}(\Omega_{\delta,\hbar}) := \Big\{v_{s,h} \in L^2(\Omega_{\delta,\hbar}) \;\Big|\; v_{s,h}\big|_K \in \mathbb{P}(K)\ \forall K \in \mathcal{T}_h(\Omega_{\delta,\hbar})\Big\}.$$



(a) Description of an example geometry (Ω, Γ) (middle) and an associated narrow band Ω_δ (right) using a continuous level set function Φ (left).



(b) Discrete reconstruction $(\Omega_\hbar, \Gamma_\hbar)$ of the same example geometry (middle) and discrete reconstruction $\Omega_{\delta,\hbar}$ of the associated narrow band (right) using a discrete level set function $\Phi_\hbar$ over some geometry mesh $\mathcal{T}_\hbar(\Omega_\Phi)$ (left).

Figure 4.3.: Discrete geometry reconstruction in the UDG approach using the level set framework. Note that (b) shows results for a discrete level set function over a very coarse geometry mesh to exemplify the difference between the exact geometry and its reconstruction. An adequately fine geometry mesh is employed in practice, yielding reconstructions similar to what is shown in (a).

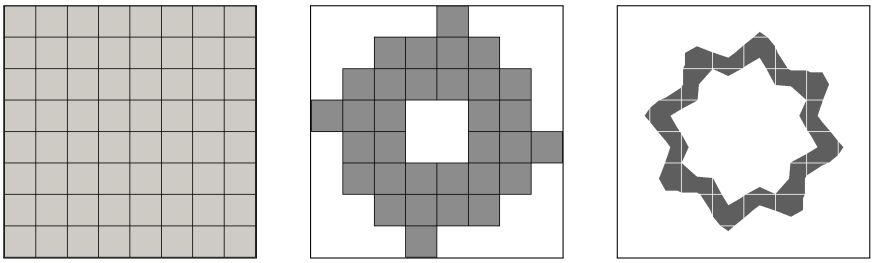


Figure 4.4.: Some fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ (left), the corresponding active mesh $\hat{\mathcal{T}}_h(\Omega_{\delta,\hbar})$ for the reconstructed narrow band $\Omega_{\delta,\hbar}$ (with $\hbar = h/2$) that is shown in the right picture of Figure 4.3b (middle), and the corresponding cut cell mesh $\mathcal{T}_h(\Omega_{\delta,\hbar})$ of $\Omega_{\delta,\hbar}$ (right).

Here, $\mathbb{P}(K)$ denotes some space of polynomial functions over an element K, such as the space $P_k(K)$ of polynomial functions of total degree less than or equal to some $k \in \mathbb{N}$, or the space $Q_k(K)$ of polynomial functions with a degree less than or equal to k in each coordinate direction.

In general, functions in $V_{s,h}(\Omega_{\delta,\hbar})$ are discontinuous and do not take a unique value along the internal skeleton $\mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar})$. Nevertheless, on each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar})$ with adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_{\delta,\hbar})$ as defined above, a function $v_{s,h} \in V_{s,h}(\Omega_{\delta,\hbar})$ has two well-defined traces $v_{s,h}|_{\partial K_E^+}$, $v_{s,h}|_{\partial K_E^-}$. Using these traces, we define the jump of a function $v_{s,h}$ on an internal face E as

$$[\![v_{s,h}]\!]\big|_E := v_{s,h}\big|_{\partial K_E^+} - v_{s,h}\big|_{\partial K_E^-}$$

and its average as

$$\{v_{s,h}\}\big|_E := \frac{1}{2}\Big(v_{s,h}\big|_{\partial K_E^+} + v_{s,h}\big|_{\partial K_E^-}\Big).$$

The definitions of these two operators $[\![\cdot]\!]$ and $\{ \cdot \}$ will not only be used for functions in $V_{s,h}(\Omega_{\delta,\hbar})$. We will rather use it for all functions that have a reasonable definition on each internal face E of the cut cell mesh $\mathcal{T}_h(\Omega_{\delta,\hbar})$, with two branches $v_{s,h}\big|_{\partial K_E^+}$ and $v_{s,h}\big|_{\partial K_E^-}$ which are associated with the adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_{\delta,\hbar})$.

For functions in $V_{s,h}(\Omega_{\delta,\hbar})$, we furthermore define a piecewise variant of the (transposed) classical gradient operator in $\mathbb{R}^d$. Given a function $v_{s,h}$, we first set

$$\nabla_h v_{s,h}\big|_K := \nabla\Big[v_{s,h}\big|_K\Big] \tag{4.15}$$

for each $K \in \mathcal{T}_h(\Omega_{\delta,\hbar})$. Subsequently, we extend this definition to the faces in $\mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar})$, i.e. to the whole bulk domain $\Omega_{\delta,\hbar}$. More specifically, on each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar})$, we define two branches by evaluating the gradient in the two adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(\Omega_{\delta,\hbar})$. We would like to emphasize that this definition does not permit single-valued evaluations on the internal skeleton $\mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar})$. This is sufficient, though, as long as only the jump and average of terms in $\nabla_h v_{s,h}$ are evaluated on internal faces. The latter will be all we need. For the sake of convenient notation, however, we will sometimes treat $\nabla_h v_{s,h}$ as if it was single-valued on $\mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar})$, choosing one of its two branches arbitrarily on each internal face. This abuse of notation will only be performed in cases where the exact value of $\nabla_h v_{s,h}$ on the internal skeleton does not matter, e.g. in bulk integrals over $\Omega_{\delta,\hbar}$, where the internal faces are sets of measure zero.

Remark 4.2.4 (Geometrically unfitted spaces)**.** *The discrete spaces $V_{s,h}(\Omega_{\delta,\hbar})$ defined above can be seen as geometrically unfitted spaces. The essential step in their construction employs piecewise polynomial standard shape functions defined on the shape regular elements which make up the active mesh $\hat{\mathcal{T}}_h(\Omega_{\delta,\hbar})$. Restricting the support of those shape functions to the reconstructed geometry*

which we are actually interested in is a separate second step in the construction.

This is beneficial in multiple ways. The spaces $V_{s,h}(\Omega_{\delta,\hbar})$ have a natural basis which is induced by the basis functions of those discrete spaces that can be defined using the full mesh elements in $\hat{\mathcal{T}}_h(\Omega_{\delta,\hbar})$. Moreover, the construction is easy and efficient, especially if the construction process can be executed completely element-locally – which holds true in our case since we use a DG ansatz. Finally, functions have a straightforward extension up to the whole active mesh. This can be useful to transfer solutions between discrete time steps in future extensions of the approach to problems on evolving geometries.

In practice, the actual restriction step can even be replaced by restricting solution functions to the reconstructed geometry as a postprocessing step. The weak formulation will employ the reconstructed geometry in any case.

Analogous definitions for the bulk part of the problem

For the bulk domain $\Omega_\hbar$, we analogously define an active mesh

$$\hat{\mathcal{T}}_h(\Omega_\hbar) := \big\{\hat{K} \in \mathcal{T}_h(\Omega_\Phi) \;\big|\; \mathrm{meas}_{\mathbb{R}^d}\big(\hat{K} \cap \Omega_\hbar\big) > 0\big\},$$

a cut cell mesh

$$\mathcal{T}_h(\Omega_\hbar) := \big\{K = \hat{K} \cap \Omega_\hbar \;\big|\; \hat{K} \in \hat{\mathcal{T}}_h(\Omega_\hbar)\big\}$$

and its internal skeleton

$$\mathcal{E}_h^{\mathrm{int}}(\Omega_\hbar) := \big\{E = \partial K_E^+ \cap \partial K_E^- \;\big|\; K_E^+, K_E^- \in \mathcal{T}_h(\Omega_\hbar),\, K_E^+ \neq K_E^-,\, \mathrm{meas}_{\mathbb{R}^{d-1}}(E) > 0\big\},$$

together with a dedicated unit normal vector field for every internal face, and discrete finite element spaces

$$V_{b,h}(\Omega_\hbar) := \Big\{v_{b,h} \in L^2(\Omega_\hbar) \;\Big|\; v_{b,h}\big|_K \in \mathbb{P}(K)\ \forall K \in \mathcal{T}_h(\Omega_\hbar)\Big\}.$$

Furthermore, we analogously define two operators $[\![\cdot]\!]$ and $\{\cdot\}$ which describe the jump and the average of functions that have a reasonable definition on each internal face in $\mathcal{E}_h^{\mathrm{int}}(\Omega_\hbar)$, with two potentially different branches on each of those faces. Considering the discrete functions in $V_{b,h}(\Omega_\hbar)$, for instance, those branches represent traces. For the latter set of functions, we also consider a piecewise variant ∇_h of the (transposed) classical gradient operator in $\mathbb{R}^d$, which is defined analogous to the piecewise variant for functions in $V_{s,h}(\Omega_{\delta,\hbar})$.

Note that we do not explicitly distinguish between both jump operators and both average operators in our notation, since the targeted variant will be always clear from the internal face on which an operator is evaluated and from the operator's argument. Likewise and for similar reasons, our notation does not distinguish between the two piecewise gradient operators.

Discretization in space

We discretize in space in three steps that are analogous to our proceeding in Section 3.2. First, we formulate the PDEs of the problem at hand in a weak sense. As a second step, we replace the geometry (Ω, Ω_δ) in the resulting weak formulation by its discrete reconstruction $(\Omega_\hbar, \Omega_{\delta,\hbar})$, and the level set function Φ by its discrete analogue $\Phi_\hbar$. In the course of this, we consequently use $\nabla_\hbar$ as a replacement for the classical gradient operator. The replacement of Φ by $\Phi_\hbar$ is performed particularly in the terms that are defined in equations (4.9), i.e. in $\tilde{\mathcal{D}}_s^{\text{ext}}$ (including the projection operator $\mathcal{P}^\Phi$), in $\tilde{f}_{s,b}^{\text{ext}}(u_b, u_s)$ and in $\tilde{f}_s^{\text{ext}}(u_s)$. It should be noted that, for these terms, we will keep the notation though, in order to avoid over-complicating our notation unnecessarily. Finally, as a third step, we restrict the set of admissible functions by considering only functions that are representable using the discrete function spaces $V_{b,h}(\Omega_\hbar)$ and $V_{s,h}(\Omega_{\delta,\hbar})$, and we employ a suitable host DG formulation for each equation.

For the modified system (4.1a, 4.1b, 4.11), which is associated with the class of parabolic model problems (4.1), this yields the following semidiscretization.

Semidiscretization 4.2.5 (Basic formulation, parabolic problems)**.** *Given suitable approximations $u_{b,h}(\cdot,0) \in V_{b,h}(\Omega_\hbar)$ and $u_{s,h}(\cdot,0) \in V_{s,h}(\Omega_{\delta,\hbar})$ of the modified system's initial values, we seek a pair of semidiscrete functions $u_{b,h}\colon \Omega_\hbar \times [0,T] \to \mathbb{R}$ and $u_{s,h}\colon \Omega_{\delta,\hbar} \times [0,T] \to \mathbb{R}$ with $u_{b,h}(\cdot,t) \in V_{b,h}(\Omega_\hbar)$ and $u_{s,h}(\cdot,t) \in V_{s,h}(\Omega_{\delta,\hbar})$, such that for each $t \in (0,T]$*

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{\Omega_\hbar} u_{b,h}\,\varphi_{b,h}\,\mathrm{d}x\right) + a_b(u_{b,h},\varphi_{b,h})\\ &\quad= c_b(u_{b,h},u_{s,h},\varphi_{b,h}) \qquad \forall \varphi_{b,h} \in V_{b,h}(\Omega_\hbar), \quad (4.16a)\end{aligned}$$

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{\Omega_{\delta,\hbar}} |\nabla_\hbar \Phi_\hbar|\,u_{s,h}\,\varphi_{s,h}\,\mathrm{d}x\right) + a_s(u_{s,h},\varphi_{s,h})\\ &\quad= \int_{\Omega_{\delta,\hbar}} \big(\tilde{f}_{s,b}^{\text{ext}}(u_{b,h},u_{s,h}) + \tilde{f}_s^{\text{ext}}(u_{s,h})\big)\,\varphi_{s,h}\,\mathrm{d}x \quad \forall \varphi_{s,h} \in V_{s,h}(\Omega_{\delta,\hbar}). \quad (4.16b)\end{aligned}$$

Here, $a_b\colon V_{b,h}(\Omega_\hbar) \times V_{b,h}(\Omega_\hbar) \to \mathbb{R}$ and $a_s\colon V_{s,h}(\Omega_{\delta,\hbar}) \times V_{s,h}(\Omega_{\delta,\hbar}) \to \mathbb{R}$ are given bilinear forms that depend on the choice of the two host DG formulations, and $c_b\colon V_{b,h}(\Omega_\hbar) \times V_{s,h}(\Omega_{\delta,\hbar}) \times V_{b,h}(\Omega_\hbar) \to \mathbb{R}$ is a potentially nonlinear form defined by

$$\begin{aligned}&c_b(u_{b,h},u_{s,h},\varphi_{b,h})\\ &\qquad:= \int_{\Gamma_\hbar} f_{b,s}(u_{b,h},u_{s,h}|_{\Gamma_\hbar})\,\varphi_{b,h}\,\mathrm{d}\sigma + \int_{\Omega_\hbar} f_b(u_{b,h})\,\varphi_{b,h}\,\mathrm{d}x. \qquad (4.17)\end{aligned}$$

For the modified system (4.5a, 4.5b, 4.12), which is associated with the class of elliptic model problems, proceeding analogously yields a fully discrete scheme for (4.5):

Scheme 4.2.6 (Basic formulation, elliptic problems)**.** *Find a pair of discrete functions* $(u_{b,h}, u_{s,h}) \in V_{b,h}(\Omega_{h}) \times V_{s,h}(\Omega_{\delta,h})$*, such that*

$$a_b(u_{b,h}, \varphi_{b,h}) = c_b(u_{b,h}, u_{s,h}, \varphi_{b,h}), \tag{4.18a}$$

$$a_s(u_{s,h}, \varphi_{s,h}) = \int_{\Omega_{\delta,h}} \big(\tilde{f}^{\mathrm{ext}}_{s,b}(u_{b,h}, u_{s,h}) + \tilde{f}^{\mathrm{ext}}_{s}(u_{s,h})\big)\, \varphi_{s,h} \,\mathrm{d}x, \tag{4.18b}$$

for all $(\varphi_{b,h}, \varphi_{s,h}) \in V_{b,h}(\Omega_{h}) \times V_{s,h}(\Omega_{\delta,h})$*. Here,* a_b*,* a_s *and* c_b *are the forms that have been introduced in Semidiscretization 4.2.5.*

The classical SIPG formulation and the SWIPG formulation, which are both described in Section 3.2, are appropriate for bulk reaction–diffusion equations of parabolic nature and for their elliptic steady-state counterparts. Therefore, they are suitable host DG formulations for each of the two PDEs in each of the two modified systems which we are considering. For the extended surface equations with their inhomogeneous, anisotropic diffusivity tensor, especially the SWIPG formulation seems attractive since it was designed to handle equations with heterogeneous diffusivity, see Section 3.2.3. In Section 4.3, we will compare the performance of our approach for both formulations.

With the classical SIPG formulation or the SWIPG formulation, the bilinear forms in Semidiscretization 4.2.5 and Scheme 4.2.6 take the form

$$a_b(u_{b,h}, \varphi_{b,h}) := a^{\mathrm{host_b}}\big(\Omega_{h},\ \mathcal{D}_b,\ u_{b,h}, \varphi_{b,h}, \gamma_b\big), \tag{4.19a}$$

$$a_s(u_{s,h}, \varphi_{s,h}) := a^{\mathrm{host_s}}\big(\Omega_{\delta,h}, \tilde{\mathcal{D}}^{\mathrm{ext}}_{s}, u_{s,h}, \varphi_{s,h}, \gamma_s\big), \tag{4.19b}$$

where we choose $\{\mathrm{host_b}, \mathrm{host_s}\} \subset \{\mathrm{sipg}, \mathrm{swipg}\}$ and define

$$\begin{aligned} a^{\mathrm{sipg}}\big(\mathcal{D}, \mathcal{D}, u_h, \varphi_h, \gamma\big) &:= \sum\nolimits_{K \in \mathcal{T}_h(\mathcal{D})} \int_K \big(\mathcal{D}\nabla_h u_h\big) \cdot \nabla_h \varphi_h \,\mathrm{d}x \\ &- \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})} \int_E \big\{\big(\mathcal{D}^{\mathrm{tr}} \nabla_h \varphi_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big\} [\![u_h]\!] + \big\{\big(\mathcal{D}\nabla_h u_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big\} [\![\varphi_h]\!] \,\mathrm{d}\sigma \\ &+ \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})} \frac{\gamma}{h_E} \int_E [\![u_h]\!][\![\varphi_h]\!] \,\mathrm{d}\sigma, \end{aligned}$$

$$\begin{aligned} a^{\mathrm{swipg}}\big(\mathcal{D}, \mathcal{D}, u_h, \varphi_h, \gamma\big) &:= \sum\nolimits_{K \in \mathcal{T}_h(\mathcal{D})} \int_K \big(\mathcal{D}\nabla_h u_h\big) \cdot \nabla_h \varphi_h \,\mathrm{d}x \\ &- \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})} \int_E \big\{\big(\mathcal{D}^{\mathrm{tr}} \nabla_h \varphi_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big\}_{\omega} [\![u_h]\!] + \big\{\big(\mathcal{D}\nabla_h u_h\big) \cdot \mathbf{n}_{\mathbf{E}}\big\}_{\omega} [\![\varphi_h]\!] \,\mathrm{d}\sigma \\ &+ \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})} \frac{\gamma}{h_E} \int_E \gamma_{\mathcal{D}}\, [\![u_h]\!][\![\varphi_h]\!] \,\mathrm{d}\sigma. \end{aligned}$$

Here, the terms $\gamma_b \in \mathbb{R}^{\geq 0}$ and $\gamma_s \in \mathbb{R}^{\geq 0}$ in equations (4.19) denote two given stabilization parameters which control penalization of jumps in the bulk part of the solution and in its surface part, respectively. See Section 3.2.2 for details on this mechanism. Furthermore, in both formulations, the term h_E is some

notion of the local mesh size at an internal face E. It represents the minimum element dimension orthogonal to E (Georgoulis et al., 2007, Section 3). The latter is characterized by the measure of cross sections of the adjacent elements K_E^+, K_E^-, considering only cross sections which are orthogonal to E. Following Georgoulis et al. (2007, Section 3), it can be approximated by setting

$$h_E := \min\{\mathrm{meas}_{\mathbb{R}^d}(K_E^+), \mathrm{meas}_{\mathbb{R}^d}(K_E^-)\} \,/\, \mathrm{meas}_{\mathbb{R}^{d-1}}(E),$$

particularly for anisotropically refined meshes. Note that our cut cell meshes $\mathcal{T}_h(\Omega_\hbar)$ and $\mathcal{T}_h(\Omega_{\delta,\hbar})$ may exhibit a considerable degree of anisotropy, especially $\mathcal{T}_h(\Omega_{\delta,\hbar})$ in case of a narrow band with a small δ. Therefore, this choice should be beneficial.

For a definition of the weighted average operator $\{\cdot\}_\omega$ and of the locally varying, diffusivity-dependent scaling function $\gamma_\mathcal{D}$ in the SWIPG formulation, please refer to the formulation's description in Section 3.2.3.

Numerical integration using local triangulations

Evaluating the integrals in Semidiscretization 4.2.5 and Scheme 4.2.6 requires computing integrals over the volume of each cut cell $K \in \mathcal{T}_h(\Omega_\hbar) \cup \mathcal{T}_h(\Omega_{\delta,\hbar})$ and over different parts $E \subset \partial K$ of its surface. Given that entities of the cut cell meshes might exhibit complicated shapes, quadrature rules based on interpolation functions are not directly applicable. Integration on the active meshes $\hat{\mathcal{T}}_h(\Omega_\hbar), \hat{\mathcal{T}}_h(\Omega_{\delta,\hbar})$ also does not work in a straightforward way, since the shape functions with restricted support (see Remark 4.2.4) are discontinuous across bulk domain boundaries.

In order to guarantee accurate evaluation of integals in an efficient manner, we construct quadrature rules for irregular-shaped elements using *local triangulations*. In particular, each cut cell K is subdivided into a disjoint set $\mathcal{T}_\hbar(K)$ of simple geometric objects, e.g. simplices and hyperrectangles, which are subsets of elements of the geometry mesh $\mathcal{T}_\hbar(\Omega_\Phi)$. For each of these simple geometric objects and for the entities which make up their surface, efficient Gaussian quadrature rules of arbitrary order are available. By summing up contributions of a proper set of quadrature points associated with these rules, we evaluate integrals over cut cells and parts of their surface.

Exploiting that the discrete level set function $\Phi_\hbar$ is piecewise (multi-)linear, local triangulations can be efficiently constructed by applying extensions of algorithms that are known as *marching cubes algorithms* (Lorensen and Cline, 1987; Lewiner et al., 2003). Based on the values of $\Phi_\hbar$ in the vertices of geometry mesh elements, the latter algorithms create a piecewise linear approximation of the $(d-1)$-dimensional zero level set of $\Phi_\hbar$, i.e. of the reconstructed hypersurface $\Gamma_\hbar$. The contribution of each element of the geometry mesh to the piecewise linear approximation of $\Gamma_\hbar$ is computed using basic algebraic operations and general geometrical data retrieved from precomputed lookup tables. *Extended* marching cubes algorithms that can be applied for construct-

ing a local triangulation $\mathcal{T}_{\hbar}(K)$ of a cut cell K are examined in Bastian and Engwer (2009); Engwer (2009); Heimann (2013); Engwer and Nüßing (2017). Those extensions additionally target d-dimensional subsets of geometry mesh elements, more precisely the two subsets which are characterized by positive values of $\Phi_{\hbar}$ and by negative values of $\Phi_{\hbar}$, respectively. Using the same efficient ideas as in the original marching cubes algorithms, approximations to those subsets are computed in terms of (possibly empty) collections of simple geometric objects.

In this thesis, we particularly use the extended marching cubes based local triangulation approach which is presented in Heimann (2013, Chapter 2). It allows for cut cells that are defined using multiple discrete level set functions. We exploit this feature to obtain local triangulations of elements in the cut cell mesh $\mathcal{T}_h(\Omega_{\delta,\hbar})$. Please note that the discrete narrow band $\Omega_{\delta,\hbar}$ can be described using two discrete level set functions of the form $\Phi_{\hbar} - c_{1/2}$ (with constants $c_1, c_2 \in \mathbb{R}$), whose zero level sets together yield $\partial\Omega_{\delta,\hbar}$.

Narrow band width scaling

We are interested in schemes that yield discrete solutions whose surface part converges toward the surface part of the solution of the original model problem (4.1) or (4.5). To obtain schemes of this kind, we scale the narrow band parameters δ_{in} and δ_{out} that have been introduced in equation (4.13) with the width h of the fundamental mesh. More precisely, we choose

$$\delta_{\text{in}} := \alpha_{\text{in}} \cdot h \quad \text{and} \quad \delta_{\text{out}} := \alpha_{\text{out}} \cdot h,$$
$$\alpha_{\text{in}}, \alpha_{\text{out}} \in \mathbb{R}^{>0} \text{ such that } \Gamma_{l,\hbar} \text{ non-empty and closed for } -\delta_{\text{in}} < l < \delta_{\text{out}}.$$

Designation: UDG and narrow band driven Eulerian SDG

In summary, the resulting numerical approach treats the surface part of the original model problem by dealing with surface differential operators from an Eulerian point of view, extending surface equations like (4.1c) and (4.5c) to a narrow band around the hypersurface under consideration, employing the UDG method for spatial discretization and scaling the width of the narrow band with the mesh width. This component of the approach can be referred to as *UDG and narrow band driven Eulerian SDG*. Here we use the term *Eulerian surface discontinuous Galerkin (Eulerian SDG)* to account for the fact that its basic concept is related to the Eulerian surface finite element method (Eulerian SFEM) that has been introduced in Dziuk and Elliott (2008), cf. Section 1.3.2.

4.2.3. Recovering discrete analogues to original conservation properties

We recall that Semidiscretization 4.2.5 is a specific spatial discretization of the modified system which employs an extended surface equation. As indicated in Section 4.2.1, we can hence not expect that the surface component of a corresponding solution pair exactly fulfills semidiscrete analogues to the

conservation properties which are embedded in the class of parabolic model problems (4.1). This even applies to global conservation properties. In fact, without suitable modification of equation (4.16b), semidiscrete analogues to properties (4.3b) and (4.4) can only be achieved approximatively by choosing a narrow band with a small δ. The latter is discussed later on in greater detail and will be underpinned by numerical results in Section 4.3.3. The same holds true for Scheme 4.2.6, a corresponding solution pair, and discrete analogues to the conservation properties which are embedded in the class of elliptic model problems (4.5).

Furthermore, in view of the data functions $f_{s,b}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar})$ and $f_s(u_{s,h}|_{\Gamma_\hbar})$, choosing extensions $f_{s,b}^{\text{ext}}(u_{b,h}, u_{s,h})$ and $f_s^{\text{ext}}(u_{s,h})$ that yield schemes with good convergence and conservation properties generally is a non-trivial task. What is commonly done in extension-based numerical methods for surface PDEs is to extend data functions by constructing extensions that are constant along normal lines through $\Gamma_\hbar$. An explicit construction of extensions of this type is not possible in our case, since our extensions depend on the unknown (extended) solution pair. Implicitly constructing extensions of this type is possible, e.g. by means of closest point projections onto $\Gamma_\hbar$ as in the closest point methods which we referred to in Section 1.3.2. However, implicitly constructed extensions usually come with some extra effort, e.g. constructing a closest point operator and performing function evaluations using this operator (especially the latter is a non-local process in general since the closest point on $\Gamma_\hbar$ of a point in some mesh element may lie in a different mesh element). Moreover, parts of the issue of recovering the original conservation properties still remain since it is an issue of discretizing the entire extended surface equation, not only an issue of choosing suitable extended data functions.

We address both issues at once by exploiting that the first and the last integral in (4.16b) and the last integral in (4.18b) can be approximated by associated integrals over the reconstructed hypersurface $\Gamma_\hbar$. These integrals are known from the sharp interface FEMs for surface PDEs which have been discussed in Section 1.3.2. In particular, we have the following theorem.

Theorem 4.2.7. *A function $g \in L^1(\Omega_{\delta,\hbar})$ with sufficient extra regularity to make the mapping $G\colon (-\delta_{\text{in}}, \delta_{\text{out}}) \to \mathbb{R}$, $l \mapsto \int_{\Gamma_{l,\hbar}} g \,\mathrm{d}\sigma$ a continuous function that is continuously extendable to the closed interval $[-\delta_{\text{in}}, \delta_{\text{out}}]$ satisfies*

$$\frac{1}{\delta} \int_{\Omega_{\delta,\hbar}} |\nabla_\hbar \Phi_\hbar| \, g \,\mathrm{d}x \quad \overset{\delta \to 0}{\longrightarrow} \quad \int_{\Gamma_\hbar} g\big|_{\Gamma_\hbar} \,\mathrm{d}\sigma. \tag{4.20}$$

Proof. Since $\Phi_\hbar$ is a piecewise (multi-)linear continuous function over some mesh, namely the geometry mesh $\mathcal{T}_\hbar(\Omega_\Phi)$, it is Lipschitz continuous (this can be shown using that $\mathcal{T}_\hbar(\Omega_\Phi)$ has a finite number of mesh elements and using that $|\nabla_\hbar \Phi_\hbar|$ is bounded on each of those elements). Thus, also the restriction of $\Phi_\hbar$ to $\Omega_{\delta,\hbar}$ is Lipschitz continuous. Therefore, together with $g \in L^1(\Omega_{\delta,\hbar})$, the requirements of the coarea formula (Federer, 1959, Theorem 3.1) are fulfilled,

which gives

$$\frac{1}{\delta}\int_{\Omega_{\delta,\hbar}} |\nabla_\hbar \Phi_\hbar|\, g \,\mathrm{d}x = \frac{1}{\delta}\int_{-\delta_{\mathrm{in}}}^{\delta_{\mathrm{out}}} \left(\int_{\Gamma_{l,\hbar}} g \,\mathrm{d}\sigma\right) \mathrm{d}l.$$

Using the assumption that the inner integral mapping G has a continuous extension to $[-\delta_{\mathrm{in}}, \delta_{\mathrm{out}}]$, according to the mean value theorem there exists a level $l_\delta \in (-\delta_{\mathrm{in}}, \delta_{\mathrm{out}})$ such that the right-hand side can be expressed as $G(l_\delta)$. Since G is continuous in $l = 0$ according to our assumption, noticing that $l_\delta \to 0$ as $\delta \to 0$ and employing that $\Gamma_{0,\hbar} = \Gamma_\hbar$ concludes the proof. □

Applying Theorem 4.2.7 to approximate the last integral in equation (4.16b) yields the following variant of Semidiscretization 4.2.5 for the class of parabolic model problems, which dispenses with the need for extended data functions $f_{s,b}^{\mathrm{ext}}(u_{b,h}, u_{s,h})$ and $f_s^{\mathrm{ext}}(u_{s,h})$.

Semidiscretization 4.2.8 (Variant A, parabolic problems)**.** *Given approximate initial values as in Semidiscretization 4.2.5, we seek a pair of functions* $u_{b,h}\colon \Omega_\hbar \times [0,T] \to \mathbb{R}$ *and* $u_{s,h}\colon \Omega_{\delta,\hbar} \times [0,T] \to \mathbb{R}$ *with* $u_{b,h}(\cdot,t) \in V_{b,h}(\Omega_\hbar)$ *and* $u_{s,h}(\cdot,t) \in V_{s,h}(\Omega_{\delta,\hbar})$*, such that for each* $t \in (0,T]$ *equation* (4.16a) *holds true and*

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{\delta}\int_{\Omega_{\delta,\hbar}} |\nabla_\hbar \Phi_\hbar|\, u_{s,h}\, \varphi_{s,h} \,\mathrm{d}x\right)\\ + \frac{1}{\delta}\, a_s(u_{s,h}, \varphi_{s,h}) = c_s(u_{b,h}, u_{s,h}, \varphi_{s,h}) \quad \forall \varphi_{s,h} \in V_{s,h}(\Omega_{\delta,\hbar}).\end{aligned} \tag{4.21}$$

Here, the modified right-hand side is written in terms of a potentially nonlinear form $c_s\colon V_{b,h}(\Omega_\hbar) \times V_{s,h}(\Omega_{\delta,\hbar}) \times V_{s,h}(\Omega_{\delta,\hbar}) \to \mathbb{R}$ *defined by*

$$c_s(u_{b,h}, u_{s,h}, \varphi_{s,h}) := \int_{\Gamma_\hbar} \left(f_{s,b}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar}) + f_s(u_{s,h}|_{\Gamma_\hbar})\right) \varphi_{s,h}\big|_{\Gamma_\hbar} \,\mathrm{d}\sigma, \tag{4.22}$$

and a_s *is the bilinear form that has been defined in equation* (4.19b)*.*

By treating equation (4.18b) in an analogue way, we obtain the following variant of Scheme 4.2.6 for the class of elliptic model problems, which has the same advantage regarding data functions. As we will show, it furthermore recovers discrete analogues to the conservation properties which are embedded in the class of elliptic model problems.

Scheme 4.2.9 (Variant A, elliptic problems)**.** *Find a pair of discrete functions* $(u_{b,h}, u_{s,h}) \in V_{b,h}(\Omega_\hbar) \times V_{s,h}(\Omega_{\delta,\hbar})$*, such that equation* (4.18a) *holds true for all* $\varphi_{b,h} \in V_{b,h}(\Omega_\hbar)$ *and*

$$\frac{1}{\delta}\, a_s(u_{s,h}, \varphi_{s,h}) = c_s(u_{b,h}, u_{s,h}, \varphi_{s,h}) \quad \forall \varphi_{s,h} \in V_{s,h}(\Omega_{\delta,\hbar}), \tag{4.23}$$

where a_s *and* c_s *are the same forms as in Semidiscretization 4.2.8.*

Finally, applying the approximation based on (4.20) additionally to the first integral in equation (4.16b) yields another variant of Semidiscretization 4.2.5, which is even more beneficial than Semidiscretization 4.2.8. As we will show, solutions which are obtained using the following second variant exactly fulfill semidiscrete analogues to the global conservation properties that are intrinsic to all parabolic model problems from class (4.1), and they satisfy semidiscrete analogues to the corresponding local conservation properties.

Semidiscretization 4.2.10 (Variant B, parabolic problems)**.** *Given approximate initial values as in Semidiscretization 4.2.5, we seek a pair of functions* $u_{b,h}\colon \Omega_\hbar \times [0,T] \to \mathbb{R}$ *and* $u_{s,h}\colon \Omega_{\delta,\hbar} \times [0,T] \to \mathbb{R}$ *with* $u_{b,h}(\cdot,t) \in V_{b,h}(\Omega_\hbar)$ *and* $u_{s,h}(\cdot,t) \in V_{s,h}(\Omega_{\delta,\hbar})$*, such that for each* $t \in (0,T]$ *equation* (4.16a) *holds true and*

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{\Gamma_\hbar} u_{s,h}\big|_{\Gamma_\hbar}\, \varphi_{s,h}\big|_{\Gamma_\hbar}\, \mathrm{d}\sigma\right) + \frac{1}{\delta}\, a_s(u_{s,h},\varphi_{s,h}) = c_s(u_{b,h},u_{s,h},\varphi_{s,h}) \quad \forall \varphi_{s,h} \in V_{s,h}(\Omega_{\delta,\hbar}), \tag{4.24}$$

where, again, a_s *and* c_s *are the same forms as in Semidiscretization 4.2.8.*

Global conservation properties

Next, we investigate our semidiscretizations and schemes regarding the global conservation properties which have been discussed in Section 4.1. Semidiscrete analogues to properties (4.3), which we want to be satisfied by semidiscrete solution pairs to model problems from class (4.1), can be formulated as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_\hbar} u_{b,h}\,\mathrm{d}x = \int_{\Gamma_\hbar} f_{b,s}(u_{b,h},u_{s,h}|_{\Gamma_\hbar})\,\mathrm{d}\sigma + \int_{\Omega_\hbar} f_b(u_{b,h})\,\mathrm{d}x, \tag{4.25a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Gamma_\hbar} u_{s,h}\big|_{\Gamma_\hbar}\,\mathrm{d}\sigma = \int_{\Gamma_\hbar} f_{s,b}(u_{b,h},u_{s,h}|_{\Gamma_\hbar})\,\mathrm{d}\sigma + \int_{\Gamma_\hbar} f_s(u_{s,h}|_{\Gamma_\hbar})\,\mathrm{d}\sigma. \tag{4.25b}$$

Provided that a solution pair $(u_{b,h},u_{s,h})$ satisfies these equalities, the derivative of the total amount $m_h(t) := \int_{\Omega_\hbar} u_{b,h}\,\mathrm{d}x + \int_{\Gamma_\hbar} u_{s,h}|_{\Gamma_\hbar}\,\mathrm{d}\sigma$ of the semidiscrete system's quantities automatically fulfills the following semidiscrete analogue to property (4.4):

$$\frac{\mathrm{d}}{\mathrm{d}t}\, m_h(t) = \int_{\Gamma_\hbar} f_{b,s}(u_{b,h},u_{s,h}|_{\Gamma_\hbar}) + f_{s,b}(u_{b,h},u_{s,h}|_{\Gamma_\hbar})\,\mathrm{d}\sigma + \int_{\Omega_\hbar} f_b(u_{b,h})\,\mathrm{d}x + \int_{\Gamma_\hbar} f_s(u_{s,h}|_{\Gamma_\hbar})\,\mathrm{d}\sigma. \tag{4.26}$$

Similarly, discrete analogues to the global conservation properties (4.7), which we want to be satisfied by discrete solution pairs $(u_{b,h},u_{s,h})$ to model

problems from class (4.5), can be formulated as

$$0 = \int_{\Gamma_\hbar} f_{b,s}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar}) \, \mathrm{d}\sigma + \int_{\Omega_\hbar} f_b(u_{b,h}) \, \mathrm{d}x, \tag{4.27a}$$

$$0 = \int_{\Gamma_\hbar} f_{s,b}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar}) \, \mathrm{d}\sigma + \int_{\Gamma_\hbar} f_s(u_{s,h}|_{\Gamma_\hbar}) \, \mathrm{d}\sigma. \tag{4.27b}$$

Considering the bulk part of a solution pair, the semidiscretizations and schemes which we have considered are based on equation (4.16a) and equation (4.18a), respectively. From those equations, they recover properties (4.25a) and (4.27a) in a straightforward way. In particular, the characteristic function $\mathbb{1}_{\Omega_\hbar}$ of $\Omega_\hbar$ is part of the discrete function spaces $V_{b,h}(\Omega_\hbar)$ and hence an admissible test function $\varphi_{b,h}$. Moreover, we have $a_b\big(u_{b,h}, \mathbb{1}_{\Omega_\hbar}\big) = 0$ for the bilinear form defined in equation (4.19a) and both choices of host DG formulations. Therefore, testing with $\varphi_{b,h} = \mathbb{1}_{\Omega_\hbar}$ in equation (4.16a) yields property (4.25a), and testing with $\varphi_{b,h} = \mathbb{1}_{\Omega_\hbar}$ in equation (4.18a) yields property (4.27a).

Considering the surface part of a solution pair, we can proceed analogously. Being part of the discrete function spaces $V_{s,h}(\Omega_{\delta,\hbar})$ which are employed in all of our semidiscretizations and schemes, the characteristic function $\mathbb{1}_{\Omega_{\delta,\hbar}}$ of $\Omega_{\delta,\hbar}$ is an admissible test function $\varphi_{s,h}$, and we have $a_s\big(u_{s,h}, \mathbb{1}_{\Omega_{\delta,\hbar}}\big) = 0$ for the bilinear form defined in equation (4.19b) and both choices of host DG formulations.

Nevertheless, as expected, we can not show that all semidiscretizations and schemes recover the global conservation property which we are wishing for. Testing with $\varphi_{s,h} = \mathbb{1}_{\Omega_{\delta,\hbar}}$ in equations (4.16b), (4.18b) and (4.21) rather yields equalities which approximate property (4.25b) or property (4.27b), provided that the narrow band parameter δ is chosen sufficiently small. The latter can be shown using Theorem 4.2.7.

However, testing with $\varphi_{s,h} = \mathbb{1}_{\Omega_{\delta,\hbar}}$ in equation (4.24) yields property (4.25b), and testing with $\varphi_{s,h} = \mathbb{1}_{\Omega_{\delta,\hbar}}$ in equation (4.23) yields property (4.27b). We hence have proven, inter alia, the two theorems which read as follows.

Theorem 4.2.11 (Semidiscrete solutions – global conservation properties)**.** *Each semidiscrete solution pair* $(u_{b,h}, u_{s,h})$ *which is obtained using Semidiscretization 4.2.10 satisfies properties* (4.25)*, i.e. the semidiscrete analogues to global conservation properties* (4.3)*. Moreover, it satisfies property* (4.26)*, i.e. the corresponding semidiscrete analogue to property* (4.4)*.*

Theorem 4.2.12 (Discrete solutions – global conservation properties)**.** *With Scheme 4.2.9, we obtain discrete solution pairs* $(u_{b,h}, u_{s,h})$ *which satisfy properties* (4.27)*, i.e. the discrete analogues to global conservation properties* (4.7)*.*

Local conservation properties

Due to the discontinuity of the discrete function spaces $V_{b,h}(\Omega_\hbar)$ and of the discrete function spaces $V_{s,h}(\Omega_{\delta,\hbar})$ which we are dealing with in all of our

semidiscretizations and schemes, Semidiscretization 4.2.10 and Scheme 4.2.9 recover analogues to conservation laws (4.2) and conservation laws (4.6), respectively, not only in the global sense discussed above, but also in some local sense. Similar to our considerations for fitted DG methods in Section 3.2.5, we use that our spaces allow for choosing the characteristic functions $\mathbb{1}_K$ of cut cell mesh elements K as test functions and obtain the following theorem.

Theorem 4.2.13 (Semidiscrete solutions – local conservation properties)**.** *Each semidiscrete solution pair $(u_{b,h}, u_{s,h})$ which is obtained using Semidiscretization 4.2.10 satisfies*

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_K u_{b,h}\ \mathrm{d}x = -\int_{\partial K} \hat{\boldsymbol{\sigma}}_{\mathbf{b,h},\boldsymbol{\partial}\mathbf{K}}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar})\cdot \mathbf{n}_{\boldsymbol{\partial}\mathbf{K}}\ \mathrm{d}\sigma + \int_K f_b(u_{b,h})\ \mathrm{d}x \tag{4.28a}$$

for all cut cells $K \in \mathcal{T}_h(\Omega_\hbar)$, and

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Gamma_\hbar\cap K} u_{s,h}\big|_{\Gamma_\hbar}\ \mathrm{d}\sigma = &-\frac{1}{\delta}\int_{\partial K}\hat{\boldsymbol{\sigma}}^{\mathbf{ext}}_{\mathbf{s,h},\boldsymbol{\partial}\mathbf{K}}(u_{s,h})\cdot\mathbf{n}_{\boldsymbol{\partial}\mathbf{K}}\ \mathrm{d}\sigma\\ &+\int_{\Gamma_\hbar\cap K} f_{s,b}(u_{b,h},u_{s,h}|_{\Gamma_\hbar}) + f_s(u_{s,h}|_{\Gamma_\hbar})\ \mathrm{d}\sigma\end{aligned} \tag{4.28b}$$

for all cut cells $K \in \mathcal{T}_h(\Omega_{\delta,\hbar})$. Here, we define a collection of local numerical fluxes $\{\hat{\boldsymbol{\sigma}}_{\mathbf{b,h},\boldsymbol{\partial}\mathbf{K}}\}_{K\in\mathcal{T}_h(\Omega_\hbar)}$ for the bulk part of the problem and a collection of local numerical fluxes $\{\hat{\boldsymbol{\sigma}}^{\mathbf{ext}}_{\mathbf{s,h},\boldsymbol{\partial}\mathbf{K}}\}_{K\in\mathcal{T}_h(\Omega_{\delta,\hbar})}$ for the surface part of the problem in the following way. Considering the faces $E \subset \partial K$ of an arbitrary but fixed cut cell $K \in \mathcal{T}_h(\Omega_\hbar)$, we set

$$\hat{\boldsymbol{\sigma}}_{\mathbf{b,h},\boldsymbol{\partial}\mathbf{K}}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar})\big|_E := \begin{cases}\{-\mathcal{D}_b\nabla_h u_{b,h}\} + \frac{\gamma_b}{h_E}[\![u_{b,h}]\!]\,\mathbf{n}_{\mathbf{E}} & E\in\mathcal{E}_h^{\mathrm{int}}(\Omega_\hbar),\\ -f_{b,s}(u_{b,h},u_{s,h}|_{\Gamma_\hbar})\,\mathbf{n}_{\mathbf{E}} & E\subset\Gamma_\hbar=\partial\Omega_\hbar\end{cases}$$

if $\mathrm{host_b} = \mathrm{sipg}$. *If* $\mathrm{host_b} = \mathrm{swipg}$ *is chosen, the term for the first case on the right-hand side of the latter definition needs to be replaced by*

$$\big\{-\mathcal{D}_b\nabla_h u_{b,h}\big\}_\omega + \frac{\gamma_b\cdot\gamma_{\mathcal{D}_b}}{h_E}[\![u_{b,h}]\!]\,\mathbf{n}_{\mathbf{E}}.$$

Furthermore, considering the faces of fixed cut cells $K \in \mathcal{T}_h(\Omega_{\delta,\hbar})$, we set

$$\hat{\boldsymbol{\sigma}}^{\mathbf{ext}}_{\mathbf{s,h},\boldsymbol{\partial}\mathbf{K}}(u_{s,h})\big|_E := \begin{cases}\{-\tilde{\mathcal{D}}_s^{\mathrm{ext}}\nabla_h u_{s,h}\} + \frac{\gamma_s}{h_E}[\![u_{s,h}]\!]\,\mathbf{n}_{\mathbf{E}} & E\in\mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar}),\\ 0\cdot\mathbf{n}_{\mathbf{E}} & E\subset\partial\Omega_{\delta,\hbar}\end{cases}$$

in the case $\mathrm{host_s} = \mathrm{sipg}$, *replacing the term for the first case on the right-hand side of this definition by*

$$\big\{-\tilde{\mathcal{D}}_s^{\mathrm{ext}}\nabla_h u_{s,h}\big\}_\omega + \frac{\gamma_s\cdot\gamma_{\tilde{\mathcal{D}}_s^{\mathrm{ext}}}}{h_E}[\![u_{s,h}]\!]\,\mathbf{n}_{\mathbf{E}}$$

if $\mathrm{host_s} = \mathrm{swipg}$ *is chosen.*

Proof. The characteristic function $\mathbb{1}_K$ of a cut cell $K \in \mathcal{T}_h(\Omega_\text{f})$ is contained in the discrete function spaces $V_{b,h}(\Omega_\text{f})$. It hence is an admissible test function $\varphi_{b,h}$ in equation (4.16a). Likewise, the characteristic function of a cut cell $K \in \mathcal{T}_h(\Omega_{\delta,\text{f}})$ is contained in the discrete function spaces $V_{s,h}(\Omega_{\delta,\text{f}})$ and thus an admissible test function $\varphi_{s,h}$ in equation (4.24). Moreover, in the case $\text{host}_\text{b} = \text{sipg}$, we have the identity

$$\begin{aligned}
&a_b(u_{b,h}, \mathbb{1}_K) - \int_{\Gamma_\text{f} \cap \partial K} f_{b,s}(u_{b,h}, u_{s,h}|_{\Gamma_\text{f}}) \,\mathrm{d}\sigma \\
&\quad = \quad \sum\nolimits_{E \in \mathcal{E}_h^\text{int}(\Omega_\text{f}),\, E \subset \partial K} \int_E \Big(-\big\{ \big(\mathcal{D}_b \nabla_h u_{b,h} \big) \cdot \mathbf{n_E} \big\} + \frac{\gamma_b}{h_E} [\![u_{b,h}]\!] \Big) [\![\mathbb{1}_K]\!] \,\mathrm{d}\sigma \\
&\qquad + \int_{\Gamma_\text{f} \cap \partial K} -f_{b,s}(u_{b,h}, u_{s,h}|_{\Gamma_\text{f}}) \,\mathrm{d}\sigma \\
&\quad = \quad \sum\nolimits_{E \in \mathcal{E}_h^\text{int}(\Omega_\text{f}),\, E \subset \partial K} \int_E \big(\hat{\boldsymbol{\sigma}}_{\mathbf{b,h,\partial K}}(u_{b,h}, u_{s,h}|_{\Gamma_\text{f}}) \cdot \mathbf{n_E} \big) [\![\mathbb{1}_K]\!] \,\mathrm{d}\sigma \\
&\qquad + \int_{\Gamma_\text{f} \cap \partial K} \hat{\boldsymbol{\sigma}}_{\mathbf{b,h,\partial K}}(u_{b,h}, u_{s,h}|_{\Gamma_\text{f}}) \cdot \mathbf{n_E} \,\mathrm{d}\sigma \\
&\quad = \quad \int_{\partial K} \hat{\boldsymbol{\sigma}}_{\mathbf{b,h,\partial K}}(u_{b,h}, u_{s,h}|_{\Gamma_\text{f}}) \cdot \mathbf{n_{\partial K}} \,\mathrm{d}\sigma
\end{aligned}$$

for an arbitrary cut cell $K \in \mathcal{T}_h(\Omega_\text{f})$, and, in the case $\text{host}_\text{s} = \text{sipg}$, we have the identity

$$\begin{aligned}
&a_s(u_{s,h}, \mathbb{1}_K) \\
&\quad = \sum\nolimits_{E \in \mathcal{E}_h^\text{int}(\Omega_{\delta,\text{f}}),\, E \subset \partial K} \int_E \Big(-\big\{ \big(\tilde{\mathcal{D}}_s^\text{ext} \nabla_h u_{s,h} \big) \cdot \mathbf{n_E} \big\} + \frac{\gamma_s}{h_E} [\![u_{s,h}]\!] \Big) [\![\mathbb{1}_K]\!] \,\mathrm{d}\sigma \\
&\quad = \sum\nolimits_{E \in \mathcal{E}_h^\text{int}(\Omega_{\delta,\text{f}}),\, E \subset \partial K} \int_E \big(\hat{\boldsymbol{\sigma}}^{\mathbf{ext}}_{\mathbf{s,h,\partial K}}(u_{s,h}) \cdot \mathbf{n_E} \big) [\![\mathbb{1}_K]\!] \,\mathrm{d}\sigma \\
&\quad = \int_{\partial K} \hat{\boldsymbol{\sigma}}^{\mathbf{ext}}_{\mathbf{s,h,\partial K}}(u_{s,h}) \cdot \mathbf{n_{\partial K}} \,\mathrm{d}\sigma
\end{aligned}$$

for an arbitrary cut cell $K \in \mathcal{T}_h(\Omega_{\delta,\text{f}})$. In the cases $\text{host}_\text{b} = \text{swipg}$ and $\text{host}_\text{s} = \text{swipg}$, both identities can be derived completely analogously using the corresponding definitions of the local numerical fluxes. Therefore, testing with $\varphi_{b,h} = \mathbb{1}_K$ in equation (4.16a) yields property (4.28a), and testing with $\varphi_{s,h} = \mathbb{1}_K$ in equation (4.24) yields property (4.28b). □

By applying the same arguments to Scheme 4.2.9, we furthermore get the following theorem.

Theorem 4.2.14 (Discrete solutions – local conservation properties)**.** *With Scheme 4.2.9, we obtain discrete solution pairs $(u_{b,h}, u_{s,h})$ which satisfy*

$$0 = -\int_{\partial K} \hat{\boldsymbol{\sigma}}_{\mathbf{b,h,\partial K}}(u_{b,h}, u_{s,h}|_{\Gamma_\text{f}}) \cdot \mathbf{n_{\partial K}} \,\mathrm{d}\sigma + \int_K f_b(u_{b,h}) \,\mathrm{d}x \tag{4.29a}$$

for all cut cells $K \in \mathcal{T}_h(\Omega_\hbar)$, and

$$0 = -\frac{1}{\delta}\int_{\partial K} \hat{\boldsymbol{\sigma}}^{\mathbf{ext}}_{\mathbf{s,h},\boldsymbol{\partial K}}(u_{s,h}) \cdot \mathbf{n}_{\boldsymbol{\partial K}} \,\mathrm{d}\sigma + \int_{\Gamma_\hbar \cap K} f_{s,b}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar}) + f_s(u_{s,h}|_{\Gamma_\hbar}) \,\mathrm{d}\sigma \tag{4.29b}$$

for all cut cells $K \in \mathcal{T}_h(\Omega_{\delta,\hbar})$. Here, the two collections of local numerical fluxes are the same as in Theorem 4.2.13.

Properties (4.28) can be seen as semidiscrete analogues to local conservation properties (4.2), and properties (4.29) can be seen as discrete analogues to local conservation properties (4.6). In particular, let us take a close look at the local numerical fluxes that are defined in Theorem 4.2.13 for either of the two considered host DG formulations. It is easy to see that, if the bulk diffusivity tensor $\mathcal{D}_b$ is continuous across internal faces and the method converges to some solution pair (u_b, u_s), the local numerical flux value $\hat{\boldsymbol{\sigma}}_{\mathbf{b,h},\boldsymbol{\partial K}}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar})$ in equation (4.28a) and in equation (4.29a) converges to the physical flux $\mathbf{q_b}$ which is considered in Section 4.1.1. Moreover, it can be shown using a codimension 1 analogue of Theorem 4.2.7 that, if the surface diffusivity tensor $\mathcal{D}_s$ is continuous across internal faces and the method converges to some solution pair (u_b, u_s), the term

$$-\frac{1}{\delta}\int_{\partial K} \hat{\boldsymbol{\sigma}}^{\mathbf{ext}}_{\mathbf{s,h},\boldsymbol{\partial K}}(u_{s,h}) \cdot \mathbf{n}_{\boldsymbol{\partial K}} \,\mathrm{d}\sigma = \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\Omega_{\delta,\hbar}),\, E \subset \partial K} -\frac{1}{\delta}\int_E \hat{\boldsymbol{\sigma}}^{\mathbf{ext}}_{\mathbf{s,h},\boldsymbol{\partial K}}(u_{s,h}) \cdot \mathbf{n}_{\boldsymbol{\partial K}} \,\mathrm{d}\sigma$$

in equations (4.28b) and (4.29b) approximates an integral of the form

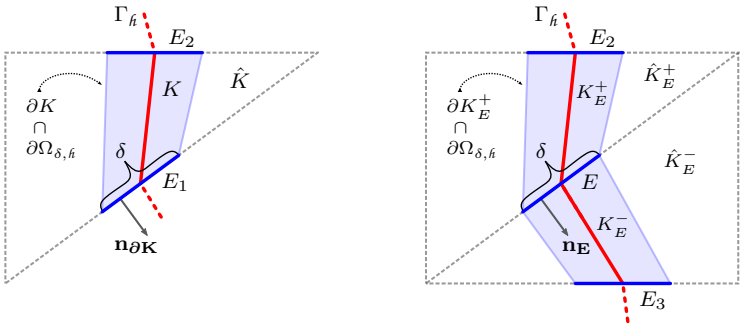
$$-\int_{\partial M} \mathbf{q_s} \cdot \boldsymbol{\mu}_{\boldsymbol{\partial M}} \,\mathrm{d}\varsigma \qquad \text{with} \qquad M := \Gamma \cap K$$

for small values of the mesh widths h and $\hbar$. Here, the field $\mathbf{q_s} := -\mathcal{D}_s \nabla_\Gamma u_s$ is the physical surface flux that is considered in Section 4.1.1. See Figure 4.5a for an illustration of the geometrical setting of equations (4.28b) and (4.29b).

4.2.4. Stabilization strategies with respect to the surface part of the solution

Next, we have a look at the surface part $u_{s,h}$ of a (semi-)discrete solution. Even though we are only interested in its restriction $u_{s,h}|_{\Gamma_\hbar}$, we obtain a function $u_{s,h}$ that lives on the reconstructed narrow band $\Omega_{\delta,\hbar}$ around $\Gamma_\hbar$ and hence has a gradient in the ambient Cartesian space.

For Semidiscretization 4.2.5 and Scheme 4.2.6, the normal component of this gradient can be controlled by the choice of the extended initial values, where required, and by the choice of the extended data functions which result in the



(a) Cut cell centered view. (b) Internal face centered view.

Figure 4.5.: The geometrical setting of spatial discretization of the surface part of the problem using UDG and narrow band driven Eulerian SDG, illustrated using triangular meshes with $\mathcal{T}_h(\Omega_\Phi) = \mathcal{T}_\ell(\Omega_\Phi)$.

terms $\tilde{\mathcal{D}}_s^{\text{ext}}$, $\tilde{f}_{s,b}^{\text{ext}}(u_{b,h}, u_{s,h})$ and $\tilde{f}_s^{\text{ext}}(u_{s,h})$. With the modifications made in Semidiscretization 4.2.8, Scheme 4.2.9 and Semidiscretization 4.2.10, however, we loose control over this property because we are effectively solving a set of surface diffusion equations with a vanishing right-hand side on the set of discrete level sets $\Gamma_{l,\ell}$ with $l \neq 0$. Meanwhile, we should avoid steep gradients normal to Γ_ℓ since they are expected to decrease the robustness of the method.

Therefore, we optionally stabilize these variants by adding a term that penalizes the normal component $(\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell})\boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}$ of the gradient of $u_{s,h}$ in the ambient Cartesian space. This idea is related to a strategy pursued by various extension-based numerical approaches to surface PDEs, namely searching for a solution $u_{s,h}$ that is the normally constant extension of a solution $u_{s,h}|_{\Gamma_\ell}$ on the reconstructed hypersurface Γ_ℓ. See Xu and Zhao (2003, Section 2), and Ruuth and Merriman (2008, Section 2.1), for instance.

More precisely, after choosing some stabilization term $j_s(u_{s,h}, \varphi_{s,h})$ which implements the idea mentioned above, we optionally add a scaled version of this term to the left-hand side of equations (4.21), (4.23) or (4.24), respectively. The particular scaling factor which we use is $\frac{1}{\delta}$. For instance, augmenting the bilinear form in equation (4.23) this way results in the following variant of Scheme 4.2.9.

Scheme 4.2.15 (Variant A stabilized, elliptic problems)**.** *We look for a pair of discrete functions* $(u_{b,h}, u_{s,h}) \in V_{b,h}(\Omega_\ell) \times V_{s,h}(\Omega_{\delta,\ell})$*, such that equation* (4.18a) *holds true for all* $\varphi_{b,h} \in V_{b,h}(\Omega_\ell)$ *and*

$$\frac{1}{\delta}\left[a_s + j_s\right](u_{s,h}, \varphi_{s,h}) = c_s(u_{b,h}, u_{s,h}, \varphi_{s,h}) \quad \forall \varphi_{s,h} \in V_{s,h}(\Omega_{\delta,\ell}), \tag{4.30}$$

where, as before, a_s *and* c_s *are the same forms as in Semidiscretization 4.2.8, and* $j_s \colon V_{s,h}(\Omega_{\delta,\ell}) \times V_{s,h}(\Omega_{\delta,\ell}) \to \mathbb{R}$ *is a form that represents some suitably chosen stabilization term.*

Two specific choices of the stabilization term $j_s(u_{s,h}, \varphi_{s,h})$ will be discussed next.

Full gradient stabilization

The first stabilization term which we investigate in this thesis is based on the idea of using full gradients as a stabilization mechanism. The latter has been introduced in related work on continuous Galerkin schemes (Deckelnick et al., 2014). We define the stabilization term $j_s(u_{s,h}, \varphi_{s,h}) = j_s^{\mathrm{fg}}(u_{s,h}, \varphi_{s,h})$ with

$$\begin{aligned} &j_s^{\mathrm{fg}}(u_{s,h}, \varphi_{s,h}) \\ &\quad := \int_{\Omega_{\delta,\ell}} \big(|\nabla_\ell \Phi_\ell| \mathcal{D}_s^{\mathrm{ext}} (\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}\big) \cdot (\nabla_h \varphi_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell} \,\mathrm{d}x \\ &\qquad + \int_{\Omega_{\delta,\ell}} \big(|\nabla_\ell \Phi_\ell| \mathcal{D}_s^{\mathrm{ext}} \mathcal{P}^{\Phi_\ell} \nabla_h u_{s,h}\big) \cdot (\nabla_h \varphi_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell} \,\mathrm{d}x \\ &\qquad + \int_{\Omega_{\delta,\ell}} \big(|\nabla_\ell \Phi_\ell| \mathcal{D}_s^{\mathrm{ext}} (\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}\big) \cdot \mathcal{P}^{\Phi_\ell} \nabla_h \varphi_{s,h} \,\mathrm{d}x. \end{aligned} \tag{4.31}$$

This term indeed penalizes the normal component $(\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}$ of the gradient of $u_{s,h}$ in the ambient Cartesian space. By imagining $\mathcal{D}_s^{\mathrm{ext}}$ as a constant extension of some constant diffusivity $\mathcal{D}_s \in \mathbb{R}$, it can be seen that $j_s^{\mathrm{fg}}(u_{s,h}, \varphi_{s,h})$ particularly penalizes some weighted $L^2(\Omega_{\delta,\ell})$-norm of this normal component. Beneficially, no extra parameter is required in the course of this. The strength of penalization is controlled automatically by the locally varying factor $|\nabla_\ell \Phi_\ell| \mathcal{D}_s^{\mathrm{ext}}$.

It should be noted that, for both choices of host DG formulations which we are considering in this chapter, the bilinear form $a_s(u_{s,h}, \varphi_{s,h})$ that has been defined in equation (4.19b) contains the volume integral

$$\begin{aligned} &\sum\nolimits_{K \in \mathcal{T}_h(\Omega_{\delta,\ell})} \int_K \big(\tilde{\mathcal{D}}_s^{\mathrm{ext}} \nabla_h u_{s,h}\big) \cdot \nabla_h \varphi_{s,h} \,\mathrm{d}x \\ &\quad = \int_{\Omega_{\delta,\ell}} \big(\tilde{\mathcal{D}}_s^{\mathrm{ext}} \nabla_h u_{s,h}\big) \cdot \nabla_h \varphi_{s,h} \,\mathrm{d}x \\ &\quad = \int_{\Omega_{\delta,\ell}} \big(|\nabla_\ell \Phi_\ell| \mathcal{D}_s^{\mathrm{ext}} \mathcal{P}^{\Phi_\ell} \nabla_h u_{s,h}\big) \cdot \mathcal{P}^{\Phi_\ell} \nabla_h \varphi_{s,h} \,\mathrm{d}x. \end{aligned} \tag{4.32}$$

Augmenting the bilinear form $a_s(u_{s,h}, \varphi_{s,h})$ with stabilization term (4.31) corresponds to replacing the integral (4.32) by

$$\int_{\Omega_{\delta,\ell}} \big(|\nabla_\ell \Phi_\ell| \mathcal{D}_s^{\mathrm{ext}} \nabla_h u_{s,h}\big) \cdot \nabla_h \varphi_{s,h} \,\mathrm{d}x,$$

i.e. it corresponds to using full gradients instead of projected gradients in the volume integral. Therefore, we call this technique *full gradient stabilization*.

Given this name, we would like to point out that other gradients in the bilinear form $a_s(u_{s,h}, \varphi_{s,h})$ remain projected gradients.

We emphasize that this stabilization strategy is easy to implement and particularly attractive since it does not require an extra parameter. Furthermore, the full gradient stabilization mechanism does not affect conservation properties, i.e. our stabilized semidiscretizations and schemes, such as Scheme 4.2.15, inherit recovered conservation properties from their unstabilized counterparts. Note that $\varphi_{s,h}$ contributes to stabilization term (4.31) only in terms of $\nabla_h \varphi_{s,h}$, and note that we have $j_s^{\mathrm{fg}}\big(u_{s,h}, \mathbb{1}_{\Omega_{\delta,\ell}}\big) = 0$ and $j_s^{\mathrm{fg}}\big(u_{s,h}, \mathbb{1}_K\big) = 0$ for all cut cells $K \in \mathcal{T}_h(\Omega_{\delta,\ell})$.

Normal penalty stabilization

Another idea which we investigate in this thesis, is to use the stabilization term $j_s(u_{s,h}, \varphi_{s,h}) = j_s^{\mathrm{np}}(u_{s,h}, \varphi_{s,h})$ with

$$\begin{aligned} &j_s^{\mathrm{np}}(u_{s,h}, \varphi_{s,h}) \\ &\quad := h\,\gamma_{\mathrm{np}} \int_{\Omega_{\delta,\ell}} (\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell} \cdot (\nabla_h \varphi_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell} \,\mathrm{d}x \\ &\quad = h\,\gamma_{\mathrm{np}} \int_{\Omega_{\delta,\ell}} (\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell})(\nabla_h \varphi_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}) \,\mathrm{d}x. \end{aligned} \tag{4.33}$$

More explicitly than with the full gradient stabilization term (4.31), it can be seen that term (4.33) penalizes the normal component $(\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_\ell}$ of the gradient of $u_{s,h}$ in the ambient Cartesian space. More precisely, it penalizes large values with respect to $\|\cdot\|_{L^2(\Omega_{\delta,\ell})}$. Here, the strength of penalization is controlled by the globally constant factor $h\gamma_{\mathrm{np}}$, where h denotes the width of the fundamental mesh, as usual, and $\gamma_{\mathrm{np}} \in \mathbb{R}^{\geq 0}$ is an extra parameter that needs to be supplied by the user.

Since the normal component of the gradient is penalized explicitly, we call this technique *normal penalty stabilization*. Investigating term (4.33) was initiated by private discussion with André Massing (Umeå University) during his visit in Münster in February 2016. While writing up the thesis, the author learned that a similar mechanism has recently been investigated in the context of trace FEMs in Burman et al. (2016a) and in Grande et al. (2016), where it has been given the names *normal gradient stabilization* and *normal derivative volume stabilization*, respectively.

The stabilization strategy is nearly as easy to implement as full gradient stabilization. Rather than simplifying existing terms in the bilinear form of the host DG formulation, the stabilization term is an extra term which needs to be computed. These computations can be performed in a straightforward way, though. Regarding conservation properties, the same considerations hold true as in the case of full gradient stabilization.

Scaled versions of those stabilization terms, and the limit $h \to 0$

For both stabilization mechanisms, we now formally study the effect of the scaled term $\frac{1}{\delta} j_s(u_{s,h}, \varphi_{s,h})$ that is actually used in our semidiscretizations and schemes. In particular, we investigate the limit $h \to 0$.

First, we consider the full gradient stabilization term $j_s^{\mathrm{fg}}(u_{s,h}, \varphi_{s,h})$. If no scaling is performed, this term penalizes some weighted $L^2(\Omega_{\delta,h})$-norm of the gradient's normal component, as discussed above. Analogous considerations for the scaled version $\frac{1}{\delta} j_s^{\mathrm{fg}}(u_{s,h}, \varphi_{s,h})$ show that, in the limit $h \to 0$, the scaled version penalizes the $L^2(\Gamma_h)$-norm of the gradient's normal component. Note that $\delta = (\alpha_{\mathrm{in}} + \alpha_{\mathrm{out}}) \cdot h$ goes to zero as $h \to 0$, and that

$$\begin{aligned}\frac{1}{\delta} j_s^{\mathrm{fg}}(u_{s,h}, \varphi_{s,h}) \quad \overset{\delta \to 0}{\longrightarrow} \quad & \int_{\Gamma_h} \left(\mathcal{D}_s^{\mathrm{ext}}(\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_h}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_h}\right) \cdot (\nabla_h \varphi_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_h}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_h} \, \mathrm{d}\sigma \\ & + \int_{\Gamma_h} \left(\mathcal{D}_s^{\mathrm{ext}} \mathcal{P}^{\Phi_h} \nabla_h u_{s,h}\right) \cdot (\nabla_h \varphi_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_h}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_h} \, \mathrm{d}\sigma \\ & + \int_{\Gamma_h} \left(\mathcal{D}_s^{\mathrm{ext}} (\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_h}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_h}\right) \cdot \mathcal{P}^{\Phi_h} \nabla_h \varphi_{s,h} \, \mathrm{d}\sigma,\end{aligned}$$

according to Theorem 4.2.7.

Considering the scaled version of the normal penalty stabilization term (4.33), we have

$$\frac{1}{\delta} j_s^{\mathrm{np}}(u_{s,h}, \varphi_{s,h}) = \frac{\gamma_{\mathrm{np}}}{\alpha_{\mathrm{in}} + \alpha_{\mathrm{out}}} \int_{\Omega_{\delta,h}} (\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_h})(\nabla_h \varphi_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_h}) \, \mathrm{d}x. \quad (4.34)$$

Just as its counterpart without scaling, the scaled term (4.34) penalizes the $L^2(\Omega_{\delta,h})$-norm of the gradient's normal component. However, the term (4.34) vanishes in the limit $h \to 0$. Its leading factor $\frac{\gamma_{\mathrm{np}}}{\alpha_{\mathrm{in}} + \alpha_{\mathrm{out}}}$ does not depend on h and $\Omega_{\delta,h}$ approaches Γ_h for small h, i.e., a set of measure zero in $\mathbb{R}^d$.

With the scaling factor of $\frac{1}{\delta}$, both stabilization mechanisms hence target $(\nabla_h u_{s,h} \cdot \boldsymbol{\nu}^{\boldsymbol{\Phi}_h}) \boldsymbol{\nu}^{\boldsymbol{\Phi}_h}$ in a slightly different manner. While full gradient stabilization targets the normal component effectively on the discrete reconstruction Γ_h of the hypersurface, normal penalty stabilization targets the normal component everywhere in the discrete reconstruction $\Omega_{\delta,h}$ of the associated narrow band, but effectively looses its influence in the limit $h \to 0$.

Interestingly, despite this theoretically-different behavior, numerical results in Section 4.3 will show that the two stabilization mechanisms perform equally well in terms of errors and condition numbers for small values of h.

4.2.5. Fully discrete schemes

For the class of elliptic model problems (4.5), we have a total of three fully discrete schemes. Scheme 4.2.6 is not considered in the remainder of this thesis, given that its performance will strongly depend on the specific choice of the

extensions $f_{s,b}^{\mathrm{ext}}(u_{b,h}, u_{s,h})$ and $f_s^{\mathrm{ext}}(u_{s,h})$, and for the other reasons stated in Section 4.2.3. Scheme 4.2.9 and its stabilized variant Scheme 4.2.15 will be further investigated in Section 4.3.

Using discretization in time, we will now derive similar, fully discrete schemes for the class of parabolic model problems (4.1). Let the considered observation period $[0,T]$ be divided into subintervals $\left[t^{n-1}, t^n\right]$ of length $\tau^n := t^n - t^{n-1}$, $n = 1, \ldots, N$, with $t^0 = 0$, $t^N = T$ and $t^n > t^{n-1}$ for $n = 1, \ldots, N$. Moreover, let $\left(u_{b,h}^n, u_{s,h}^n\right)$ and $\diamond^n$ denote a fully discrete solution pair and an entity $\diamond$ that may be time-dependent, respectively, each evaluated at $t = t^n$.

Backward Euler time-stepping

The backward Euler method is the simplest implicit first order in time method available. Employing the backward Euler method for time-stepping yields schemes of the following kind.

Scheme 4.2.16 (General structure using backward Euler time-stepping)**.** *Let $u_{b,h}^0 \in V_{b,h}(\Omega_\hbar)$ and $u_{s,h}^0 \in V_{s,h}(\Omega_{\delta,\hbar})$ denote approximate initial values as in Semidiscretization 4.2.5. For $n = 1, \ldots, N$, we seek a pair of discrete functions $\left(u_{b,h}^n, u_{s,h}^n\right) \in V_{b,h}(\Omega_\hbar) \times V_{s,h}(\Omega_{\delta,\hbar})$, such that*

$$m_b(u_{b,h}^n, \varphi_{b,h}) + \tau^n r_b^n(u_{b,h}^n, \varphi_{b,h}) = m_b\left(u_{b,h}^{n-1}, \varphi_{b,h}\right), \tag{4.35a}$$

$$m_s(u_{s,h}^n, \varphi_{s,h}) + \tau^n r_s^n(u_{s,h}^n, \varphi_{s,h}) = m_s\left(u_{s,h}^{n-1}, \varphi_{s,h}\right), \tag{4.35b}$$

for all $(\varphi_{b,h}, \varphi_{s,h}) \in V_{b,h}(\Omega_\hbar) \times V_{s,h}(\Omega_{\delta,\hbar})$. Here, we define forms

$$m_b(u_{b,h}, \varphi_{b,h}) := \int_{\Omega_\hbar} u_{b,h}\, \varphi_{b,h} \,\mathrm{d}x \qquad \text{and}$$

$$r_b(u_{b,h}, \varphi_{b,h}) := a_b(u_{b,h}, \varphi_{b,h}) - c_b(u_{b,h}, u_{s,h}, \varphi_{b,h}),$$

where a_b is the bilinear form that has been defined in equation (4.19a) *and c_b is the potentially nonlinear form defined in equation* (4.17)*. The forms m_s and r_s are defined by the specific variant of the scheme.*

In particular, taking Semidiscretization 4.2.8 and Semidiscretization 4.2.10 as a basis, we obtain the following fully discrete schemes.

Scheme 4.2.17 (Variant A, parabolic problems)**.** *We apply Scheme 4.2.16 with*

$$m_s(u_{s,h}, \varphi_{s,h}) := \frac{1}{\delta} \int_{\Omega_{\delta,\hbar}} |\nabla_\hbar \Phi_\hbar|\, u_{s,h}\, \varphi_{s,h} \,\mathrm{d}x \qquad \text{and}$$

$$r_s(u_{s,h}, \varphi_{s,h}) := \frac{1}{\delta}\, a_s(u_{s,h}, \varphi_{s,h}) - c_s(u_{b,h}, u_{s,h}, \varphi_{s,h}),$$

where a_s is the bilinear form that has been defined in equation (4.19b) *and c_s is the potentially nonlinear form defined in equation* (4.22)*.*

Scheme 4.2.18 (Variant B, parabolic problems)**.** *We apply Scheme 4.2.16 with*

$$m_s(u_{s,h}, \varphi_{s,h}) := \int_{\Gamma_\hbar} u_{s,h}\big|_{\Gamma_\hbar}\, \varphi_{s,h}\big|_{\Gamma_\hbar}\, \mathrm{d}\sigma$$

and $r_s(u_{s,h}, \varphi_{s,h})$ *as in Scheme 4.2.17.*

Moreover, augmenting the chosen bilinear form $a_s(u_{s,h}, \varphi_{s,h})$ by one of the stabilization terms $j_s(u_{s,h}, \varphi_{s,h})$ which have been discussed in Section 4.2.4 yields the following stabilized variant of Scheme 4.2.18.

Scheme 4.2.19 (Variant B stabilized, parabolic problems)**.** *Given one of our stabilization terms* $j_s(u_{s,h}, \varphi_{s,h})$*, we apply Scheme 4.2.16 with* $m_s(u_{s,h}, \varphi_{s,h})$ *as in Scheme 4.2.18 and*

$$r_s(u_{s,h}, \varphi_{s,h}) := \frac{1}{\delta}\big[a_s + j_s\big](u_{s,h}, \varphi_{s,h}) - c_s(u_{b,h}, u_{s,h}, \varphi_{s,h}),$$

where, again, a_s *is the bilinear form that has been defined in equation* (4.19b) *and* c_s *is the potentially nonlinear form defined in equation* (4.22)*.*

Regarding the general structure of these schemes, which is provided by equations (4.35), we note that the forms r_b and r_s may be time-dependent. This is due to the fact that both diffusivity tensors $\mathcal{D}_b$ and $\mathcal{D}_s$, and the source/sink densities $f_b(u_{b,h})$, $f_s(u_{s,h}|_{\Gamma_\hbar})$, $f_{b,s}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar})$ and $f_{s,b}(u_{b,h}, u_{s,h}|_{\Gamma_\hbar})$ may be time-dependent. This time-dependency is passed on to the forms a_b, c_b, a_s, j_s (when choosing $j_s = j_s^{\mathrm{fg}}$) and c_s.

Conservation properties

Scheme 4.2.18 and its stabilized variant Scheme 4.2.19 recover fully discrete analogues to the global conservation properties (4.3) and (4.4) which have been discussed in Section 4.1. This is the subject of the following theorem.

Theorem 4.2.20 (Fully discrete solutions – global conservation properties)**.** *At each step* $n = 1, \dots, N$*, a solution pair* $\big(u_{b,h}^n, u_{s,h}^n\big) \in V_{b,h}(\Omega_\hbar) \times V_{s,h}(\Omega_{\delta,\hbar})$ *that is obtained using Scheme 4.2.18 or Scheme 4.2.19 satisfies*

$$\begin{aligned}
\int_{\Omega_\hbar} u_{b,h}^n\, \mathrm{d}x &= \int_{\Omega_\hbar} u_{b,h}^{n-1}\, \mathrm{d}x \\
&\qquad + \tau^n \bigg(\int_{\Gamma_\hbar} f_{b,s}^n\big(u_{b,h}^n, u_{s,h}^n\big|_{\Gamma_\hbar}\big)\, \mathrm{d}\sigma + \int_{\Omega_\hbar} f_b^n\big(u_{b,h}^n\big)\, \mathrm{d}x \bigg), \\
\int_{\Gamma_\hbar} u_{s,h}^n\big|_{\Gamma_\hbar}\, \mathrm{d}\sigma &= \int_{\Gamma_\hbar} u_{s,h}^{n-1}\big|_{\Gamma_\hbar}\, \mathrm{d}\sigma + \tau^n \int_{\Gamma_\hbar} f_{s,b}^n\big(u_{b,h}^n, u_{s,h}^n\big|_{\Gamma_\hbar}\big) + f_s^n\big(u_{s,h}^n\big|_{\Gamma_\hbar}\big)\, \mathrm{d}\sigma.
\end{aligned}$$

The latter identities are fully discrete analogues to the global conservation properties (4.3)*. In addition, combining both identities yields a fully discrete*

analogue to property (4.4)*, which describes the evolution of the total amount* $m_h^n := \int_{\Omega_\hbar} u_{b,h}^n \,\mathrm{d}x + \int_{\Gamma_\hbar} u_{s,h}^n|_{\Gamma_\hbar} \,\mathrm{d}\sigma$ *of the fully discrete system's quantities:*

$$m_h^n = m_h^{n-1} + \tau^n \Bigg(\int_{\Gamma_\hbar} f_{b,s}^n\big(u_{b,h}^n, u_{s,h}^n\big|_{\Gamma_\hbar}\big) + f_{s,b}^n\big(u_{b,h}^n, u_{s,h}^n\big|_{\Gamma_\hbar}\big) \,\mathrm{d}\sigma + \int_{\Omega_\hbar} f_b^n\big(u_{b,h}^n\big) \,\mathrm{d}x + \int_{\Gamma_\hbar} f_s^n\big(u_{s,h}^n\big|_{\Gamma_\hbar}\big) \,\mathrm{d}\sigma \Bigg).$$

Proof. The theorem can be proven in a straightforward way, using the same arguments as in the proof of Theorem 4.2.11 as well as the fact that we have $j_s\big(u_{s,h}, \mathbb{1}_{\Omega_{\delta,\hbar}}\big) = 0$ for all stabilization terms considered in this thesis, as discussed in Section 4.2.4. □

Similar fully discrete analogues to local conservation properties (4.2) are also recovered by Scheme 4.2.18 and Scheme 4.2.19. The latter can be shown by applying the arguments that are used in the proof of Theorem 4.2.13.

4.3. Numerical results

To validate the practicability of the schemes which have been introduced in Section 4.2, and to investigate their convergence and conditioning properties, we implemented the schemes in C++ using the Distributed and Unified Numerics Environment (`DUNE`)[1]. With this implementation, an extensive set of numerical studies has been performed. Its main outcome will be presented in the remainder of this section.

Details on the implementation and on how the code can be obtained to reproduce the results which we present in the following can be found in Appendix A.2.

4.3.1. Linear elliptic model problems

Construction of analytical test problems

To further investigate Scheme 4.2.9 from Section 4.2.3 and Scheme 4.2.15 from Section 4.2.4, we construct specific models from the class of elliptic model problems (4.5) which has been introduced in Section 4.1.2. These specific models are linear elliptic models with a known analytical solution, making it possible to perform numerical convergence studies for our schemes and to numerically investigate condition numbers that are associated with solving the systems of linear equations which result from applying the schemes. For the sake of simplicity with respect to the availability of analytical solutions, we use a circular two-dimensional geometry centered at the origin and employ constant, scalar diffusivities. In this special case, closed formulas are available for the surface differential operator, that are based on polar coordinates.

[1] `https://www.dune-project.org`

In particular, by choosing $d := 2$, $\Omega := \left\{\underline{\mathbf{x}} \in \mathbb{R}^2 \;\middle|\; |\underline{\mathbf{x}}| < 1\right\}$, $\mathcal{D}_b := \mathcal{I}$ and $\mathcal{D}_s := \mathcal{I}$ in class (4.5), we obtain the restricted problem of finding $u_b \colon \Omega \to \mathbb{R}$ and $u_s \colon \Gamma \to \mathbb{R}$ with

$$-\Delta u_b = f_b(u_b) \qquad \text{in} \quad \Omega, \tag{4.36a}$$
$$-\nabla u_b \cdot \boldsymbol{\nu} = -f_{b,s}(u_b, u_s) \qquad \text{on} \quad \Gamma, \tag{4.36b}$$
$$-\Delta_\Gamma u_s = f_{s,b}(u_b, u_s) + f_s(u_s) \qquad \text{on} \quad \Gamma. \tag{4.36c}$$

Here, we use $f_{b,s}(u_b, u_s) := -f_{s,b}(u_b, u_s)$, where $f_{s,b}(u_b, u_s) := u_b|_\Gamma - u_s$. Furthermore, we employ a term $f_b(u_b)$ that is linear in u_b, and a term $f_s(u_s)$ of the specific form $f_s(u_s) := \tilde{f}_s - u_s$ with a given data function $\tilde{f}_s \colon \Gamma \to \mathbb{R}$. The latter will render the problem uniquely solvable, given our choices of $f_{b,s}(u_b, u_s)$ and $f_{s,b}(u_b, u_s)$. Note that we either need to choose a term $f_s(u_s)$ which effectively depends on u_s, like the term given above, or $f_b(u_b)$ is required to be a term dependent on u_b. If both terms were chosen independent of u_b and u_s, respectively, classical solutions to problem (4.36) would not be unique, if existent. In fact, given a classical solution (u_b, u_s), the pair $(u_b + c, u_s + c)$ would also be a classical solution for every constant $c \in \mathbb{R}$.

To construct specific models with known analytical solutions, we follow the idea of the method of manufactured solutions (see e.g. Salari and Knupp, 2000). Our particular approach for system (4.36) is the following:

1. Choose an appropriate u_b, e.g. a harmonic function or an eigenfunction of the Laplacian.
2. Determine $f_b(u_b)$ by means of bulk equation (4.36a), yielding $f_b(u_b) \equiv 0$ if u_b is chosen to be a harmonic function, and $f_b(u_b) = -\lambda\, u_b$ if it is chosen to be an eigenfunction of the Laplacian with some eigenvalue λ.
3. Calculate u_s from boundary condition (4.36b).
4. Identify $\tilde{f}_s$ via surface equation (4.36c).

Some linear elliptic test problems which we constructed by applying this procedure are specified in Table 4.2. They have been derived with the help of the following remark.

Remark 4.3.1 (Representation of differential operators in coordinate systems for circular geometries)**.**

1. *Using a parametrization $\underline{\mathbf{x}} = r\underline{\boldsymbol{\theta}} \in \mathbb{R}^d$, with $r \in \mathbb{R}^{\geq 0}$ being the radial distance of $\underline{\mathbf{x}}$ to the origin $(0, \ldots, 0)^{\mathrm{tr}} \in \mathbb{R}^d$ and $\underline{\boldsymbol{\theta}}$ being an element of the $(d-1)$-dimensional unit sphere S^{d-1}, the action of the Laplacian on a twice differentiable scalar field $u \colon \mathbb{R}^d \to \mathbb{R}$ can be represented as*

$$\Delta u = \frac{\partial^2 u}{\partial r^2} + \frac{d-1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\Delta_{S^{d-1}} u.$$

Furthermore, using polar coordinates (r, ϕ) for $d = 2$, the Laplace–Beltrami

	Elliptic 2d test problem "1"	**Elliptic 2d test problem "2"**	**Elliptic 2d test problem "3"**
$u_b(\underline{\mathbf{x}})$	x_1	$\mathrm{x}_0\mathrm{x}_1$	$-\mathrm{x}_0^5 + 10\mathrm{x}_0^3\mathrm{x}_1^2 - 5\mathrm{x}_0\mathrm{x}_1^4 + 3\mathrm{x}_0^2\mathrm{x}_1 - \mathrm{x}_1^3$
$f_b(u_b)(\underline{\mathbf{x}})$	0	0	0
$u_s(\underline{\mathbf{x}})$	$2\mathrm{x}_1$	$3\mathrm{x}_0\mathrm{x}_1$	$-96\mathrm{x}_0^5 + 120\mathrm{x}_0^3 + 16\mathrm{x}_0^2\mathrm{x}_1 - 30\mathrm{x}_0 - 4\mathrm{x}_1$
$\tilde{f}_s(\underline{\mathbf{x}})$	$5\mathrm{x}_1$	$17\mathrm{x}_0\mathrm{x}_1$	$-2576\mathrm{x}_0^5 + 3320\mathrm{x}_0^3 + 172\mathrm{x}_0^2\mathrm{x}_1 - 805\mathrm{x}_0 - 43\mathrm{x}_1$

	Elliptic 2d test problem "6"
$u_b(\underline{\mathbf{x}})$	$\sin(\alpha\mathrm{x}_0 + \beta\mathrm{x}_1)$ with $\alpha, \beta \in \mathbb{R}$ (e.g. $\alpha = 8\pi, \beta = 2\pi$)
$f_b(u_b)(\underline{\mathbf{x}})$	$(\alpha^2 + \beta^2)\, u_b(\underline{\mathbf{x}})$
$u_s(\underline{\mathbf{x}})$	$(\alpha\mathrm{x}_0 + \beta\mathrm{x}_1)\cos(\alpha\mathrm{x}_0 + \beta\mathrm{x}_1) + \sin(\alpha\mathrm{x}_0 + \beta\mathrm{x}_1)$
$\tilde{f}_s(\underline{\mathbf{x}})$	$\big[(-\alpha^3 + 3\alpha\beta^2)\mathrm{x}_0^3 + (-3\alpha^2\beta + \beta^3)\mathrm{x}_0^2\mathrm{x}_1$ $+(\alpha^3 - 2\alpha\beta^2 + 4\alpha)\mathrm{x}_0 + (\alpha^2\beta + 4\beta)\mathrm{x}_1\big]\cos(\alpha\mathrm{x}_0 + \beta\mathrm{x}_1)$ $+\big[(-4\alpha^2 + 4\beta^2)\mathrm{x}_0^2 - 8\alpha\beta\mathrm{x}_0\mathrm{x}_1 + 3\alpha^2 - \beta^2 + 1\big]\sin(\alpha\mathrm{x}_0 + \beta\mathrm{x}_1)$

Table 4.2.: Linear elliptic test problems: data functions $f_b(u_b)$ and $\tilde{f}_s$, and the associated solution (u_b, u_s) of system (4.36) in two-dimensional Cartesian coordinates $\underline{\mathbf{x}} = (\mathrm{x}_0, \mathrm{x}_1)$.

operator on S^1 applied to the field u can be represented as

$$\Delta_{S^1} u = \frac{\partial^2 u}{\partial \phi^2},$$

where ϕ represents the azimuthal angle. Similarly, using spherical coordinates (r, θ, ϕ) for $d = 3$, the Laplace–Beltrami operator on S^2 applied to the field u takes the form

$$\Delta_{S^2} u = \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial u}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2 u}{\partial\phi^2},$$

where θ represents the zenith angle (also known as inclination angle) and ϕ represents the azimuthal angle.

2. *Using polar coordinates (r, ϕ) for $d = 2$, spherical coordinates (r, θ, ϕ) for $d = 3$ or, in general, a parametrization $(r, \underline{\boldsymbol{\theta}})$ as described above, the normal derivative on S^{d-1} of a differentiable scalar field $u\colon \mathbb{R}^d \to \mathbb{R}$ can be represented as*

$$\nabla u \cdot \boldsymbol{\nu}_{\mathbf{S^{d-1}}} = \frac{\partial u}{\partial r}.$$

3. *Conversions between two-dimensional Cartesian coordinates* $\underline{\mathrm{x}} = (\mathrm{x}_0, \mathrm{x}_1)$ *and polar coordinates* (r, ϕ) *are given by*

$$\begin{aligned} \mathrm{x}_0 &= r\,\cos\phi, & r &= |\underline{\mathbf{x}}|, \\ \mathrm{x}_1 &= r\,\sin\phi, & \phi &= \operatorname{arctan2}(\mathrm{x}_1, \mathrm{x}_0). \end{aligned}$$

The function $\operatorname{arctan2}(\mathrm{x}_1, \mathrm{x}_0)$ *for the azimuthal angle will be defined below. Similarly, conversions between three-dimensional Cartesian coordinates* $\underline{\mathbf{x}} = (\mathrm{x}_0, \mathrm{x}_1, \mathrm{x}_2)$ *and spherical coordinates* (r, θ, ϕ) *are given by*

$$\begin{aligned} \mathrm{x}_0 &= r\,\sin\theta\,\cos\phi, \\ \mathrm{x}_1 &= r\,\sin\theta\,\sin\phi, \\ \mathrm{x}_2 &= r\,\cos\theta, \\ r &= |\underline{\mathbf{x}}|, \\ \theta &= \arccos\frac{\mathrm{x}_2}{r}, \\ \phi &= \operatorname{arctan2}(\mathrm{x}_1, \mathrm{x}_0), \end{aligned}$$

$$\operatorname{arctan2}(\mathrm{x}_1, \mathrm{x}_0) := \begin{cases} \arctan\left(\frac{\mathrm{x}_1}{\mathrm{x}_0}\right) & \mathrm{x}_0 > 0, \\ \arctan\left(\frac{\mathrm{x}_1}{\mathrm{x}_0}\right) + \pi & \mathrm{x}_0 < 0,\ \mathrm{x}_1 \geq 0, \\ \arctan\left(\frac{\mathrm{x}_1}{\mathrm{x}_0}\right) - \pi & \mathrm{x}_0 < 0,\ \mathrm{x}_1 < 0, \\ +\frac{\pi}{2} & \mathrm{x}_0 = 0,\ \mathrm{x}_1 > 0, \\ -\frac{\pi}{2} & \mathrm{x}_0 = 0,\ \mathrm{x}_1 < 0, \\ \textit{undefined} & \mathrm{x}_0 = 0,\ \mathrm{x}_1 = 0. \end{cases}$$

In the following, we particularly look at results for elliptic 2d test problem "2" and results for elliptic 2d test problem "3". Graphs of the corresponding solution pairs (u_b, u_s) are depicted in Figure 4.6. The remaining test problems that are specified in Table 4.2 will not serve as test problems within the scope of this thesis. They yield comparable results, though.

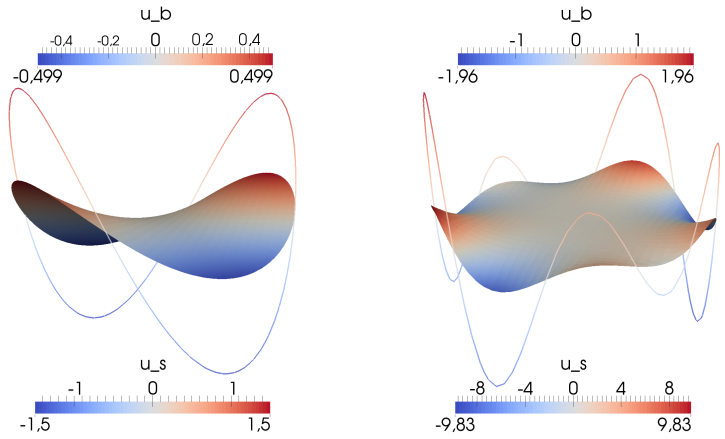
The corresponding systems of linear equations

Applying Scheme 4.2.9 or Scheme 4.2.15 to the class of problems that are described by system (4.36), e.g. to one of our linear elliptic test problems, algebraically yields a system of linear equations

$$A\underline{\mathbf{u}} = \underline{\mathbf{b}}. \tag{4.37}$$

Here, $\underline{\mathbf{u}}$ denotes the vector of degrees of freedom (DOFs) that are associated with the unknown discrete solution pair $(u_{b,h}, u_{s,h}) \in V_{b,h}(\Omega_{\mathit{fl}}) \times V_{s,h}(\Omega_{\delta,\mathit{fl}})$, and each linear equation, i.e. each line of system (4.37), corresponds to testing with one basis function of the discrete space $V_{b,h}(\Omega_{\mathit{fl}}) \times V_{s,h}(\Omega_{\delta,\mathit{fl}})$.

In our implementation, we order the entries in $\underline{\mathbf{u}}$ such that its first entries $\mathrm{u}_1, \ldots, \mathrm{u}_{\dim_b}$ with $\dim_b := \dim\big(V_{b,h}(\Omega_{\mathit{fl}})\big)$ are DOFs associated with $u_{b,h}$ and its remaining entries $\mathrm{u}_{\dim_b+1}, \ldots, \mathrm{u}_{\dim_b+\dim_s}$ with $\dim_s := \dim\big(V_{s,h}(\Omega_{\delta,\mathit{fl}})\big)$ are DOFs associated with $u_{s,h}$. Furthermore, we order the test functions in the same manner, such that the i-th line of system (4.37) corresponds to testing with the basis function which is associated with the i-th DOF in $\underline{\mathbf{u}}$. In this case, the matrix in system (4.37) and the vector on the right-hand side have



(a) Elliptic 2d test problem "2". (b) Elliptic 2d test problem "3".

Figure 4.6.: Graph of the solution pairs (u_b, u_s) of elliptic 2d test problem "2" and elliptic 2d test problem "3", cf. Table 4.2.

a block structure of the form

$$A = \begin{pmatrix} A_b & A_{bs} \\ A_{sb} & A_s \end{pmatrix}, \quad \underline{\mathbf{b}} = \begin{pmatrix} \underline{\mathbf{b}}_b \\ \underline{\mathbf{b}}_s \end{pmatrix}.$$

In view of equation (4.18a), the upper left block $A_b \in \mathbb{R}^{\dim_b \times \dim_b}$ of the matrix comprises contributions of all terms in $a_b(u_{b,h}, \varphi_{b,h})$, and contributions of terms in $c_b(u_{b,h}, u_{s,h}, \varphi_{b,h})$ that effectively depend on $u_{b,h}$. Choosing SIPG or SWIPG as host DG formulation for the bulk part of the problem results in a symmetric block A_b. Analogously, in view of equations (4.23) and (4.30), the lower right block $A_s \in \mathbb{R}^{\dim_s \times \dim_s}$ comprises contributions of all terms in $\frac{1}{\delta}\, a_s(u_{s,h}, \varphi_{s,h})$, and contributions of terms in $c_s(u_{b,h}, u_{s,h}, \varphi_{s,h})$ that effectively depend on $u_{s,h}$. In case of the stabilized scheme, furthermore all terms in $\frac{1}{\delta}\, j_s(u_{s,h}, \varphi_{s,h})$ contribute to A_s. Together with or without one of the optional stabilization terms that have been discussed in Section 4.2.4, choosing SIPG or SWIPG as host DG formulation for the surface part of the problem results in a symmetric block A_s, as long as the diffusivity $\mathcal{D}_s = \mathcal{I}$ is extended in such a way that its extension $\mathcal{D}_s^{\text{ext}}$ is a symmetric tensor. The latter trivially holds true for constant extensions.

The upper right block $A_{bs} \in \mathbb{R}^{\dim_b \times \dim_s}$ comprises contributions of those terms in $c_b(u_{b,h}, u_{s,h}, \varphi_{b,h})$ that effectively depend on $u_{s,h}$. The same holds true for the lower left block $A_{sb} \in \mathbb{R}^{\dim_s \times \dim_b}$ and terms in $c_s(u_{b,h}, u_{s,h}, \varphi_{s,h})$ that effectively depend on $u_{b,h}$. In case of the data functions $f_{b,s}(u_b, u_s)$ and

$f_{s,b}(u_b, u_s)$ which we choose for test problems of the form (4.36), we have

$$c_b(u_{b,h}, u_{s,h}, \varphi_{b,h}) = \int_{\Gamma_\hbar} -\big(u_{b,h} - u_{s,h}|_{\Gamma_\hbar}\big)\, \varphi_{b,h}\ \mathrm{d}\sigma + \int_{\Omega_\hbar} f_b(u_{b,h})\, \varphi_{b,h}\ \mathrm{d}x,$$
$$c_s(u_{b,h}, u_{s,h}, \varphi_{s,h}) = \int_{\Gamma_\hbar} \big(u_{b,h} - u_{s,h}|_{\Gamma_\hbar}\big)\, \varphi_{s,h}|_{\Gamma_\hbar} + f_s(u_{s,h}|_{\Gamma_\hbar})\, \varphi_{s,h}|_{\Gamma_\hbar}\ \mathrm{d}\sigma.$$

Thus, the contributions to blocks A_{bs} and A_{sb} are exactly those of the terms

$$\int_{\Gamma_\hbar} u_{s,h}|_{\Gamma_\hbar}\, \varphi_{b,h}\ \mathrm{d}\sigma \qquad \text{and} \qquad \int_{\Gamma_\hbar} u_{b,h}\, \varphi_{s,h}|_{\Gamma_\hbar}\ \mathrm{d}\sigma,$$

respectively. It can be seen easily that the latter two terms result in blocks with

$$A_{bs} = A_{sb}^{\mathrm{tr}}.$$

Under the conditions which have been discussed above for symmetry of the blocks A_b and A_s, the matrix A hence is a symmetric matrix.

As briefly explained in Section 1.5, every numerical scheme should be designed in such a way that it results in useful systems of algebraic equations. In particular, that kind of errors which are unavoidable while solving the system still need to stay within manageable limits. These limits assure that numerical solutions are practically usable. In Appendix B, we describe this aspect in a more detailed way. We show that an appropriate indicator for systems of linear equations is given by a real number that is known as the *spectral condition number* of the system matrix. As shown in Appendix B.3, it can be expressed in terms of eigenvalues of matrices and can hence be accessed relatively easy.

In this light, when applying our schemes to the linear elliptic test problems that we are considering, we will investigate the spectral condition number $\kappa_2(A)$ which is associated with solving the corresponding system (4.37). To perform our studies, we compute $\kappa_2(A)$ numerically, exploiting symmetry of the system matrix A if possible. For details on how we do this, please refer to Appendix B.4 and Appendix B.5.

Common simulation parameters

In our numerical studies for linear elliptic model problems, we use the following simulation parameters and related choices, unless otherwise stated.

The circular geometry which is associated with system (4.36) is described choosing the level set domain $\Omega_\Phi := (-2, 2)^2$ and the level set function Φ that is defined by $\Phi(\underline{\mathbf{x}}) := |\underline{\mathbf{x}}| - 1.0$. To perform the extension process which is described in Section 4.2.1, we use a narrow band Ω_δ with equal parameters $\alpha_{\mathrm{in}} = \alpha_{\mathrm{out}} =: \alpha \in \mathbb{R}^{>0}$, and choose the corresponding constant extension to Ω_δ as extension $\mathcal{D}_s^{\mathrm{ext}}$ of the constant diffusivity $\mathcal{D}_s = \mathcal{I}$.

For the UDG discretization, we use Cartesian fundamental meshes $\mathcal{T}_h(\Omega_\Phi)$

and Cartesian geometry meshes $\mathcal{T}_{\hbar}(\Omega_\Phi)$ with varying mesh widths h and $\hbar$, respectively. To obtain $\mathcal{T}_h(\Omega_\Phi)$, we start with one entity which corresponds to Ω_Φ and perform a certain number $\mathfrak{r} \in \mathbb{N}$ of uniform mesh refinements. We either use the choice $\mathcal{T}_{\hbar}(\Omega_\Phi) := \mathcal{T}_h(\Omega_\Phi)$, i.e. no separate geometry mesh, or a mesh $\mathcal{T}_{\hbar}(\Omega_\Phi)$ which results from a fixed number of additional uniform refinements for each of the entities in $\mathcal{T}_h(\Omega_\Phi)$.

On each fundamental mesh, we choose discrete spaces $V_{s,h}(\Omega_{\delta,\hbar})$ and $V_{b,h}(\Omega_\hbar)$ which locally (i.e. on each cut cell K) resemble $\mathbb{P}(K) := P_k(K)$, the space of polynomial functions of total degree less than or equal to some $k \in \mathbb{N}$ over the domain K. For their construction, we use monomial basis functions (i.e. the set $\{1, \mathrm{x}_0, \mathrm{x}_1\}$ for $k = 1$ and the set $\{1, \mathrm{x}_0, \mathrm{x}_1, \mathrm{x}_0\mathrm{x}_1, \mathrm{x}_0^2, \mathrm{x}_1^2\}$ for $k = 2$) on the reference element of the fundamental mesh elements. As polynomial degrees, we employ $k = 1$ or $k = 2$, choosing the degree equally for both discrete spaces.

As host DG formulations, we use SIPG for the bulk part of the problem and either SIPG or SWIPG for its surface part. In particular, we choose $\mathrm{host_b} :=$ sipg with $\gamma_b := 5$ for elliptic 2d test problem "2", and with $\gamma_b := 10$ for elliptic 2d test problem "3". Moreover, we choose $\mathrm{host_s} :=$ sipg with $\gamma_s := 7.0$ or $\mathrm{host_s} :=$ swipg with $\gamma_s := 22.5$. If normal penalty stabilization is performed, we employ the penalty parameter $\gamma_{\mathrm{np}} := 40.0$.

Error measures

In order to analyze the convergence of our schemes, we compute *relative errors*

$$\mathcal{E}_{b,L^2(\Omega_\hbar)}(h) := \|u_b - u_{b,h}\|_{L^2(\Omega_\hbar)} \;/\; \|u_b\|_{L^2(\Omega_\hbar)}\,, \tag{4.38a}$$
$$\mathcal{E}_{s,L^2(\Gamma_\hbar)}(h) := \|u_s - u_{s,h}\|_{L^2(\Gamma_\hbar)} \;/\; \|u_s\|_{L^2(\Gamma_\hbar)}\,, \tag{4.38b}$$
$$\mathcal{E}_{b,H^1(\Omega_\hbar)}(h) := \|u_b - u_{b,h}\|_{H^1(\Omega_\hbar)} \;/\; \|u_b\|_{H^1(\Omega_\hbar)}\,, \tag{4.38c}$$
$$\mathcal{E}_{s,H^1(\Gamma_\hbar)}(h) := \|u_s - u_{s,h}\|_{H^1(\Gamma_\hbar)} \;/\; \|u_s\|_{H^1(\Gamma_\hbar)}\,. \tag{4.38d}$$

Here, the pair $(u_{b,h}, u_{s,h}) \in V_{b,h}(\Omega_\hbar) \times V_{s,h}(\Omega_{\delta,\hbar})$ again denotes the numerical solution, and

$$v_b \in H^1(\Omega_\hbar): \quad \|v_b\|_{H^1(\Omega_\hbar)} := \left(\|v_b\|^2_{L^2(\Omega_\hbar)} + \left\|\nabla v_b\right\|^2_{L^2(\Omega_\hbar)}\right)^{\frac{1}{2}},$$
$$v_s \in H^1(\Gamma_\hbar): \quad \|v_s\|_{H^1(\Gamma_\hbar)} := \left(\|v_s\|^2_{L^2(\Gamma_\hbar)} + \left\|\nabla_{\Gamma_\hbar} v_s\right\|^2_{L^2(\Gamma_\hbar)}\right)^{\frac{1}{2}}.$$

Note that ∇v_b and $\nabla_{\Gamma_\hbar} v_s$ in this definition shall be understood in a weak sense, as usual when considering derivatives in Sobolev spaces. Since both components of the numerical solution are only piecewise H^1-functions, we actually compute $H^1(\cdot)$-norms in a piecewise manner, i.e., we replace gradients in the above definition by piecewise variants, cf. equation (4.15).

On this basis, we compute *experimental orders of convergence*, assuming that the relative errors defined in equations (4.38) satisfy estimates of the form $\mathcal{E}(h) \in \mathcal{O}(h^z)$, where $z \in \mathbb{R}^{>0}$ is known as the order of convergence

with respect to $\mathcal{E}(h)$. For an error $\mathcal{E}(h)$ and for the grid sizes h_1 and h_2, the experimental order of convergence is defined as

$$\operatorname{eoc}\big[\mathcal{E}\big](h_1, h_2) := \log \frac{\mathcal{E}(h_1)}{\mathcal{E}(h_2)} \left(\log \frac{h_1}{h_2} \right)^{-1},$$

where log denotes the logarithm operator to an arbitrarily chosen base.

Numerical study: h-refinement and polynomial degree $k = 1$

As a first numerical study, we investigate relative errors, associated experimental orders of convergence and the spectral condition number of the system matrix with respect to the width h of the fundamental mesh, starting with polynomial degree $k = 1$. To do so, we employ a strategy known as h-refinement. We perform computations while considering a sequence of fundamental meshes with $h \to 0$, which results from refinement of some coarse initial mesh. At the same time, we keep the parameter α of the narrow band fixed.

First, we consider Scheme 4.2.9, choosing SIPG as host DG formulation for the surface part of the problem and not using separate geometry meshes. Results for elliptic 2d test problem "2" are shown in Table 4.3 and Figure 4.7, results for elliptic 2d test problem "3" in Table 4.4 and Figure 4.8. Note that the discrete narrow band $\Omega_{\delta,h}$ exceeds the level set domain Ω_Φ for large values of α and h. Data points are missing in those cases. It can be observed that the unstabilized scheme is capable of achieving optimal order convergence with respect to the two H^1-norms and the two L^2-norms that define the relative errors which we are considering. Provided that the narrow band parameter α is chosen small enough, both components of the numerical solution $(u_{b,h}, u_{s,h})$ converge to their corresponding component of the solution pair (u_b, u_s) with order 1 in $\|\cdot\|_{H^1(\Omega_h)}$ and $\|\cdot\|_{H^1(\Gamma_h)}$, respectively, and with order 2 in $\|\cdot\|_{L^2(\Omega_h)}$ and $\|\cdot\|_{L^2(\Gamma_h)}$. As usual with finite element schemes, the spectral condition number $\kappa_2(A)$ which is associated with solving the corresponding linear system (4.37) grows with decreasing mesh width h. Moreover, it depends on the choice of the narrow band parameter α. The growth rate of $\kappa_2(A)$ with respect to h increases with decreasing α. Asymptotically, we have $\kappa_2(A) \in \mathcal{O}(h^{-3})$ in the parameter regime $1.25 \geq \alpha \geq 0.01$ which we are considering in Figure 4.7 and Figure 4.8.

Similar observations can be made for Scheme 4.2.15, i.e. for the stabilized scheme, either using full gradient stabilization or normal penalty stabilization. For being able to compare its results with those of the unstabilized scheme, we again choose $\text{host}_\text{s} := \text{sipg}$ and do not use separate geometry meshes. As shown for elliptic 2d test problem "2" in Table 4.5 and Figure 4.9, and for elliptic 2d test problem "3" in Table 4.6 and Figure 4.10, the two stabilization mechanisms perform equally well. Both yield similar errors, especially for small values of h, and the spectral condition numbers which are associated with solving their corresponding linear system (4.37) are nearly identical. We obtain

$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	–	–	–	–	–	–	–	–
2	$1.41 \cdot 10^{0}$	–	–	–	–	–	–	–	–
3	$7.07 \cdot 10^{-1}$	$5.68 \cdot 10^{-1}$	–	$3.95 \cdot 10^{-1}$	–	$3.45 \cdot 10^{-1}$	–	$4.30 \cdot 10^{-1}$	–
4	$3.54 \cdot 10^{-1}$	$2.52 \cdot 10^{-1}$	1.17	$2.60 \cdot 10^{-1}$	0.6	$2.40 \cdot 10^{-1}$	0.52	$2.78 \cdot 10^{-1}$	0.63
5	$1.77 \cdot 10^{-1}$	$1.10 \cdot 10^{-1}$	1.2	$1.12 \cdot 10^{-1}$	1.21	$1.19 \cdot 10^{-1}$	1.01	$1.65 \cdot 10^{-1}$	0.76
6	$8.84 \cdot 10^{-2}$	$3.61 \cdot 10^{-2}$	1.61	$3.79 \cdot 10^{-2}$	1.57	$4.84 \cdot 10^{-2}$	1.3	$6.82 \cdot 10^{-2}$	1.27
7	$4.42 \cdot 10^{-2}$	$1.02 \cdot 10^{-2}$	1.82	$1.18 \cdot 10^{-2}$	1.68	$2.02 \cdot 10^{-2}$	1.26	$3.88 \cdot 10^{-2}$	0.81
8	$2.21 \cdot 10^{-2}$	$2.81 \cdot 10^{-3}$	1.86	$3.75 \cdot 10^{-3}$	1.66	$9.24 \cdot 10^{-3}$	1.13	$1.86 \cdot 10^{-2}$	1.06

(a) $\alpha := 1.25$.

$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	–	–	–	–	–	–	–	–
2	$1.41 \cdot 10^{0}$	$6.64 \cdot 10^{0}$	–	$1.60 \cdot 10^{0}$	–	$1.73 \cdot 10^{0}$	–	$2.00 \cdot 10^{0}$	–
3	$7.07 \cdot 10^{-1}$	$5.40 \cdot 10^{-1}$	3.62	$2.16 \cdot 10^{-1}$	2.88	$3.50 \cdot 10^{-1}$	2.31	$4.82 \cdot 10^{-1}$	2.05
4	$3.54 \cdot 10^{-1}$	$1.28 \cdot 10^{-1}$	2.08	$5.14 \cdot 10^{-2}$	2.07	$1.53 \cdot 10^{-1}$	1.2	$1.72 \cdot 10^{-1}$	1.48
5	$1.77 \cdot 10^{-1}$	$3.55 \cdot 10^{-2}$	1.85	$1.62 \cdot 10^{-2}$	1.67	$7.32 \cdot 10^{-2}$	1.06	$1.21 \cdot 10^{-1}$	0.51
6	$8.84 \cdot 10^{-2}$	$8.05 \cdot 10^{-3}$	2.14	$3.76 \cdot 10^{-3}$	2.1	$3.56 \cdot 10^{-2}$	1.04	$4.59 \cdot 10^{-2}$	1.4
7	$4.42 \cdot 10^{-2}$	$2.18 \cdot 10^{-3}$	1.88	$1.13 \cdot 10^{-3}$	1.73	$1.76 \cdot 10^{-2}$	1.02	$3.03 \cdot 10^{-2}$	0.6
8	$2.21 \cdot 10^{-2}$	$5.24 \cdot 10^{-4}$	2.06	$3.13 \cdot 10^{-4}$	1.86	$8.73 \cdot 10^{-3}$	1.01	$1.39 \cdot 10^{-2}$	1.13

(b) $\alpha := 0.50$.

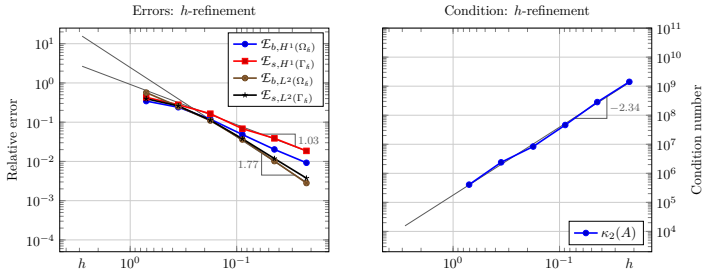
$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	$6.64 \cdot 10^{0}$	–	$1.56 \cdot 10^{0}$	–	$1.74 \cdot 10^{0}$	–	$1.37 \cdot 10^{0}$	–
2	$1.41 \cdot 10^{0}$	$6.84 \cdot 10^{0}$	-0.04	$1.26 \cdot 10^{0}$	0.31	$1.81 \cdot 10^{0}$	-0.06	$9.48 \cdot 10^{-1}$	0.53
3	$7.07 \cdot 10^{-1}$	$5.50 \cdot 10^{-1}$	3.63	$1.56 \cdot 10^{-1}$	3.01	$3.64 \cdot 10^{-1}$	2.31	$3.70 \cdot 10^{-1}$	1.36
4	$3.54 \cdot 10^{-1}$	$1.30 \cdot 10^{-1}$	2.09	$4.15 \cdot 10^{-2}$	1.91	$1.56 \cdot 10^{-1}$	1.22	$1.84 \cdot 10^{-1}$	1.01
5	$1.77 \cdot 10^{-1}$	$3.57 \cdot 10^{-2}$	1.86	$9.79 \cdot 10^{-3}$	2.08	$7.43 \cdot 10^{-2}$	1.07	$8.65 \cdot 10^{-2}$	1.09
6	$8.84 \cdot 10^{-2}$	$8.01 \cdot 10^{-3}$	2.16	$2.44 \cdot 10^{-3}$	2	$3.57 \cdot 10^{-2}$	1.06	$4.35 \cdot 10^{-2}$	0.99
7	$4.42 \cdot 10^{-2}$	$2.16 \cdot 10^{-3}$	1.89	$5.99 \cdot 10^{-4}$	2.03	$1.76 \cdot 10^{-2}$	1.02	$2.17 \cdot 10^{-2}$	1
8	$2.21 \cdot 10^{-2}$	$5.14 \cdot 10^{-4}$	2.07	$1.49 \cdot 10^{-4}$	2.01	$8.73 \cdot 10^{-3}$	1.01	$1.09 \cdot 10^{-2}$	0.99

(c) $\alpha := 0.05$.

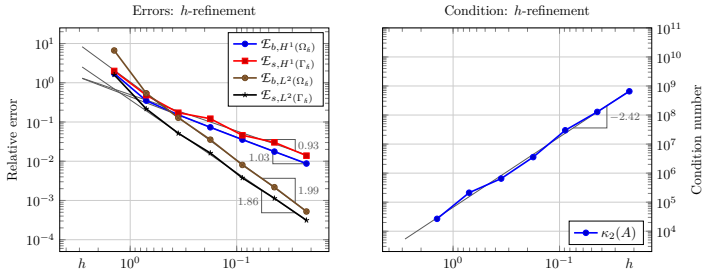
$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	$6.61 \cdot 10^{0}$	–	$1.67 \cdot 10^{0}$	–	$1.73 \cdot 10^{0}$	–	$1.10 \cdot 10^{0}$	–
2	$1.41 \cdot 10^{0}$	$6.84 \cdot 10^{0}$	-0.05	$1.26 \cdot 10^{0}$	0.41	$1.81 \cdot 10^{0}$	-0.06	$8.47 \cdot 10^{-1}$	0.37
3	$7.07 \cdot 10^{-1}$	$5.45 \cdot 10^{-1}$	3.65	$1.82 \cdot 10^{-1}$	2.79	$3.57 \cdot 10^{-1}$	2.34	$3.26 \cdot 10^{-1}$	1.38
4	$3.54 \cdot 10^{-1}$	$1.28 \cdot 10^{-1}$	2.09	$5.52 \cdot 10^{-2}$	1.72	$1.55 \cdot 10^{-1}$	1.21	$1.82 \cdot 10^{-1}$	0.84
5	$1.77 \cdot 10^{-1}$	$3.55 \cdot 10^{-2}$	1.86	$1.29 \cdot 10^{-2}$	2.1	$7.40 \cdot 10^{-2}$	1.06	$8.29 \cdot 10^{-2}$	1.14
6	$8.84 \cdot 10^{-2}$	$7.99 \cdot 10^{-3}$	2.15	$3.34 \cdot 10^{-3}$	1.95	$3.57 \cdot 10^{-2}$	1.05	$4.33 \cdot 10^{-2}$	0.94
7	$4.42 \cdot 10^{-2}$	$2.15 \cdot 10^{-3}$	1.89	$7.81 \cdot 10^{-4}$	2.1	$1.76 \cdot 10^{-2}$	1.02	$2.05 \cdot 10^{-2}$	1.08
8	$2.21 \cdot 10^{-2}$	$5.13 \cdot 10^{-4}$	2.07	$1.95 \cdot 10^{-4}$	2	$8.73 \cdot 10^{-3}$	1.01	$1.03 \cdot 10^{-2}$	0.99

(d) $\alpha := 0.01$.

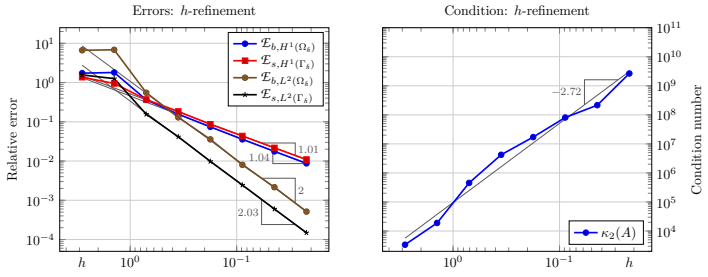
Table 4.3.: Errors in numerical solutions of elliptic 2d test problem "2", obtained using Scheme 4.2.9, $k = 1$, $\text{host}_\text{s} := \text{sipg}$, and no separate geometry meshes.



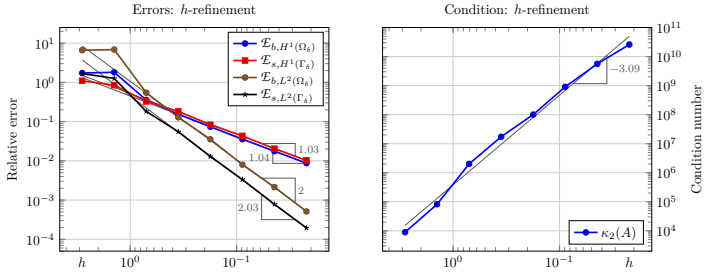
(a) $\alpha := 1.25$.



(b) $\alpha := 0.50$.



(c) $\alpha := 0.05$.



(d) $\alpha := 0.01$.

Figure 4.7.: Errors in numerical solutions of elliptic 2d test problem "2" and spectral condition number associated with the corresponding system of linear equations, obtained using Scheme 4.2.9, $k = 1$, $\text{host}_s := \text{sipg}$, and no separate geometry meshes.

$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	–	–	–	–	–	–	–	–
2	$1.41 \cdot 10^{0}$	–	–	–	–	–	–	–	–
3	$7.07 \cdot 10^{-1}$	$8.45 \cdot 10^{0}$	–	$2.27 \cdot 10^{0}$	–	$3.29 \cdot 10^{0}$	–	$1.14 \cdot 10^{0}$	–
4	$3.54 \cdot 10^{-1}$	$2.02 \cdot 10^{0}$	2.06	$1.04 \cdot 10^{0}$	1.13	$9.66 \cdot 10^{-1}$	1.77	$8.45 \cdot 10^{-1}$	0.44
5	$1.77 \cdot 10^{-1}$	$6.94 \cdot 10^{-1}$	1.54	$8.15 \cdot 10^{-1}$	0.35	$6.68 \cdot 10^{-1}$	0.53	$8.16 \cdot 10^{-1}$	0.05
6	$8.84 \cdot 10^{-2}$	$5.49 \cdot 10^{-1}$	0.34	$5.11 \cdot 10^{-1}$	0.67	$4.65 \cdot 10^{-1}$	0.52	$5.22 \cdot 10^{-1}$	0.65
7	$4.42 \cdot 10^{-2}$	$3.03 \cdot 10^{-1}$	0.86	$2.65 \cdot 10^{-1}$	0.95	$2.47 \cdot 10^{-1}$	0.91	$3.16 \cdot 10^{-1}$	0.72
8	$2.21 \cdot 10^{-2}$	$1.38 \cdot 10^{-1}$	1.14	$1.27 \cdot 10^{-1}$	1.06	$1.16 \cdot 10^{-1}$	1.09	$1.88 \cdot 10^{-1}$	0.75

(a) $\alpha := 1.25$.

$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	–	–	–	–	–	–	–	–
2	$1.41 \cdot 10^{0}$	$1.35 \cdot 10^{1}$	–	$2.59 \cdot 10^{0}$	–	$4.98 \cdot 10^{0}$	–	$1.54 \cdot 10^{0}$	–
3	$7.07 \cdot 10^{-1}$	$7.30 \cdot 10^{0}$	0.89	$1.77 \cdot 10^{0}$	0.55	$2.81 \cdot 10^{0}$	0.83	$8.78 \cdot 10^{-1}$	0.81
4	$3.54 \cdot 10^{-1}$	$1.64 \cdot 10^{0}$	2.15	$4.44 \cdot 10^{-1}$	1.99	$7.32 \cdot 10^{-1}$	1.94	$3.80 \cdot 10^{-1}$	1.21
5	$1.77 \cdot 10^{-1}$	$3.03 \cdot 10^{-1}$	2.44	$2.10 \cdot 10^{-1}$	1.08	$2.68 \cdot 10^{-1}$	1.45	$2.63 \cdot 10^{-1}$	0.53
6	$8.84 \cdot 10^{-2}$	$8.42 \cdot 10^{-2}$	1.85	$1.13 \cdot 10^{-1}$	0.89	$1.32 \cdot 10^{-1}$	1.02	$1.62 \cdot 10^{-1}$	0.7
7	$4.42 \cdot 10^{-2}$	$3.89 \cdot 10^{-2}$	1.11	$5.63 \cdot 10^{-2}$	1.01	$6.60 \cdot 10^{-2}$	1	$1.07 \cdot 10^{-1}$	0.6
8	$2.21 \cdot 10^{-2}$	$1.74 \cdot 10^{-2}$	1.16	$2.60 \cdot 10^{-2}$	1.11	$3.14 \cdot 10^{-2}$	1.07	$6.57 \cdot 10^{-2}$	0.7

(b) $\alpha := 0.50$.

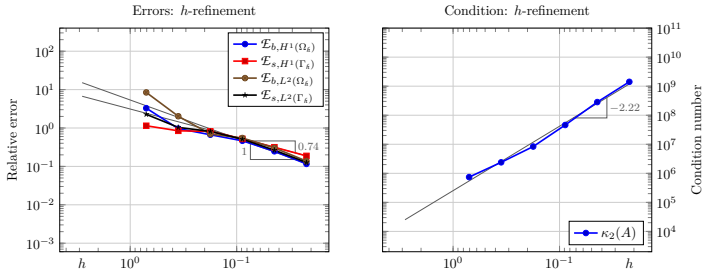
$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	$1.51 \cdot 10^{1}$	–	$2.82 \cdot 10^{0}$	–	$5.40 \cdot 10^{0}$	–	$1.22 \cdot 10^{0}$	–
2	$1.41 \cdot 10^{0}$	$1.35 \cdot 10^{1}$	0.16	$2.58 \cdot 10^{0}$	0.13	$4.95 \cdot 10^{0}$	0.13	$1.23 \cdot 10^{0}$	0
3	$7.07 \cdot 10^{-1}$	$7.60 \cdot 10^{0}$	0.83	$1.79 \cdot 10^{0}$	0.53	$2.87 \cdot 10^{0}$	0.79	$7.82 \cdot 10^{-1}$	0.65
4	$3.54 \cdot 10^{-1}$	$1.99 \cdot 10^{0}$	1.93	$4.73 \cdot 10^{-1}$	1.92	$8.57 \cdot 10^{-1}$	1.74	$3.27 \cdot 10^{-1}$	1.26
5	$1.77 \cdot 10^{-1}$	$4.86 \cdot 10^{-1}$	2.04	$1.14 \cdot 10^{-1}$	2.05	$2.97 \cdot 10^{-1}$	1.53	$1.60 \cdot 10^{-1}$	1.04
6	$8.84 \cdot 10^{-2}$	$1.24 \cdot 10^{-1}$	1.97	$2.89 \cdot 10^{-2}$	1.98	$1.20 \cdot 10^{-1}$	1.3	$7.29 \cdot 10^{-2}$	1.13
7	$4.42 \cdot 10^{-2}$	$2.92 \cdot 10^{-2}$	2.08	$7.33 \cdot 10^{-3}$	1.98	$5.38 \cdot 10^{-2}$	1.16	$3.99 \cdot 10^{-2}$	0.87
8	$2.21 \cdot 10^{-2}$	$7.19 \cdot 10^{-3}$	2.02	$2.17 \cdot 10^{-3}$	1.76	$2.54 \cdot 10^{-2}$	1.08	$2.11 \cdot 10^{-2}$	0.92

(c) $\alpha := 0.05$.

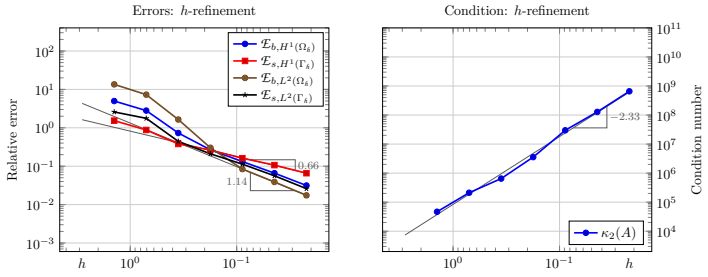
$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	$1.68 \cdot 10^{1}$	–	$3.32 \cdot 10^{0}$	–	$6.00 \cdot 10^{0}$	–	$1.23 \cdot 10^{0}$	–
2	$1.41 \cdot 10^{0}$	$1.35 \cdot 10^{1}$	0.32	$2.61 \cdot 10^{0}$	0.35	$4.94 \cdot 10^{0}$	0.28	$1.18 \cdot 10^{0}$	0.06
3	$7.07 \cdot 10^{-1}$	$7.85 \cdot 10^{0}$	0.78	$1.95 \cdot 10^{0}$	0.42	$2.98 \cdot 10^{0}$	0.73	$7.27 \cdot 10^{-1}$	0.7
4	$3.54 \cdot 10^{-1}$	$2.09 \cdot 10^{0}$	1.91	$5.17 \cdot 10^{-1}$	1.92	$8.82 \cdot 10^{-1}$	1.76	$2.96 \cdot 10^{-1}$	1.3
5	$1.77 \cdot 10^{-1}$	$5.08 \cdot 10^{-1}$	2.04	$1.21 \cdot 10^{-1}$	2.1	$2.98 \cdot 10^{-1}$	1.56	$1.31 \cdot 10^{-1}$	1.17
6	$8.84 \cdot 10^{-2}$	$1.31 \cdot 10^{-1}$	1.95	$3.11 \cdot 10^{-2}$	1.96	$1.21 \cdot 10^{-1}$	1.31	$5.90 \cdot 10^{-2}$	1.16
7	$4.42 \cdot 10^{-2}$	$3.13 \cdot 10^{-2}$	2.07	$7.34 \cdot 10^{-3}$	2.08	$5.39 \cdot 10^{-2}$	1.16	$3.09 \cdot 10^{-2}$	0.93
8	$2.21 \cdot 10^{-2}$	$7.92 \cdot 10^{-3}$	1.99	$1.86 \cdot 10^{-3}$	1.98	$2.55 \cdot 10^{-2}$	1.08	$1.52 \cdot 10^{-2}$	1.02

(d) $\alpha := 0.01$.

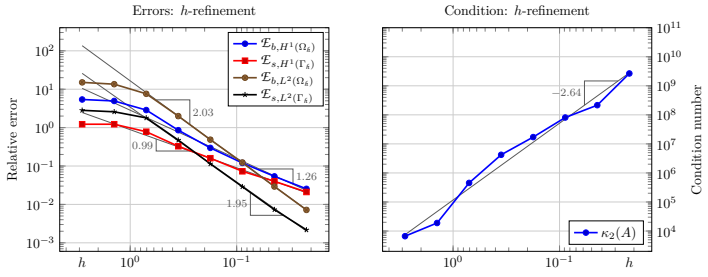
Table 4.4.: Errors in numerical solutions of elliptic 2d test problem "3", obtained using Scheme 4.2.9, $k = 1$, $\text{host}_\text{s} :=$ sipg, and no separate geometry meshes.



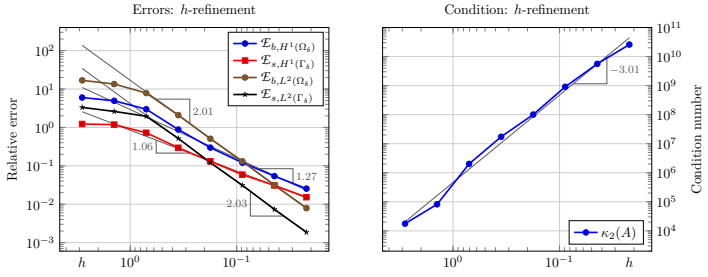
(a) $\alpha := 1.25$.



(b) $\alpha := 0.50$.



(c) $\alpha := 0.05$.



(d) $\alpha := 0.01$.

Figure 4.8.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number associated with the corresponding system of linear equations, obtained using Scheme 4.2.9, $k = 1$, $\text{host}_s :=$ sipg, and no separate geometry meshes.

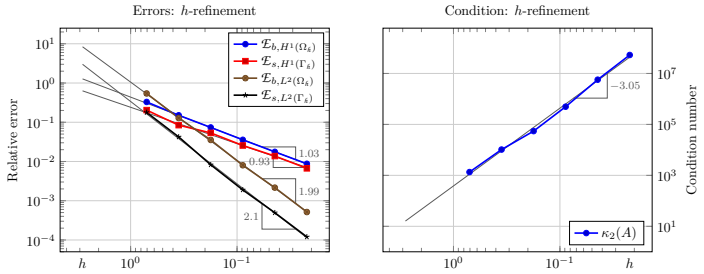
$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
3	$7.07 \cdot 10^{-1}$	$5.42 \cdot 10^{-1}$	–	$1.78 \cdot 10^{-1}$	–	$3.29 \cdot 10^{-1}$	–	$2.04 \cdot 10^{-1}$	–
4	$3.54 \cdot 10^{-1}$	$1.28 \cdot 10^{-1}$	2.08	$4.23 \cdot 10^{-2}$	2.07	$1.51 \cdot 10^{-1}$	1.13	$8.43 \cdot 10^{-2}$	1.27
5	$1.77 \cdot 10^{-1}$	$3.54 \cdot 10^{-2}$	1.85	$8.26 \cdot 10^{-3}$	2.36	$7.38 \cdot 10^{-2}$	1.03	$5.35 \cdot 10^{-2}$	0.66
6	$8.84 \cdot 10^{-2}$	$8.01 \cdot 10^{-3}$	2.14	$1.89 \cdot 10^{-3}$	2.13	$3.57 \cdot 10^{-2}$	1.05	$2.56 \cdot 10^{-2}$	1.07
7	$4.42 \cdot 10^{-2}$	$2.16 \cdot 10^{-3}$	1.89	$4.95 \cdot 10^{-4}$	1.93	$1.76 \cdot 10^{-2}$	1.02	$1.38 \cdot 10^{-2}$	0.89
8	$2.21 \cdot 10^{-2}$	$5.15 \cdot 10^{-4}$	2.07	$1.20 \cdot 10^{-4}$	2.04	$8.73 \cdot 10^{-3}$	1.01	$6.69 \cdot 10^{-3}$	1.04

(a) Full gradient stabilization, $\alpha := 1.25$.

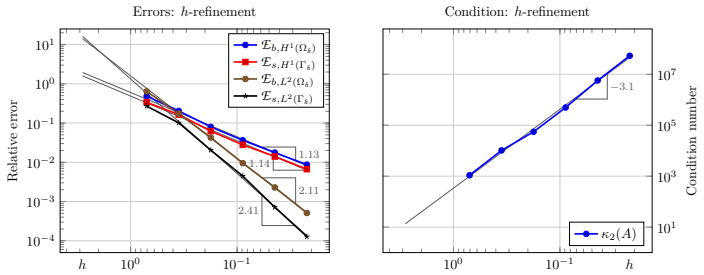
$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
3	$7.07 \cdot 10^{-1}$	$6.41 \cdot 10^{-1}$	–	$2.71 \cdot 10^{-1}$	–	$4.86 \cdot 10^{-1}$	–	$3.39 \cdot 10^{-1}$	–
4	$3.54 \cdot 10^{-1}$	$1.74 \cdot 10^{-1}$	1.88	$1.01 \cdot 10^{-1}$	1.42	$2.04 \cdot 10^{-1}$	1.25	$1.65 \cdot 10^{-1}$	1.04
5	$1.77 \cdot 10^{-1}$	$4.29 \cdot 10^{-2}$	2.02	$2.04 \cdot 10^{-2}$	2.31	$7.99 \cdot 10^{-2}$	1.35	$6.23 \cdot 10^{-2}$	1.4
6	$8.84 \cdot 10^{-2}$	$9.51 \cdot 10^{-3}$	2.17	$4.46 \cdot 10^{-3}$	2.2	$3.64 \cdot 10^{-2}$	1.14	$2.78 \cdot 10^{-2}$	1.16
7	$4.42 \cdot 10^{-2}$	$2.29 \cdot 10^{-3}$	2.05	$7.15 \cdot 10^{-4}$	2.64	$1.77 \cdot 10^{-2}$	1.04	$1.40 \cdot 10^{-2}$	0.99
8	$2.21 \cdot 10^{-2}$	$5.12 \cdot 10^{-4}$	2.16	$1.28 \cdot 10^{-4}$	2.48	$8.73 \cdot 10^{-3}$	1.02	$6.67 \cdot 10^{-3}$	1.07

(b) Normal penalty stabilization, $\alpha := 1.25$.

Table 4.5.: Errors in numerical solutions of elliptic 2d test problem "2", obtained using Scheme 4.2.15 with the two considered stabilization terms, $k = 1$, $\text{host}_s := \text{sipg}$, and no separate geometry meshes.



(a) Full gradient stabilization, $\alpha := 1.25$.



(b) Normal penalty stabilization, $\alpha := 1.25$.

Figure 4.9.: Errors in numerical solutions of elliptic 2d test problem "2" and spectral condition number, obtained using Scheme 4.2.15 with the two considered stabilization terms, $k = 1$, $\text{host}_s := \text{sipg}$, and no separate geometry meshes.

$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	–	–	–	–	–	–	–	–
2	$1.41 \cdot 10^{0}$	–	–	–	–	–	–	–	–
3	$7.07 \cdot 10^{-1}$	$6.90 \cdot 10^{0}$	–	$1.72 \cdot 10^{0}$	–	$2.67 \cdot 10^{0}$	–	$9.34 \cdot 10^{-1}$	–
4	$3.54 \cdot 10^{-1}$	$1.61 \cdot 10^{0}$	2.1	$4.91 \cdot 10^{-1}$	1.81	$7.19 \cdot 10^{-1}$	1.89	$4.15 \cdot 10^{-1}$	1.17
5	$1.77 \cdot 10^{-1}$	$4.00 \cdot 10^{-1}$	2.01	$1.54 \cdot 10^{-1}$	1.67	$2.63 \cdot 10^{-1}$	1.45	$2.08 \cdot 10^{-1}$	1
6	$8.84 \cdot 10^{-2}$	$1.05 \cdot 10^{-1}$	1.93	$4.16 \cdot 10^{-2}$	1.89	$1.13 \cdot 10^{-1}$	1.21	$8.56 \cdot 10^{-2}$	1.28
7	$4.42 \cdot 10^{-2}$	$2.68 \cdot 10^{-2}$	1.97	$1.05 \cdot 10^{-2}$	1.98	$5.28 \cdot 10^{-2}$	1.1	$4.40 \cdot 10^{-2}$	0.96
8	$2.21 \cdot 10^{-2}$	$6.84 \cdot 10^{-3}$	1.97	$2.62 \cdot 10^{-3}$	2.01	$2.53 \cdot 10^{-2}$	1.06	$2.20 \cdot 10^{-2}$	1

(a) Full gradient stabilization, $\alpha := 1.25$.

$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	–	–	–	–	–	–	–	–
2	$1.41 \cdot 10^{0}$	–	–	–	–	–	–	–	–
3	$7.07 \cdot 10^{-1}$	$5.37 \cdot 10^{0}$	–	$1.22 \cdot 10^{0}$	–	$2.09 \cdot 10^{0}$	–	$7.85 \cdot 10^{-1}$	–
4	$3.54 \cdot 10^{-1}$	$1.51 \cdot 10^{0}$	1.83	$3.72 \cdot 10^{-1}$	1.72	$7.29 \cdot 10^{-1}$	1.52	$3.99 \cdot 10^{-1}$	0.98
5	$1.77 \cdot 10^{-1}$	$4.14 \cdot 10^{-1}$	1.87	$9.81 \cdot 10^{-2}$	1.92	$2.81 \cdot 10^{-1}$	1.38	$1.85 \cdot 10^{-1}$	1.11
6	$8.84 \cdot 10^{-2}$	$1.05 \cdot 10^{-1}$	1.98	$2.49 \cdot 10^{-2}$	1.98	$1.17 \cdot 10^{-1}$	1.27	$8.24 \cdot 10^{-2}$	1.17
7	$4.42 \cdot 10^{-2}$	$2.67 \cdot 10^{-2}$	1.97	$7.42 \cdot 10^{-3}$	1.74	$5.30 \cdot 10^{-2}$	1.14	$4.31 \cdot 10^{-2}$	0.93
8	$2.21 \cdot 10^{-2}$	$6.86 \cdot 10^{-3}$	1.96	$2.88 \cdot 10^{-3}$	1.37	$2.53 \cdot 10^{-2}$	1.07	$2.21 \cdot 10^{-2}$	0.97

(b) Normal penalty stabilization, $\alpha := 1.25$.

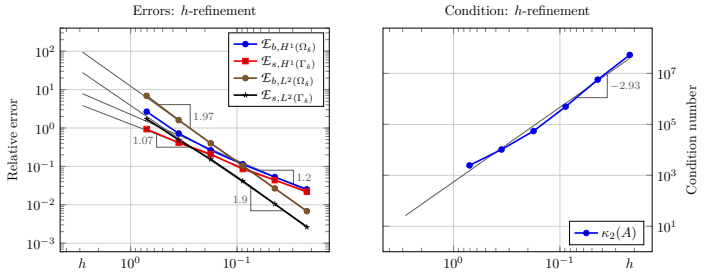
$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	–	–	–	–	–	–	–	–
2	$1.41 \cdot 10^{0}$	$1.18 \cdot 10^{1}$	–	$2.28 \cdot 10^{0}$	–	$4.39 \cdot 10^{0}$	–	$1.47 \cdot 10^{0}$	–
3	$7.07 \cdot 10^{-1}$	$6.33 \cdot 10^{0}$	0.9	$1.49 \cdot 10^{0}$	0.62	$2.43 \cdot 10^{0}$	0.85	$8.69 \cdot 10^{-1}$	0.75
4	$3.54 \cdot 10^{-1}$	$1.76 \cdot 10^{0}$	1.85	$4.20 \cdot 10^{-1}$	1.82	$7.94 \cdot 10^{-1}$	1.62	$3.64 \cdot 10^{-1}$	1.25
5	$1.77 \cdot 10^{-1}$	$4.54 \cdot 10^{-1}$	1.96	$1.08 \cdot 10^{-1}$	1.96	$2.92 \cdot 10^{-1}$	1.44	$1.75 \cdot 10^{-1}$	1.06
6	$8.84 \cdot 10^{-2}$	$1.17 \cdot 10^{-1}$	1.96	$2.76 \cdot 10^{-2}$	1.96	$1.20 \cdot 10^{-1}$	1.28	$7.85 \cdot 10^{-2}$	1.15
7	$4.42 \cdot 10^{-2}$	$2.93 \cdot 10^{-2}$	1.99	$6.93 \cdot 10^{-3}$	2	$5.40 \cdot 10^{-2}$	1.16	$4.16 \cdot 10^{-2}$	0.92
8	$2.21 \cdot 10^{-2}$	$7.42 \cdot 10^{-3}$	1.98	$1.75 \cdot 10^{-3}$	1.98	$2.55 \cdot 10^{-2}$	1.08	$2.09 \cdot 10^{-2}$	0.99

(c) Full gradient stabilization, $\alpha := 0.50$.

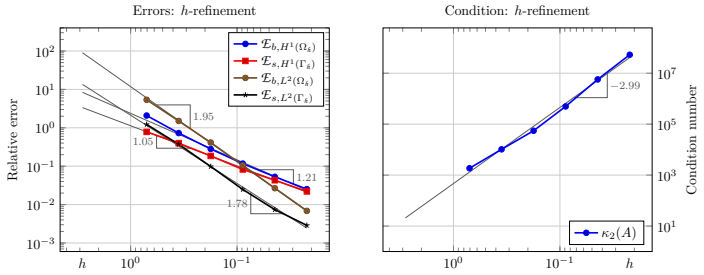
$\mathfrak{r}$	h	$\mathcal{E}_{b,L^2(\Omega_h)}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h)}$	eoc	$\mathcal{E}_{b,H^1(\Omega_h)}$	eoc	$\mathcal{E}_{s,H^1(\Gamma_h)}$	eoc
1	$2.83 \cdot 10^{0}$	–	–	–	–	–	–	–	–
2	$1.41 \cdot 10^{0}$	$8.93 \cdot 10^{0}$	–	$1.74 \cdot 10^{0}$	–	$3.29 \cdot 10^{0}$	–	$1.31 \cdot 10^{0}$	–
3	$7.07 \cdot 10^{-1}$	$5.67 \cdot 10^{0}$	0.66	$1.39 \cdot 10^{0}$	0.32	$2.20 \cdot 10^{0}$	0.58	$9.62 \cdot 10^{-1}$	0.44
4	$3.54 \cdot 10^{-1}$	$1.62 \cdot 10^{0}$	1.81	$4.33 \cdot 10^{-1}$	1.69	$7.81 \cdot 10^{-1}$	1.49	$4.24 \cdot 10^{-1}$	1.18
5	$1.77 \cdot 10^{-1}$	$4.19 \cdot 10^{-1}$	1.95	$1.14 \cdot 10^{-1}$	1.92	$2.97 \cdot 10^{-1}$	1.39	$1.88 \cdot 10^{-1}$	1.18
6	$8.84 \cdot 10^{-2}$	$1.10 \cdot 10^{-1}$	1.93	$2.90 \cdot 10^{-2}$	1.98	$1.22 \cdot 10^{-1}$	1.29	$8.03 \cdot 10^{-2}$	1.22
7	$4.42 \cdot 10^{-2}$	$2.84 \cdot 10^{-2}$	1.95	$7.02 \cdot 10^{-3}$	2.05	$5.41 \cdot 10^{-2}$	1.17	$4.17 \cdot 10^{-2}$	0.95
8	$2.21 \cdot 10^{-2}$	$7.46 \cdot 10^{-3}$	1.93	$1.75 \cdot 10^{-3}$	2	$2.55 \cdot 10^{-2}$	1.09	$2.09 \cdot 10^{-2}$	1

(d) Normal penalty stabilization, $\alpha := 0.50$.

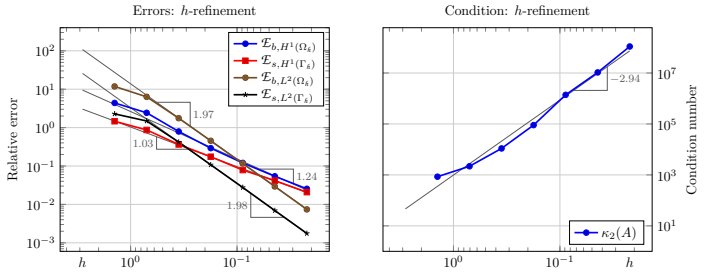
Table 4.6.: Errors in numerical solutions of elliptic 2d test problem "3", obtained using Scheme 4.2.15 with the two considered stabilization terms, $k = 1$, $\text{host}_\text{s} :=$ sipg, and no separate geometry meshes.



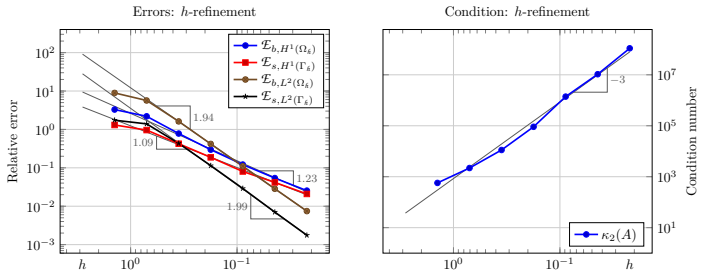
(a) Full gradient stabilization, $\alpha := 1.25$.



(b) Normal penalty stabilization, $\alpha := 1.25$.



(c) Full gradient stabilization, $\alpha := 0.50$.



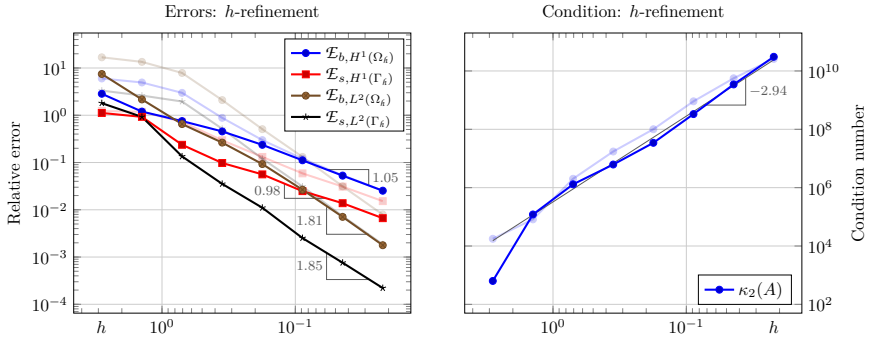
(d) Normal penalty stabilization, $\alpha := 0.50$.

Figure 4.10.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number, obtained using Scheme 4.2.15 with the two considered stabilization terms, $k = 1$, $\text{host}_s := \text{sipg}$, and no separate geometry meshes.

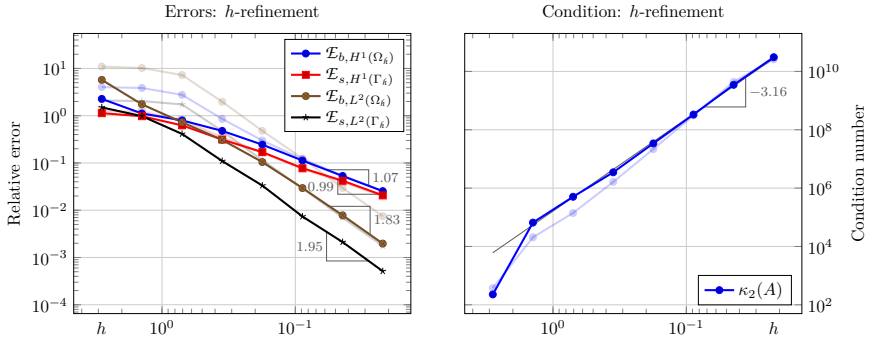
optimal order convergence, but already for much larger values of the narrow band parameter α than with the unstabilized scheme. At the same time, the errors gained with these large values of α are as small as those produced by the unstabilized scheme with small values of α. This indicates that the convergence behavior of the stabilized scheme with either mechanism is less sensitive to choosing the width of the narrow band. Taking a closer look at the spectral condition number, we have $\kappa_2(A) \in \mathcal{O}(h^{-3})$ in the parameter regime $1.25 \geq \alpha \geq 0.50$ which we are considering in Figure 4.9 and Figure 4.10. Note that $\kappa_2(A)$ hence asymptotically is the same as with the unstabilized scheme with small values of α. However, the size of $\kappa_2(A)$ is smaller by up to two orders of magnitude. Thus, both its convergence and conditioning properties render the stabilized scheme with one of the two stabilization mechanisms the preferable choice.

For the unstabilized Scheme 4.2.9 and its stabilized variant Scheme 4.2.15, we next investigate the effect of employing separate geometry meshes to obtain a geometry description which is more accurate. Results for elliptic 2d test problem "3" which use 2 additional uniform refinements for each of the entities in the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$, i.e. results which use geometry meshes with $\hbar = h/4$, are shown in Figure 4.11. To allow for direct comparison, the figure also contains results that are obtained in the same setting, but without using separate geometry meshes. For each single variant of the scheme, we see that similar orders of convergence and asymptotics of the spectral condition number $\kappa_2(A)$ are achieved with and without using separate geometry meshes. However, using separate geometry meshes reduces all errors for large values of h, especially when being combined with normal penalty stabilization. Moreover, it generally reduces the error with respect to each component's $L^2(\cdot)$-norm. Interestingly, this effect is most pronounced for the unstabilized scheme and $\mathcal{E}_{s,L^2(\Gamma_\hbar)}$, and also $\mathcal{E}_{s,H^1(\Gamma_\hbar)}$ decreases in a uniform way. Scheme 4.2.9 hence outperforms its stabilized variant in this special case. Without showing more evidence here in this thesis, we would like to point out that most of these benefits of separate geometry meshes are observable also for larger values of the narrow band parameter α and for other test problems, e.g. elliptic 2d test problem "2".

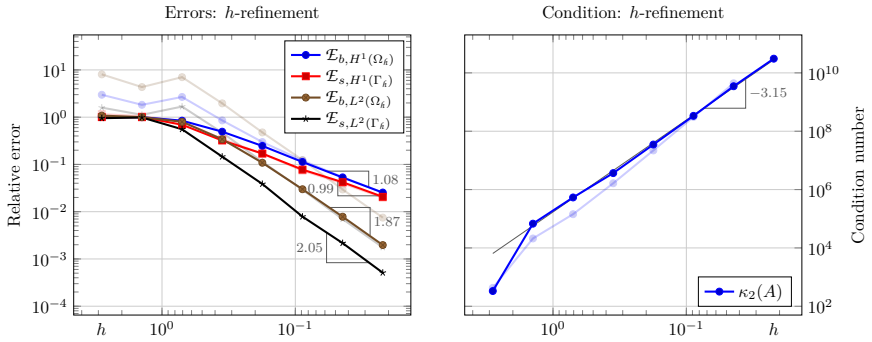
For the reasons stated in Section 4.2.2, we finally study the effect of using SWIPG instead of SIPG as host DG formulation for the surface part of the problem. To do so, we repeat the computations which lead to the results from Figure 4.11, this time choosing $\text{host}_\text{s} := \text{swipg}$. As shown in Figure 4.12, both host DG formulations yield nearly identical errors. Furthermore, both formulations result in systems of linear equations with a spectral condition number $\kappa_2(A)$ of similar asymptotics. In particular, we approximately have $\kappa_2(A) \in \mathcal{O}(h^{-3})$. Together with normal penalty stabilization, the SWIPG formulation even seems to lower the asymptotics of $\kappa_2(A)$. In addition, it reduces the spectral condition number for each single variant of the scheme. We note that the value of $\kappa_2(A)$ goes down by up to two orders of magnitude for the settings which we are considering in Figure 4.12. Also without using



(a) Unstabilized scheme, $\alpha := 0.01$.

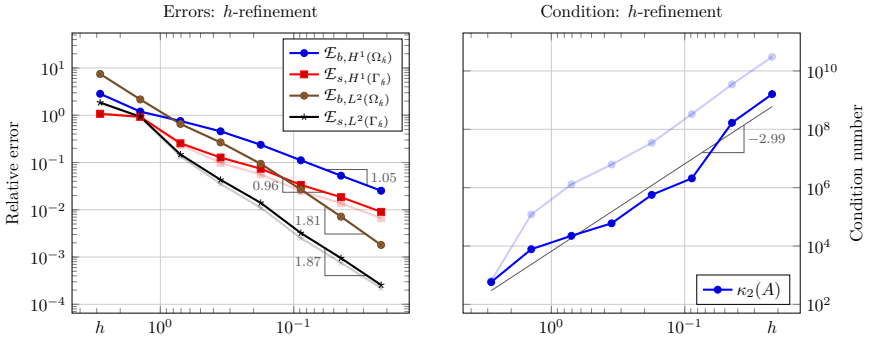


(b) Full gradient stabilization, $\alpha := 0.01$.

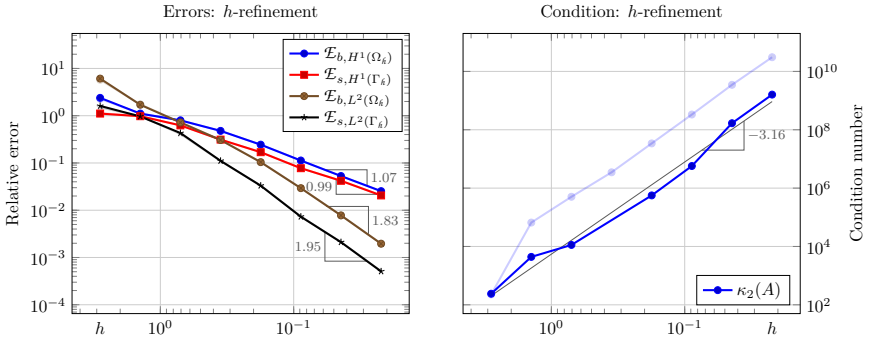


(c) Normal penalty stabilization, $\alpha := 0.01$.

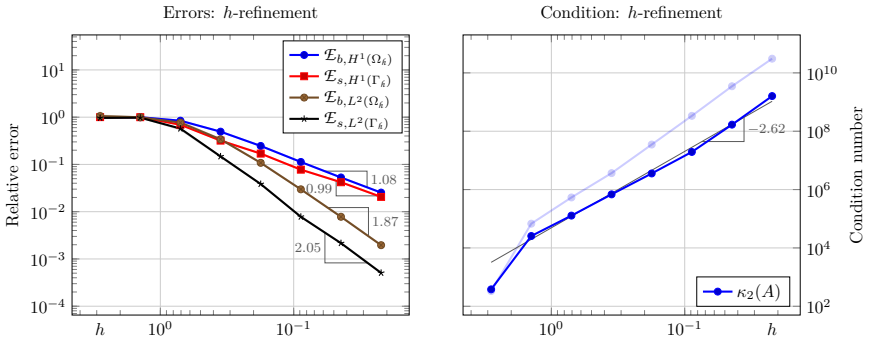
Figure 4.11.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number associated with the corresponding system of linear equations, obtained using Scheme 4.2.9 or Scheme 4.2.15 with the two considered stabilization terms, $k = 1$, $\text{host}_\text{s} := \text{sipg}$, and separate geometry meshes with $\hbar = h/4$. Corresponding results that are obtained without using separate geometry meshes are depicted transparently in the background.



(a) Unstabilized scheme, $\alpha := 0.01$.



(b) Full gradient stabilization, $\alpha := 0.01$.



(c) Normal penalty stabilization, $\alpha := 0.01$.

Figure 4.12.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number associated with the corresponding system of linear equations, obtained using Scheme 4.2.9 or Scheme 4.2.15 with the two considered stabilization terms, $k = 1$, $\text{host}_\text{s} := \text{swipg}$, and separate geometry meshes with $\hbar = h/4$. Corresponding results that are obtained using $\text{host}_\text{s} := \text{sipg}$ are depicted transparently in the background.

separate geometry meshes, for larger values of the narrow band parameter α, and for other test problems, the SWIPG formulation improves $\kappa_2(A)$ while yielding comparable convergence results. Therefore, SWIPG indeed seems to be the preferable choice for the surface part of the problem.

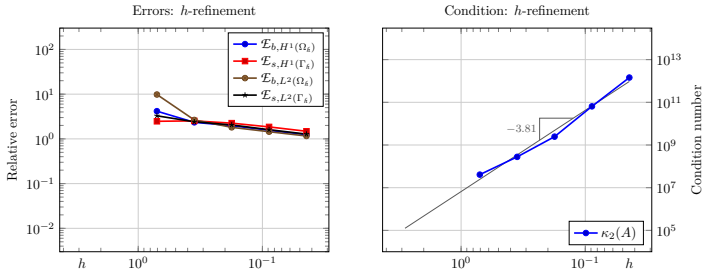
Numerical study: h-refinement and polynomial degree $k = 2$

As a second numerical study, we run similar experiments using polynomial degree $k = 2$. Again, we investigate relative errors, associated experimental orders of convergence and the spectral condition number of the system matrix with respect to the width h of the fundamental mesh, performing h-refinement ($h \to 0$, α fixed). In the course of this, we arbitrarily choose SIPG as host DG formulation for the surface part of the problem.

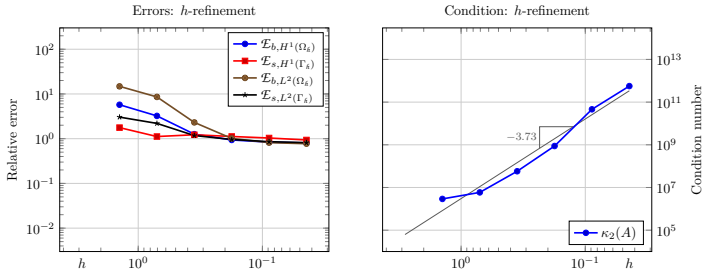
As with the above study for $k = 1$, we first consider Scheme 4.2.9 without using separate geometry meshes. Results for elliptic 2d test problem "3" are shown in Figure 4.13. We observe that the unstabilized scheme does not perform well. Even if the narrow band parameter α is chosen small, convergence either stagnates or convergence rates are not satisfactory. Errors tend to be even larger than in the same experiments with $k = 1$, cf. Figure 4.8. Meanwhile, the spectral condition number $\kappa_2(A)$ associated with solving the corresponding linear system (4.37) grows at a high rate with respect to h, especially for large values of α. In comparison with the corresponding experiments for $k = 1$, values of $\kappa_2(A)$ are also larger by multiple orders of magnitude.

Our experiments above have shown that these deficiencies of Scheme 4.2.9 are not apparent for polynomial degree $k = 1$. They seem to emerge from the higher polynomial degree. Since the error which results from geometry approximation is known to have a crucial influence on the order of convergence in higher order schemes, we next investigate the effect of employing separate geometry meshes to obtain a geometry reconstruction which is more accurate. We repeat the computations which lead to the results from Figure 4.13, this time using 5 additional uniform refinements for each of the entities in the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$, i.e. we use geometry meshes with $\hbar = h/32$. The results are shown in Figure 4.14. Unfortunately, these results indicate that the smaller geometry approximation error does not improve a lot. Although errors for large values of h and errors for small values of α are slightly reduced, convergence rates are still not satisfactory. At the same time, the spectral condition number $\kappa_2(A)$ only decreases slightly. It is still much larger than with the corresponding experiments for $k = 1$. Therefore, the unstabilized scheme does not seem suitable for obtaining higher order schemes.

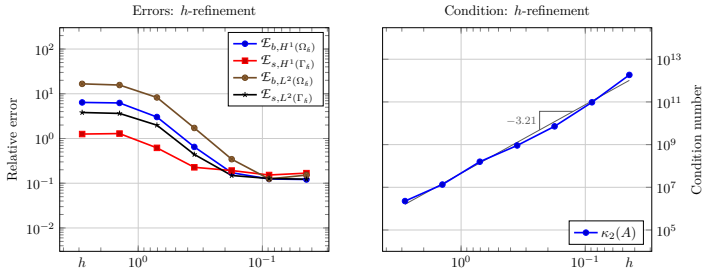
We now perform the same two experiments for Scheme 4.2.15, i.e. for the stabilized scheme, together with full gradient stabilization or normal penalty stabilization. As with the unstabilized scheme, we start without using separate geometry meshes. Working with elliptic 2d test problem "3" as usual, it is shown in Figure 4.15 that the stabilized scheme produces reasonable results. As with our experiments for $k = 1$, the two stabilization mechanisms again



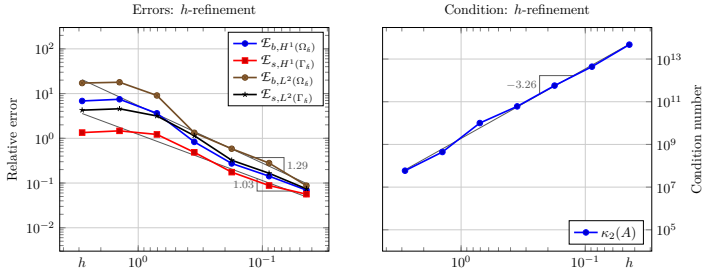
(a) $\alpha := 1.25$.



(b) $\alpha := 0.50$.

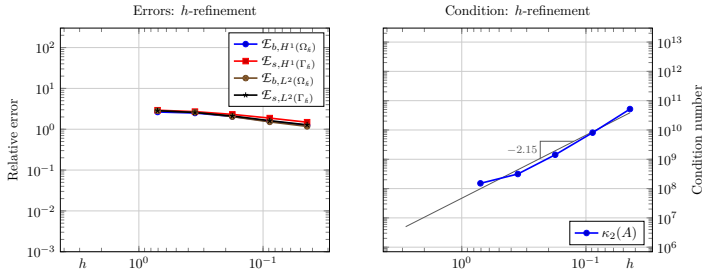


(c) $\alpha := 0.05$.

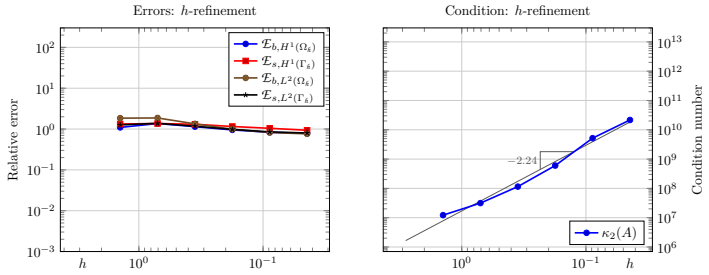


(d) $\alpha := 0.01$.

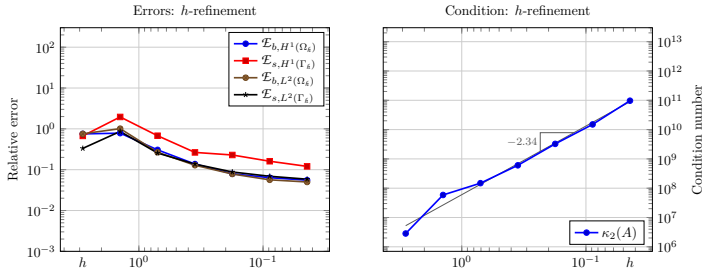
Figure 4.13.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number associated with the corresponding system of linear equations, obtained using Scheme 4.2.9, $k = 2$, $\text{host}_\text{s} :=$ sipg, and no separate geometry meshes.



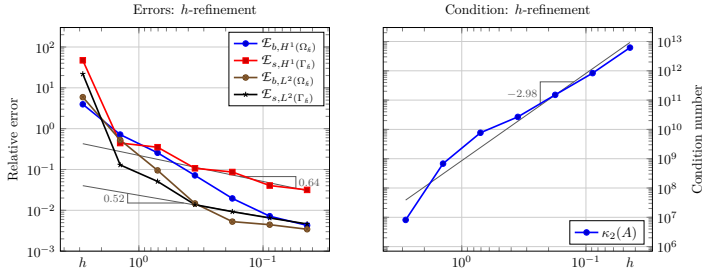
(a) $\alpha := 1.25$.



(b) $\alpha := 0.50$.

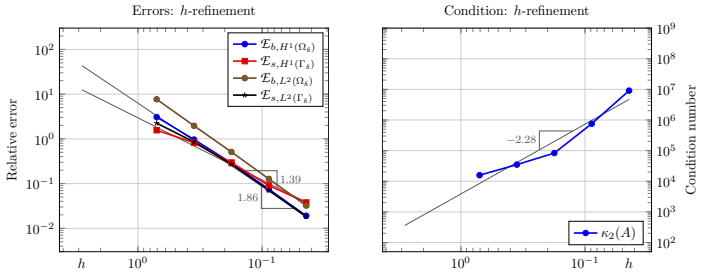


(c) $\alpha := 0.05$.

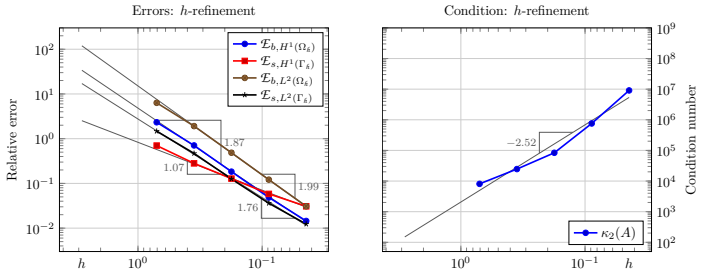


(d) $\alpha := 0.01$.

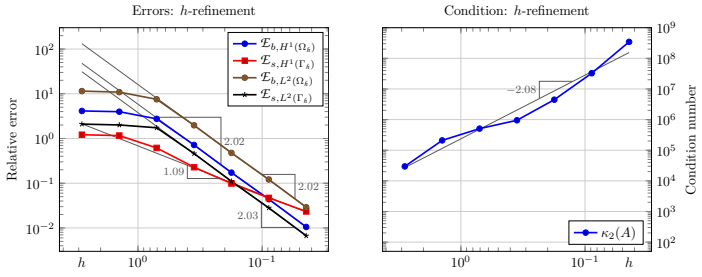
Figure 4.14.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number associated with the corresponding system of linear equations, obtained using Scheme 4.2.9, $k = 2$, $\text{host}_\text{s} := \text{sipg}$, and separate geometry meshes with $\hbar = h/32$.



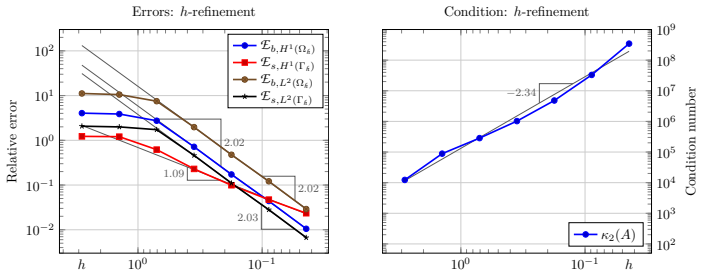
(a) Full gradient stabilization, $\alpha := 1.25$.



(b) Normal penalty stabilization, $\alpha := 1.25$.



(c) Full gradient stabilization, $\alpha := 0.05$.



(d) Normal penalty stabilization, $\alpha := 0.05$.

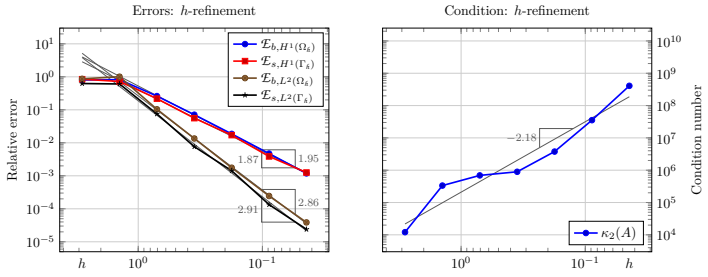
Figure 4.15.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number, obtained using Scheme 4.2.15 with the two considered stabilization terms, $k = 2$, $\text{host}_s := \text{sipg}$, and no separate geometry meshes.

perform equally well. They yield similar errors, especially for small values of h and small values of the narrow band parameter α. Also the spectral condition numbers of their corresponding system matrix in linear system (4.37) are nearly identical for small values of h. The scheme exhibits a promising convergence behavior already for large values of α and it is clearly observable for small values of α that both components of the numerical solution $(u_{b,h}, u_{s,h})$ converge to their corresponding component of the solution pair (u_b, u_s). This convergence is of suboptimal order, though. We obtain order 1 with respect to $\mathcal{E}_{s,H^1(\Gamma_\hbar)}$, and order 2 with respect to $\mathcal{E}_{b,H^1(\Omega_\hbar)}$, $\mathcal{E}_{b,L^2(\Omega_\hbar)}$ and $\mathcal{E}_{s,L^2(\Gamma_\hbar)}$. In comparison with optimal order convergence, convergence of the scheme hence is reduced by order 1 in $\|\cdot\|_{H^1(\Gamma_\hbar)}$ and $\|\cdot\|_{L^2(\Gamma_\hbar)}$ regarding the surface component $u_{s,h}$, and by order 1 in $\|\cdot\|_{L^2(\Omega_\hbar)}$ regarding the bulk component $u_{b,h}$. Taking a closer look at the spectral condition number $\kappa_2(A)$, we see that its values are in a similar range as with corresponding experiments for $k = 1$, cf. Figure 4.10. The dependency on h is slightly different, though, which can be particularly observed by comparing Figure 4.15a with Figure 4.10a, and Figure 4.15b with Figure 4.10b. For small values of h, the values of $\kappa_2(A)$ that are obtained with $k = 2$ are larger than the corresponding values that are obtained with $k = 1$. But, interestingly, the growth rate of $\kappa_2(A)$ with respect to h is a little smaller when using polynomial degree $k = 2$.

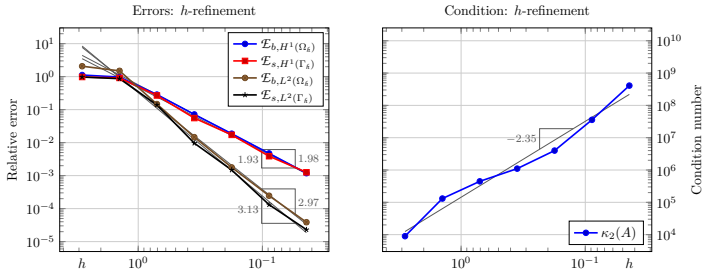
Repeating the computations which lead to the results from Figure 4.15, this time using geometry meshes with $\hbar = h/32$, it can be observed that the stabilized scheme is capable of achieving optimal order convergence with respect to the two H^1-norms and the two L^2-norms that define the relative errors which we are considering. In particular, we obtain the results shown in Figure 4.16. Provided that the narrow band parameter α is chosen small enough (notably smaller than with corresponding experiments for $k = 1$), both components of the numerical solution converge with order 2 in $\|\cdot\|_{H^1(\Omega_\hbar)}$ and $\|\cdot\|_{H^1(\Gamma_\hbar)}$, respectively, and with order 3 in $\|\cdot\|_{L^2(\Omega_\hbar)}$ and $\|\cdot\|_{L^2(\Gamma_\hbar)}$. For the stabilized scheme, the issue of suboptimal order convergence can thus be cured by using a geometry reconstruction which is more accurate. Also the spectral condition number $\kappa_2(A)$ is in a reasonable range. As observable by comparing Figure 4.16a with Figure 4.15c, and Figure 4.16b with Figure 4.15d, we obtain very similar spectral condition numbers with and without using separate geometry meshes. Therefore, Scheme 4.2.15 and both stabilization mechanisms seem suitable for obtaining higher order schemes with reasonable conditioning properties.

Numerical study: α-refinement

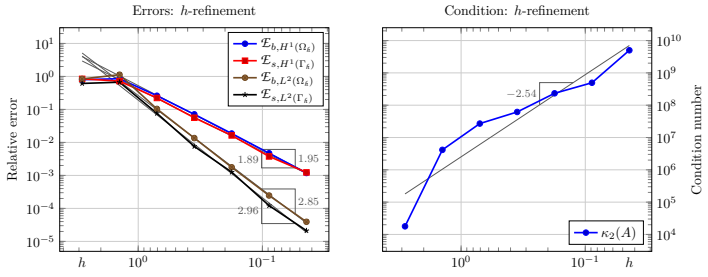
The above h-refinement studies indicate that approximation errors and the spectral condition number which is associated with solving the corresponding system of linear equations both depend on the narrow band parameter α. We now study this aspect in a more detailed way by investigating relative errors and the spectral condition number of the system matrix with respect to α. To



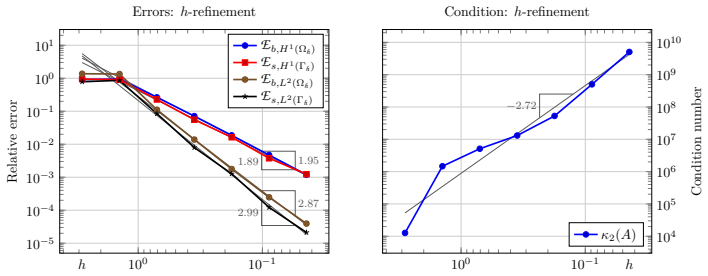
(a) Full gradient stabilization, $\alpha := 0.05$.



(b) Normal penalty stabilization, $\alpha := 0.05$.



(c) Full gradient stabilization, $\alpha := 0.01$.



(d) Normal penalty stabilization, $\alpha := 0.01$.

Figure 4.16.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number, obtained using Scheme 4.2.15 with the two considered stabilization terms, $k = 2$, $\text{host}_s := \text{sipg}$, and separate geometry meshes with $\hbar = h/32$.

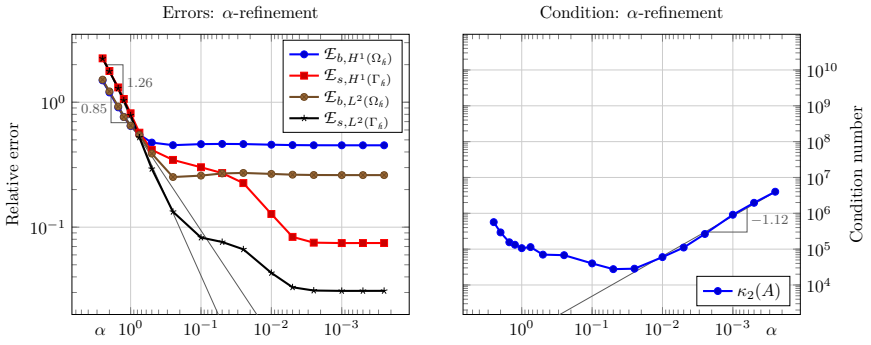
do so, we employ a strategy which we call α-refinement. Keeping the width h of the fundamental mesh fixed, we perform computations while considering a sequence of narrow bands with $\alpha \to 0$.

Taking elliptic 2d test problem "3" as an example, we particularly study α-dependency of the unstabilized Scheme 4.2.9, and of each of its stabilized variants that are given by Scheme 4.2.15 with one of the two stabilization terms considered in this thesis. In the course of this, we minimize the influence of geometry approximation errors by employing a separate geometry mesh of adequate accuracy and we arbitrarily choose SWIPG as host DG formulation for the surface part of the problem. Choosing SIPG instead produces similar results, accompanied by larger values of the spectral condition number. Results for polynomial degree $k = 1$ which use geometry meshes with $\hbar = h/4$ are shown in Figure 4.17, and results for polynomial degree $k = 2$ which use geometry meshes with $\hbar = h/32$ are shown in Figure 4.18.

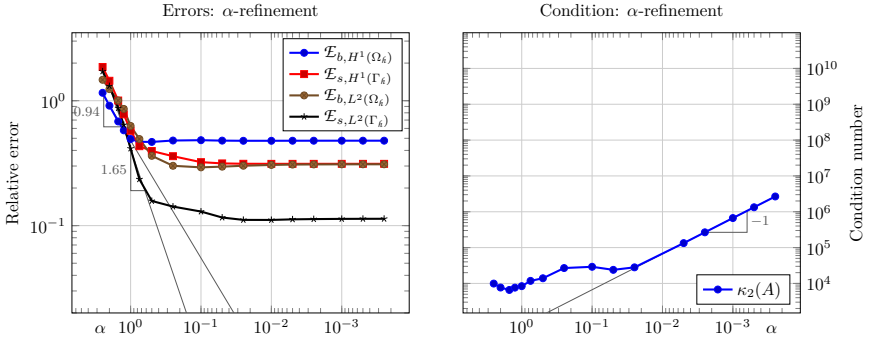
For each single variant of the scheme and for both polynomial degrees, a similar pattern in the convergence behavior with respect to α can be observed. The considered error measures diminish with decreasing values of α and they eventually reach a constant level. For each error measure, this level depends on the polynomial degree. Roughly speaking, the final error level is smaller for $k = 2$ by one order of magnitude. To some extent, it furthermore depends on the variant of the scheme. Regarding the final error level for a fixed polynomial degree, particularly for $k = 2$, we see that all variants of the scheme yield almost identical bulk errors $\mathcal{E}_{b,H^1(\Omega_\hbar)}$ and $\mathcal{E}_{b,L^2(\Omega_\hbar)}$ for the smallest values of α which we are considering in Figure 4.17 and in Figure 4.18. In contrast, the surface errors $\mathcal{E}_{s,H^1(\Gamma_\hbar)}$ and $\mathcal{E}_{s,L^2(\Gamma_\hbar)}$ are similar for both stabilization mechanisms, but the unstabilized scheme reaches error levels that are more or less one order of magnitude smaller. With the unstabilized scheme, the errors in the surface part of the solution hence are more sensitive to the narrow band parameter α than those in the bulk part.

This convergence behavior can be explained by means of different errors which result from distinct sources. On the one hand, we have the usual DG discretization error which arises from applying the UDG method. It depends on the width h of the fundamental mesh and on the polynomial degree k. For each k, this error is fixed since we are considering a fixed h in the current study. On the other hand, we have an additional error due to the integral approximations that are performed. We recall that Scheme 4.2.9 results from applying Theorem 4.2.7 to approximate the last integral in equation (4.18b) by an integral over the reconstructed hypersurface $\Gamma_\hbar$. This approximation error depends on the narrow band parameter α and vanishes for $\alpha \to 0$. In combination, both error sources explain why the considered error measures decrease with decreasing values of α and reach a k-dependent level in the end. The α-dependent integral approximation error dominates for large values of α, but is dominated by the DG discretization error for small values of α.

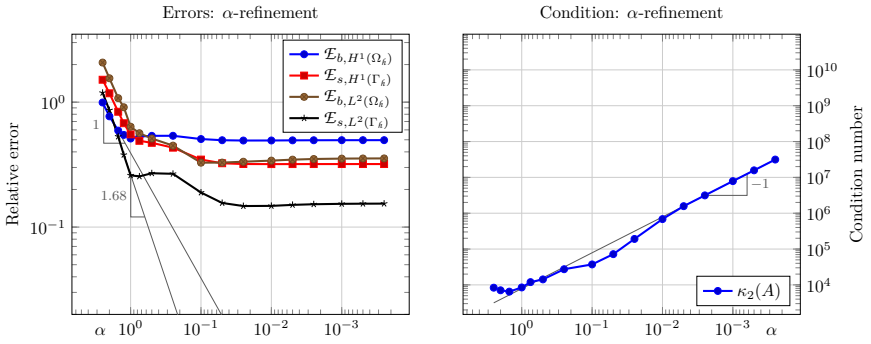
The influence of the particular scheme on the final level of the surface errors $\mathcal{E}_{s,H^1(\Gamma_\hbar)}$ and $\mathcal{E}_{s,L^2(\Gamma_\hbar)}$ can be explained by the effect of our stabilization



(a) Unstabilized scheme.

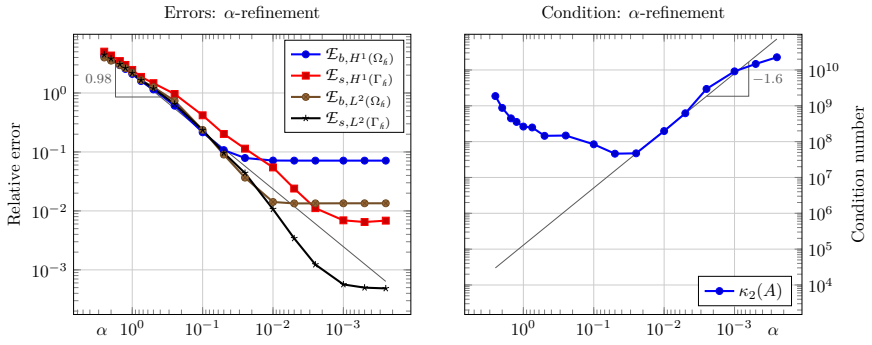


(b) Full gradient stabilization.

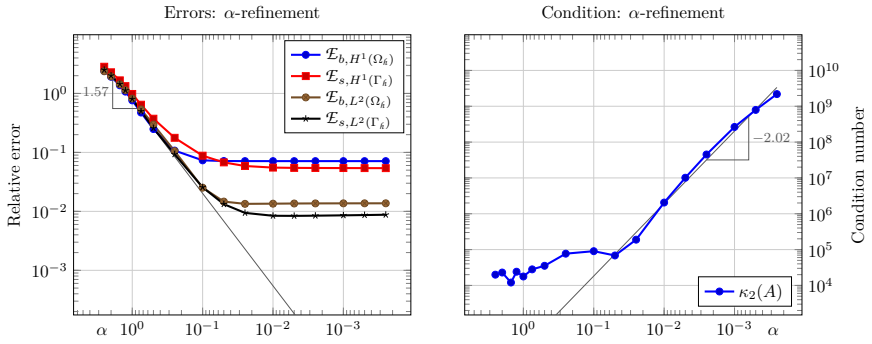


(c) Normal penalty stabilization.

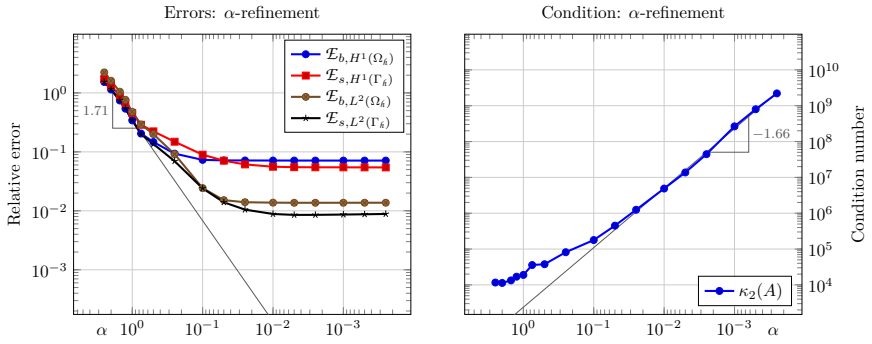
Figure 4.17.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number, obtained using α-refinement, Scheme 4.2.9 or Scheme 4.2.15 with the two considered stabilization terms, $k = 1$, $\text{host}_\text{s} := \text{swipg}$, $\mathfrak{r} = 4$ ($h \approx 3.54 \cdot 10^{-1}$), and a separate geometry mesh with $\hat{h} = h/4$.



(a) Unstabilized scheme.



(b) Full gradient stabilization.



(c) Normal penalty stabilization.

Figure 4.18.: Errors in numerical solutions of elliptic 2d test problem "3" and spectral condition number, obtained using α-refinement, Scheme 4.2.9 or Scheme 4.2.15 with the two considered stabilization terms, $k = 2$, $\text{host}_\text{s} := \text{swipg}$, $\mathfrak{r} = 4$ ($h \approx 3.54 \cdot 10^{-1}$), and a separate geometry mesh with $\hat{h} = h/32$.

terms. Those terms impose additional constraints on the surface component of the set of possible solutions. By filtering out solutions with a surface component which has a steep gradient in normal direction, these constraints effectively reduce the solution space. Hence, although taking care of robust convergence behavior with respect to h, the stabilization terms can increase the h-dependent part of the discretization error, particularly in the surface component of the solution. Note that we have observed the same effect in Figure 4.11 and in Figure 4.12. We would like to point out that this effect can only be observed when using separate geometry meshes, i.e., if the geometry reconstruction is accurate enough. It is known that geometry approximation errors are contained in the h-dependent part of the discretization error. Without using separate geometry meshes, these geometry-related errors seem to be large enough for the additional stabilization terms to not have a notable negative influence on the considered error measures.

Looking at the spectral condition number, no global relationship between $\kappa_2(A)$ and α can be identified for any single variant of the scheme. In addition, we observe no clear relationship for large values of α. However, we have $\kappa_2(A) \in \mathcal{O}(\alpha^{-1})$ for small values of α and polynomial degree $k = 1$, and we approximately have $\kappa_2(A) \in \mathcal{O}(\alpha^{-2})$ for small values of α and polynomial degree $k = 2$.

4.3.2. Linear parabolic model problems

Construction of analytical test problems

To further investigate Scheme 4.2.18 and Scheme 4.2.19 from Section 4.2.5, we construct specific models from the class of parabolic model problems (4.1) which has been introduced in Section 4.1.1. These specific models are linear parabolic models with a known analytical solution. As with the linear elliptic test problems from Section 4.3.1, the analytical solution allows for performing numerical convergence studies for our schemes.

In view of Remark 4.3.1 and for the sake of simplicity with respect to the availability of analytical solutions, we again choose the circular geometry $\Omega := \left\{\underline{\mathbf{x}} \in \mathbb{R}^d \;\middle|\; |\underline{\mathbf{x}}| < 1\right\}$ with $d := 2$ in class (4.1), and employ constant, scalar diffusivities $\mathcal{D}_b := \mathcal{I}$ and $\mathcal{D}_s := \mathcal{I}$. We obtain the restricted problem of finding $u_b \colon \Omega \times [0,T] \to \mathbb{R}$ and $u_s \colon \Gamma \times [0,T] \to \mathbb{R}$ with

$$\partial_t u_b - \Delta u_b = f_b(u_b) \qquad \text{in} \quad \Omega \times (0,T], \tag{4.39a}$$

$$-\nabla u_b \cdot \boldsymbol{\nu} = -f_{b,s}(u_b, u_s) \qquad \text{on} \quad \Gamma \times (0,T], \tag{4.39b}$$

$$\partial_t u_s - \Delta_\Gamma u_s = f_{s,b}(u_b, u_s) + f_s(u_s) \qquad \text{on} \quad \Gamma \times (0,T], \tag{4.39c}$$

where $u_b(\cdot, 0)$ and $u_s(\cdot, 0)$ are given initial values. As with the corresponding elliptic problem (4.36), we use coupling terms $f_{b,s}(u_b, u_s) := -f_{s,b}(u_b, u_s)$, where $f_{s,b}(u_b, u_s) := u_b|_{\Gamma\times(0,T]} - u_s$. Furthermore, we employ a term $f_b(u_b)$ that is linear in u_b, and a term $f_s(u_s)$ that is linear in u_s.

Implementing the idea of the method of manufactured solutions again (cf. Section 4.3.1), we construct specific models (4.39) with some known analytical solution using the following approach:

1. Choose an appropriate u_b, e.g. the product of an eigenfunction of the Laplacian and of the expression $\exp(\lambda t)$, where $\lambda \in \mathbb{R}$ is the corresponding eigenvalue.
2. Determine $f_b(u_b)$ by means of bulk equation (4.39a), yielding $f_b(u_b) \equiv 0$ if u_b is chosen to be the product of an eigenfunction of the Laplacian and of the expression $\exp(\lambda t)$, where $\lambda \in \mathbb{R}$ is the corresponding eigenvalue.
3. Calculate u_s from boundary condition (4.39b).
4. Identify $f_s(u_s)$ via surface equation (4.39c).
5. Obtain initial values $u_b(\cdot, 0)$ and $u_s(\cdot, 0)$ by restricting the expressions $u_b(\underline{\mathbf{x}}, t)$ and $u_s(\underline{\mathbf{x}}, t)$ to $t = 0$.

By applying this procedure, linear parabolic test problems can be constructed with the help of Remark 4.3.1. Some test problems that have been constructed this way are specified in Table 4.7. In the following, we particularly look at results for parabolic 2d test problem "3" and results for parabolic 2d test problem "4". Graphs of the corresponding solution pairs (u_b, u_s) are depicted in Figure 4.19. Parabolic 2d test problem "1" which is also specified in Table 4.7 will not serve as a test problem within the scope of this thesis. It yields comparable results, though.

Common simulation parameters

In our numerical studies for linear parabolic model problems, we use the same simulation parameters as those which are listed in Section 4.3.1. This includes using SIPG as host DG formulation for the bulk part of the problem. We use the penalty parameter $\gamma_b := 5$ both for parabolic 2d test problem "3" and for parabolic 2d test problem "4".

The free parameters of each test problem are chosen exactly as proposed in Table 4.7. Note that, given this particular choice of test problem parameters, the function values of solutions to our test problems decay rapidly with respect to time, cf. Figure 4.19. Therefore, we choose a small maximum observation time $T := 0.1$ for each test problem.

In each simulation run, we perform several time steps. In particular, we use a uniform step size of the form $\tau^n := 0.01\, h^z$ with some constant $z \in \mathbb{N}$.

	Parabolic 2d test problem "1"
$u_b(\underline{\mathbf{x}},t)$	$c\exp(-\beta^2 t)\sin(\beta \mathrm{x}_1)$ with $c,\beta\in\mathbb{R}$ (e.g. $c=1,\beta=1\pi$)
$f_b(u_b)(\underline{\mathbf{x}},t)$	0
$u_s(\underline{\mathbf{x}},t)$	$c\exp(-\beta^2 t)\big[\beta \mathrm{x}_1\cos(\beta \mathrm{x}_1)+\sin(\beta \mathrm{x}_1)\big]$
$f_s(u_s)(\underline{\mathbf{x}},t)$	$c\beta\exp(-\beta^2 t)\big[\beta^2\mathrm{x}_0^2\mathrm{x}_1\cos(\beta \mathrm{x}_1)$ $+4\beta\mathrm{x}_0^2\sin(\beta \mathrm{x}_1)+(3-\beta^2)\mathrm{x}_1\cos(\beta \mathrm{x}_1)-2\beta\sin(\beta \mathrm{x}_1)\big]$

	Parabolic 2d test problem "3"
$u_b(\underline{\mathbf{x}},t)$	$c\exp\big(-(\alpha^2+\beta^2)t\big)\sin(\alpha \mathrm{x}_0)\sin(\beta \mathrm{x}_1)$ with $c,\alpha,\beta\in\mathbb{R}$ (e.g. $c=1,\alpha=0.5\pi,\beta=1\pi$)
$f_b(u_b)(\underline{\mathbf{x}},t)$	0
$u_s(\underline{\mathbf{x}},t)$	$c\exp\big(-(\alpha^2+\beta^2)t\big)\big[\beta \mathrm{x}_1\sin(\alpha \mathrm{x}_0)\cos(\beta \mathrm{x}_1)$ $+\alpha \mathrm{x}_0\cos(\alpha \mathrm{x}_0)\sin(\beta \mathrm{x}_1)+\sin(\alpha \mathrm{x}_0)\sin(\beta \mathrm{x}_1)\big]$
$f_s(u_s)(\underline{\mathbf{x}},t)$	$-c\exp\big(-(\alpha^2+\beta^2)t\big)\big[(3\alpha^2\beta-\beta^3)\mathrm{x}_0^2\mathrm{x}_1\sin(\alpha \mathrm{x}_0)\cos(\beta \mathrm{x}_1)$ $+(\alpha^3-3\alpha\beta^2)\mathrm{x}_0^3\cos(\alpha \mathrm{x}_0)\sin(\beta \mathrm{x}_1)+(4\alpha^2-4\beta^2)\mathrm{x}_0^2\sin(\alpha \mathrm{x}_0)\sin(\beta \mathrm{x}_1)$ $+8\alpha\beta \mathrm{x}_0\mathrm{x}_1\cos(\alpha \mathrm{x}_0)\cos(\beta \mathrm{x}_1)+(\beta^3-3\beta)\mathrm{x}_1\sin(\alpha \mathrm{x}_0)\cos(\beta \mathrm{x}_1)$ $+(3\alpha\beta^2-3\alpha)\mathrm{x}_0\cos(\alpha \mathrm{x}_0)\sin(\beta \mathrm{x}_1)+(-2\alpha^2+2\beta^2)\sin(\alpha \mathrm{x}_0)\sin(\beta \mathrm{x}_1)\big]$

	Parabolic 2d test problem "4"
$u_b(\underline{\mathbf{x}},t)$	$c\exp(-\gamma^2 t)\big[\sin(\gamma \mathrm{x}_0)+\sin(\gamma \mathrm{x}_1)\big]$ with $c,\gamma\in\mathbb{R}$ (e.g. $c=1,\gamma=1.25\pi$)
$f_b(u_b)(\underline{\mathbf{x}},t)$	0
$u_s(\underline{\mathbf{x}},t)$	$c\exp(-\gamma^2 t)\big[\gamma \mathrm{x}_0\cos(\gamma \mathrm{x}_0)+\gamma \mathrm{x}_1\cos(\gamma \mathrm{x}_1)+\sin(\gamma \mathrm{x}_0)+\sin(\gamma \mathrm{x}_1)\big]$
$f_s(u_s)(\underline{\mathbf{x}},t)$	$-c\gamma\exp(-\gamma^2 t)\big[\gamma^2\mathrm{x}_0^3\cos(\gamma \mathrm{x}_0)-\gamma^2\mathrm{x}_0^2\mathrm{x}_1\cos(\gamma \mathrm{x}_1)$ $+4\gamma\mathrm{x}_0^2\sin(\gamma \mathrm{x}_0)-4\gamma\mathrm{x}_0^2\sin(\gamma \mathrm{x}_1)+\gamma^2\mathrm{x}_1\cos(\gamma \mathrm{x}_1)$ $-3\mathrm{x}_0\cos(\gamma \mathrm{x}_0)-3\mathrm{x}_1\cos(\gamma \mathrm{x}_1)-2\gamma\sin(\gamma \mathrm{x}_0)+2\gamma\sin(\gamma \mathrm{x}_1)\big]$

Table 4.7.: Linear parabolic test problems: data functions $f_b(u_b)$ and $f_s(u_s)$, and the associated solution (u_b,u_s) of system (4.39) in two-dimensional Cartesian coordinates $\underline{\mathbf{x}}=(\mathrm{x}_0,\mathrm{x}_1)$.

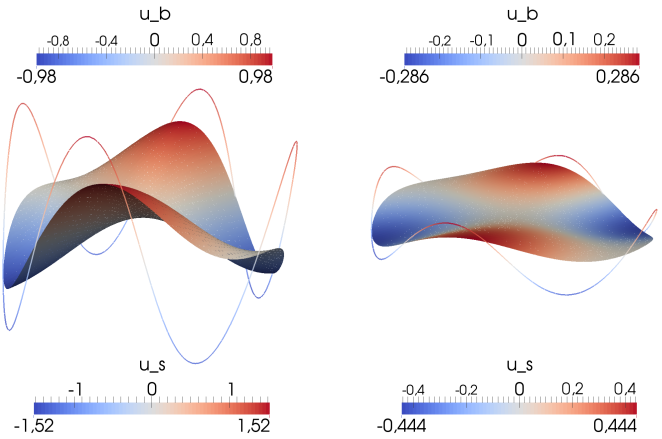
Error measures

In order to analyze the convergence of our schemes, we compute *space–time errors*

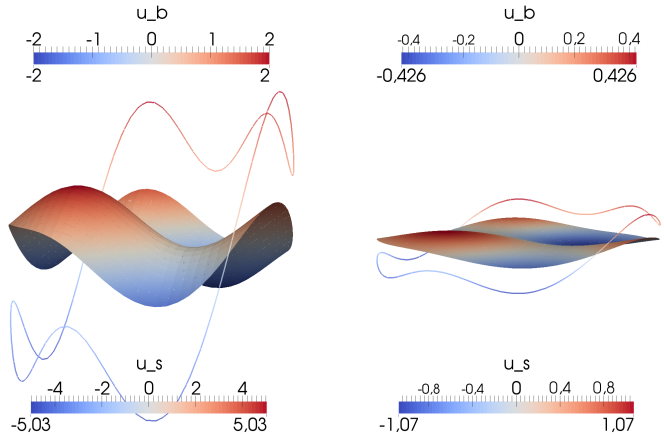
$$\mathcal{E}_{b,L^\infty,*}(h):=\sup_{t\in[0,T]}\mathcal{E}_{b,*}(h),\qquad \mathcal{E}_{b,L^2,*}(h):=\left(\int_0^T\mathcal{E}_{b,*}^2(h)\,\mathrm{d}t\right)^{\frac{1}{2}},\tag{4.40a}$$

$$\mathcal{E}_{s,L^\infty,*}(h):=\sup_{t\in[0,T]}\mathcal{E}_{s,*}(h),\qquad \mathcal{E}_{s,L^2,*}(h):=\left(\int_0^T\mathcal{E}_{s,*}^2(h)\,\mathrm{d}t\right)^{\frac{1}{2}}.\tag{4.40b}$$

Here, $*$ is a placeholder for some normed function space and $\mathcal{E}_{b,*}$ and $\mathcal{E}_{s,*}$ denote associated spatial error measures for the bulk component and for the



(a) Parabolic 2d test problem “3”.



(b) Parabolic 2d test problem “4”.

Figure 4.19.: Graph of the solution pairs (u_b, u_s) of parabolic 2d test problem “3” and parabolic 2d test problem “4” at times $t = 0$ (left) and $t = 0.1$ (right), cf. Table 4.7. The free parameters of each test problem are chosen exactly as proposed in Table 4.7.

surface component of the solution, respectively. We use the spatial error measures that are defined in Section 4.3.1. Watching the $L^\infty(0,T)$-norm in time, we hence investigate the errors $\mathcal{E}_{b,L^\infty,L^2(\Omega_h)}$, $\mathcal{E}_{s,L^\infty,L^2(\Gamma_h)}$, $\mathcal{E}_{b,L^\infty,H^1(\Omega_h)}$ and $\mathcal{E}_{s,L^\infty,H^1(\Gamma_h)}$. Analogously, watching the $L^2(0,T)$-norm in time, we investigate the errors $\mathcal{E}_{b,L^2,L^2(\Omega_h)}$, $\mathcal{E}_{s,L^2,L^2(\Gamma_h)}$ $\mathcal{E}_{b,L^2,H^1(\Omega_h)}$ and $\mathcal{E}_{s,L^2,H^1(\Gamma_h)}$.

Assuming that the space–time errors defined in equations (4.40) satisfy estimates of the form $\mathcal{E}(h) \in \mathcal{O}(h^z)$ with $z \in \mathbb{R}^{>0}$, we compute experimental

orders of convergence as defined in Section 4.3.1.

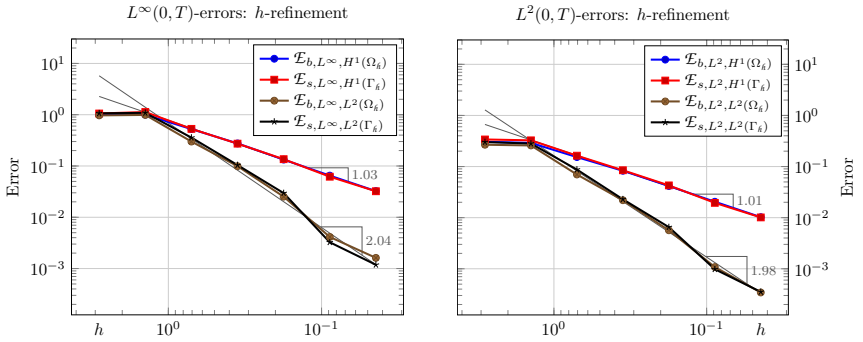
Numerical convergence study: h-refinement and polynomial degree $k = 1$

As a first numerical study, we investigate space–time errors and associated experimental orders of convergence with respect to the width h of the fundamental mesh, starting with polynomial degree $k = 1$. As with our numerical studies for linear elliptic model problems in Section 4.1.2, we employ h-refinement to do this, i.e. we perform computations while considering a sequence of fundamental meshes with $h \to 0$. Meanwhile, we keep the parameter α of the narrow band fixed. Given that the total discretization error comprises spatial errors and an error with respect to time, we need to ensure that also the latter temporal error diminishes with respect to h. Otherwise, we can not expect to observe convergence with respect to h. The backward Euler time-stepping method which is employed by Scheme 4.2.18 and by Scheme 4.2.19 is known to cause a temporal error of order 1. Theoretically, we hence need to scale the time step size τ^n with the power of h that corresponds to the largest convergence rate which we want to be able to see in one of our error measures. In practice, we will scale τ^n with some power of h that is large enough to be able to observe optimal order convergence with respect to h in the spatial L^2-norms which we are considering, i.e. convergence of order 2. As host DG formulation for the surface part of the problem, we arbitrarily choose SWIPG. Since our studies in Section 4.1.2 indicate that geometry approximation errors are not crucial in computations with polynomial degree $k = 1$, we do not use separate geometry meshes.

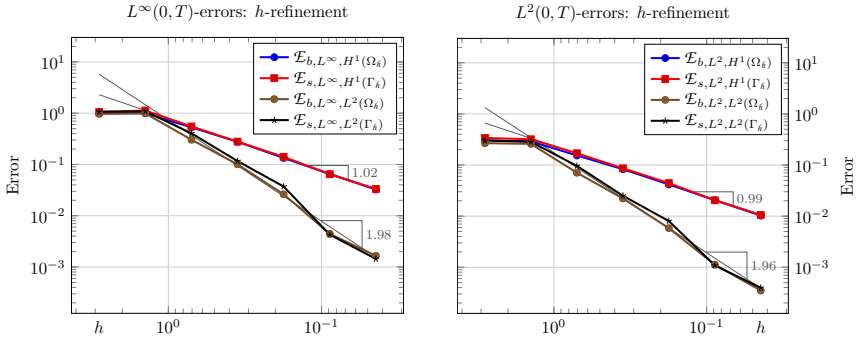
We consider Scheme 4.2.18 and Scheme 4.2.19 with either full gradient stabilization or normal penalty stabilization, always choosing the narrow band parameter $\alpha := 0.05$. Results for parabolic 2d test problem "3" are shown in Figure 4.20, results for parabolic 2d test problem "4" in Figure 4.21. Optimal order convergence with respect to all error measures which we are examining can be observed with the unstabilized scheme and with the stabilized scheme and both stabilization mechanisms. In particular, both components of the numerical solution $\left(u_{b,h}^n, u_{s,h}^n\right)$ converge to their corresponding component of the solution pair (u_b, u_s) with order 1 in the $\|\cdot\|_{H^1(\Omega_h)}$ and $\|\cdot\|_{H^1(\Gamma_h)}$ based error measures, respectively, and with order 2 in the error measures that are defined using the norms $\|\cdot\|_{L^2(\Omega_h)}$ and $\|\cdot\|_{L^2(\Gamma_h)}$. It is worth noting that a time step size τ^n which scales linearly with h is already sufficient to observe this behavior for parabolic 2d test problem "3". Obtaining those results for parabolic 2d test problem "4" requires using a time step size that scales with the expected factor of h^2.

Numerical convergence study: h-refinement and polynomial degree $k = 2$

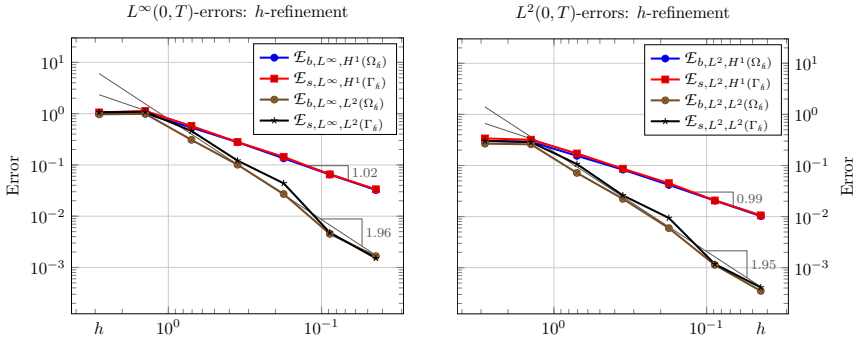
As a second numerical study, we run similar experiments for Scheme 4.2.19 using polynomial degree $k = 2$. Again, we investigate space–time errors and



(a) Unstabilized scheme, $\alpha := 0.05$.

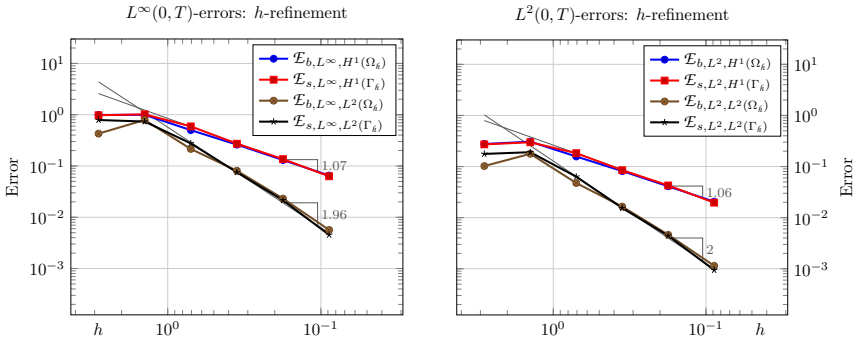


(b) Full gradient stabilization, $\alpha := 0.05$.

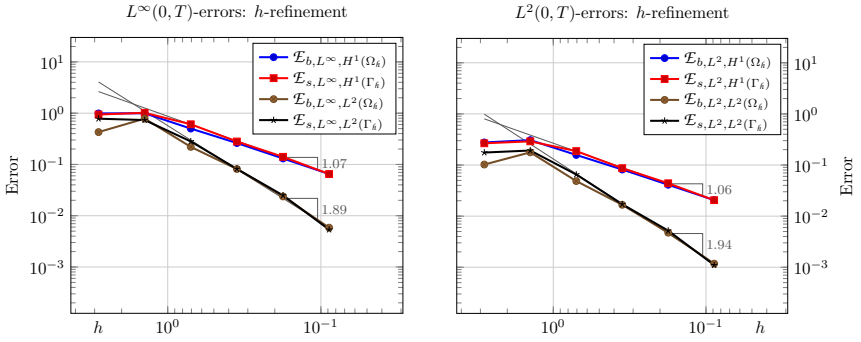


(c) Normal penalty stabilization, $\alpha := 0.05$.

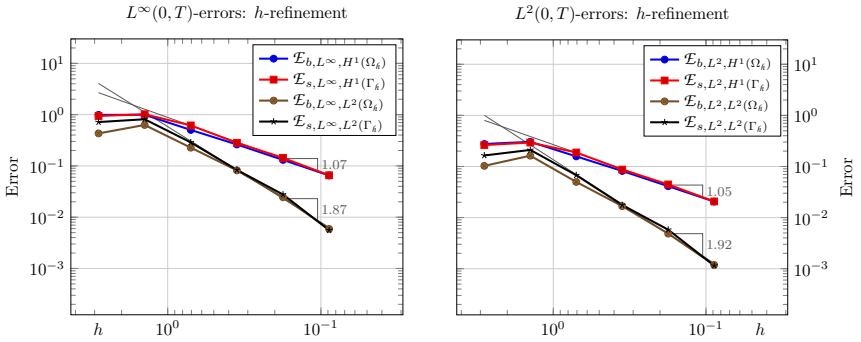
Figure 4.20.: Errors in numerical solutions of parabolic 2d test problem "3", obtained using Scheme 4.2.18 or Scheme 4.2.19 with the two considered stabilization terms, time step size $\tau^n := 0.01\,h$, $k = 1$, $\text{host}_\text{s} :=$ swipg, and no separate geometry meshes.



(a) Unstabilized scheme, $\alpha := 0.05$.

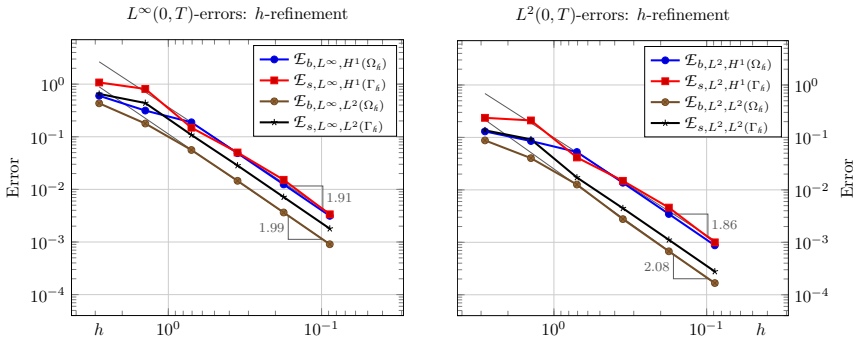


(b) Full gradient stabilization, $\alpha := 0.05$.

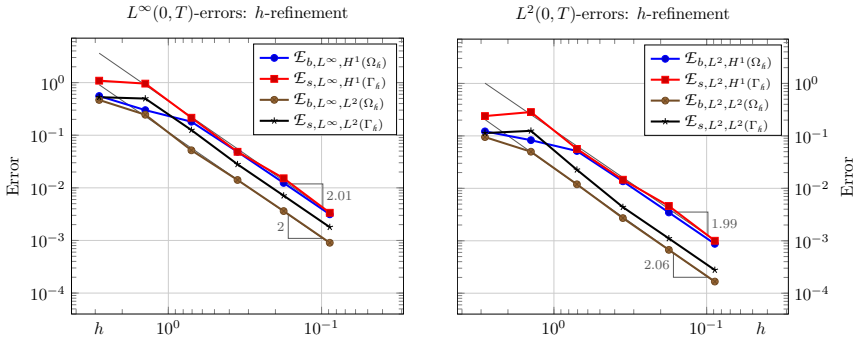


(c) Normal penalty stabilization, $\alpha := 0.05$.

Figure 4.21.: Errors in numerical solutions of parabolic 2d test problem "4", obtained using Scheme 4.2.18 or Scheme 4.2.19 with the two considered stabilization terms, time step size $\tau^n := 0.01\,h^2$, $k = 1$, $\text{host}_\text{s} :=$ swipg, and no separate geometry meshes.



(a) Full gradient stabilization, $\alpha := 0.05$.

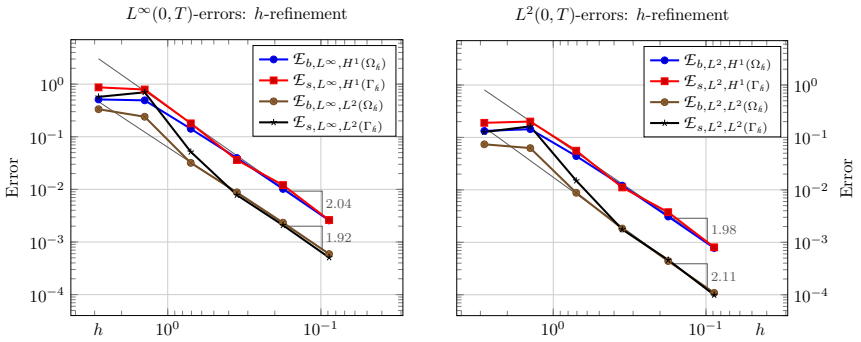


(b) Normal penalty stabilization, $\alpha := 0.05$.

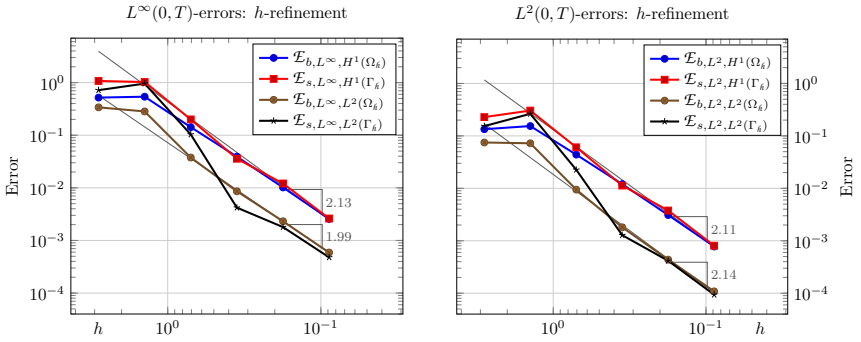
Figure 4.22.: Errors in numerical solutions of parabolic 2d test problem "4", obtained using Scheme 4.2.19 with the two considered stabilization terms, time step size $\tau^n := 0.01\,h^2$, $k = 2$, $\text{host}_\text{s} := \text{sipg}$, and separate geometry meshes with $\hbar = h/32$.

associated experimental orders of convergence with respect to the width h of the fundamental mesh, performing h-refinement ($h \to 0$, α fixed) while scaling the time step size τ^n with some power of h that is large enough to be able to see higher order convergence. In the course of this, we arbitrarily choose SIPG as host DG formulation for the surface part of the problem and we use geometry meshes with $\hbar = h/32$. As we have seen in our numerical studies for linear elliptic model problems, the latter allows for obtaining higher order schemes, when being employed together with one of the two stabilization mechanisms that are considered in this thesis.

We again choose the fixed narrow band parameter value $\alpha := 0.05$ in our study. Results for parabolic 2d test problem "4" are shown in Figure 4.22. As with the corresponding experiments for $k = 1$ above, both stabilization mechanisms produce a nearly identical outcome. Both components of the



(a) Full gradient stabilization, $\alpha := 0.05$.



(b) Normal penalty stabilization, $\alpha := 0.05$.

Figure 4.23.: Errors in numerical solutions of parabolic 2d test problem "3", obtained using Scheme 4.2.19 with the two considered stabilization terms, time step size $\tau^n := 0.01\,h^2$, $k = 2$, $\text{host}_\text{s} := \text{sipg}$, and separate geometry meshes with $\hbar = h/32$.

numerical solution $\left(u_{b,h}^n, u_{s,h}^n\right)$ converge to their corresponding component of the solution pair (u_b, u_s). The stabilized scheme particularly features optimal order convergence, i.e. convergence of order 2, in the $\|\cdot\|_{H^1(\Omega_\hbar)}$ and $\|\cdot\|_{H^1(\Gamma_\hbar)}$ based error measures, respectively. Nevertheless, it also exhibits convergence of order 2 in the error measures that are defined using the norms $\|\cdot\|_{L^2(\Omega_\hbar)}$ and $\|\cdot\|_{L^2(\Gamma_\hbar)}$, which corresponds to suboptimal order convergence.

Figure 4.23 shows results that are obtained for parabolic 2d test problem "3". They are very similar to the results which we obtain for parabolic 2d test problem "4". However, for large values of h, we observe a higher convergence rate in the error measures that are defined using spatial L^2-norms. This suggests that optimal order convergence with respect to these error measures can be recovered by scaling the time step size τ^n with some even larger power of h. Given that this is impractical, backward Euler time-stepping should

Model	Parameter	Value
GOR	$\mathcal{D}_b$	10.0
	$\mathcal{D}_s$	0.0025
	α	0.0033
	β	0.0067
	γ	0.01733
	E_c	1.0

Model	Parameter	Value
WP	$\mathcal{D}_b$	10.0
	$\mathcal{D}_s$	0.1
	k_0	0.067
	γ	1.0
	K	1.0
	δ	1.0

Table 4.8.: Values of the parameters of the simplified GOR model and values of the parameters of the WP model, which we employ in this thesis complementary to the data functions that are specified in Table 1.1.

instead be replaced by a more advanced method that achieves higher order accuracy in time.

4.3.3. Application: Nonlinear parabolic models for cell polarization

In the remainder of this section, we verify that our schemes also yield reasonable results in real-world applications and are capable of replacing simulation frameworks like the one which has been presented in Section 1.4. We apply the fully discrete schemes from Section 4.2.5 to two time-dependent nonlinear models from cell biology that can be formulated within the class of parabolic model problems (4.1).

Models

In Section 1.4.1, we introduced the simplified GOR model and the bulk–surface formulation of the WP model, which can both be used to study basic features of an intracellular redistribution process known as cell polarization, cf. Section 1.4. Each model can be formulated using equations (4.1), together with the corresponding data functions specified in Table 1.1. It should be noted that $f_{b,s}(u_b, u_s)$ and $f_{s,b}(u_b, u_s)$ are nonlinear terms in the solution variable u_s. This nonlinearity arises in boundary condition (4.1b) and in one of the reaction terms of surface PDE (4.1c).

Both models deal with cell polarization at the microscopic level, i.e. at the level of single cells. Correspondingly, their conserved quantities are masses of intracellular proteins and lipids. Given that data functions are employed that satisfy $f_b \equiv 0$, $f_s \equiv 0$ and $f_{b,s} = -f_{s,b}$, the global conservation properties which have been discussed in Section 4.1 induce that the system's total mass $m(t)$ stays constant in time, see property (4.4). Especially for the WP model, this aspect of mass conservation is a fundamental hypothesis and an essential ingredient of the polarization mechanism (Mori et al., 2008).

Common simulation parameters

In each simulation, we use the model parameters that are specified in Table 4.8. If not mentioned separately, we define the data function k_{stimulus} in Table 1.1 by setting

$$k_{\text{stimulus}}(\underline{\mathbf{x}}, t) := 0.$$

The remaining simulation parameters and related choices are given as follows, unless otherwise stated.

We perform simulations on different geometries (Ω, Γ) in $\mathbb{R}^2$ that are described choosing a square level set domain $\Omega_\Phi \subset \mathbb{R}^2$ and some suitable level set function Φ. The latter is either defined by an analytical expression or it is obtained as the result of a level set based image segmentation algorithm which we apply to imaging data. To perform the extension process which is described in Section 4.2.1, we use a narrow band Ω_δ with equal parameters $\alpha_{\text{in}} = \alpha_{\text{out}} =: \alpha \in \mathbb{R}^{>0}$, and we choose the corresponding constant extension to Ω_δ as extension $\mathcal{D}_s^{\text{ext}}$ of the constant diffusivities $\mathcal{D}_s$ from Table 4.8.

For the UDG discretization, we employ a Cartesian fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ and a Cartesian geometry mesh $\mathcal{T}_\hbar(\Omega_\Phi)$ with a given mesh width h and $\hbar$, respectively. To obtain $\mathcal{T}_h(\Omega_\Phi)$, we start with one entity which corresponds to Ω_Φ and perform a certain number $\mathfrak{r} \in \mathbb{N}$ of uniform mesh refinements. We either use the choice $\mathcal{T}_\hbar(\Omega_\Phi) := \mathcal{T}_h(\Omega_\Phi)$, i.e. no separate geometry mesh, or a mesh $\mathcal{T}_\hbar(\Omega_\Phi)$ which results from a fixed number of additional uniform refinements for each of the entities in $\mathcal{T}_h(\Omega_\Phi)$.

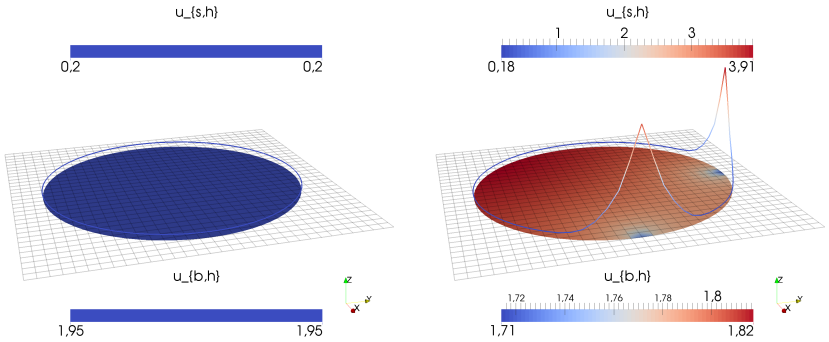
On the fundamental mesh, we choose discrete spaces $V_{s,h}(\Omega_{\delta,\hbar})$ and $V_{b,h}(\Omega_\hbar)$ which locally (i.e. on each cut cell K) resemble $\mathbb{P}(K) := P_k(K)$, the space of polynomial functions of total degree less than or equal to some $k \in \mathbb{N}$ over the domain K. Particularly choosing the polynomial degree $k = 1$ for both discrete spaces, we construct $V_{s,h}(\Omega_{\delta,\hbar})$ and $V_{b,h}(\Omega_\hbar)$ using monomial basis functions (i.e. the set $\{1, \mathrm{x}_0, \mathrm{x}_1\}$ for $k = 1$) on the reference element of the fundamental mesh elements.

As host DG formulations, we use SIPG for the bulk part of the problem and either SIPG or SWIPG for its surface part, specifically choosing $\text{host}_\text{b} := \text{sipg}$ with $\gamma_b := 10.0$, and either $\text{host}_\text{s} := \text{sipg}$ with $\gamma_s := 2.0$ or $\text{host}_\text{s} := \text{swipg}$ with $\gamma_s := 22.5$.

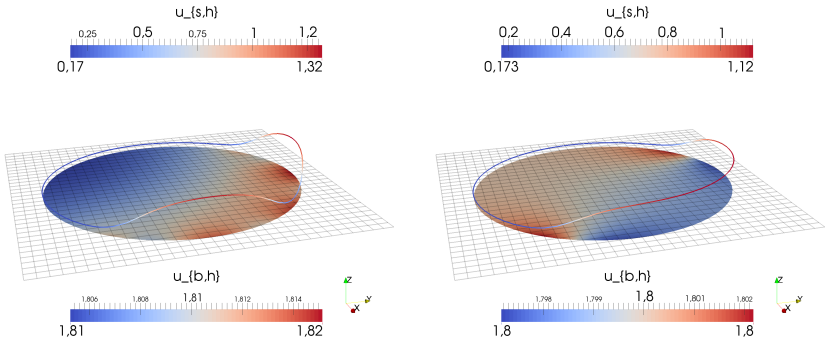
In each simulation run, we perform several time steps with a uniform step size $\tau^n \in \mathbb{R}$. Due to the nonlinearity of the models which we are considering, each time step results in a system of algebraic equations that are nonlinear. To solve this nonlinear system, we employ Newton's method.

Simulation: WP model on a circular cell

In our first simulation, we consider the WP model on a cell which is represented by some circular geometry (Ω, Γ) in $\mathbb{R}^2$. The latter shall comprise the open disk of center $(0, 0)^{\text{tr}}$ and radius 4.5, and the corresponding circle making up its



(a) $t^n = t^0 = 0$ (discrete initial values). (b) $t^n = t^{20} = 5$.



(c) $t^n = t^{100} = 25$. (d) $t^n = t^{400} = 100 =: T$.

Figure 4.24.: Numerical solution $\left(u_{b,h}^n, u_{s,h}^n\right)$ of the WP model on a circular cell at different discrete times t^n, obtained using Scheme 4.2.19 with full gradient stabilization, time step size $\tau^n := 0.25$, $k = 1$, $\text{host}_\text{s} := \text{sipg}$, $\mathfrak{r} = 5$ ($h \approx 4.86 \cdot 10^{-1}$), $\alpha := 0.05$, and no separate geometry meshes. We visualize function values of $u_{b,h}^n$ by colors (according to the color bar that is shown) and show the graph of $u_{s,h}^n$ on $\Gamma_{\hbar}$ (employing a similar color scheme). The corresponding fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ with $\Omega_\Phi := (-5.5, 5.5)^2$ is depicted in black.

boundary. Both can be described using the level set domain $\Omega_\Phi := (-5.5, 5.5)^2$ and the level set function Φ that is defined by $\Phi(\underline{\mathbf{x}}) := |\underline{\mathbf{x}}| - 4.5$.

We start with initial values $u_b(\cdot, 0) \equiv 1.95$ and $u_s(\cdot, 0) \equiv 0.2$, which represent a stable, homogeneous steady state of the model. In the first period of the simulation, we excite the system in order to enter an unstable, transient steady state. We apply two stimuli by means of the data function k_{stimulus} in

Table 1.1, particularly setting

$$k_{\text{stimulus}}(\underline{\mathbf{x}}, t) := \begin{cases} 2.4 & \text{if } t < 20 \text{ and } \left|\underline{\mathbf{x}} - (0, 4.5)^{\text{tr}}\right| < 0.2, \\ 2.0 & \text{if } t < 20 \text{ and } \left|\underline{\mathbf{x}} - (4.5, 0)^{\text{tr}}\right| < 0.2, \\ 0 & \text{otherwise.} \end{cases}$$

After the stimuli are turned off at time $t = 20$, two waves emerge from the two stimulus sites. These waves travel along the cell membrane Γ and eventually merge. Mass conservation finally pins the system into a stable, polarized steady state (cf. Giese, Eigel, Westerheide, Engwer and Klipp, 2015a).

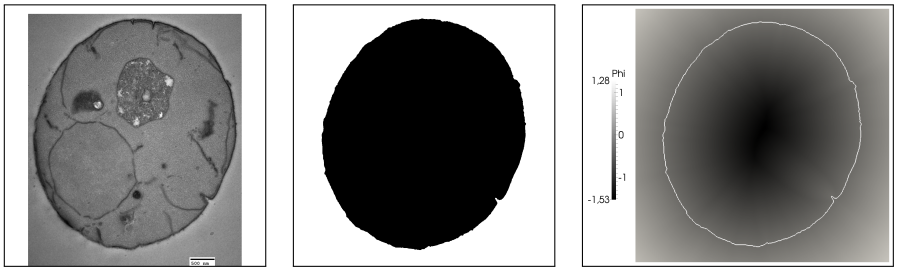
The above behavior is perfectly reproduced by numerical solutions that are obtained using our globally conservative Scheme 4.2.18 or its stabilized analogue, i.e. Scheme 4.2.19. Figure 4.24 shows an example numerical solution that is obtained using Scheme 4.2.19 with full gradient stabilization. The choices $\text{host}_\text{s} := \text{sipg}$, $\alpha := 0.05$ and the choice of the stabilization mechanism were made arbitrarily in Figure 4.24. Without showing concrete evidence here in this thesis, we would like to point out that essentially the same results are obtained without stabilization or with normal penalty stabilization, with $\text{host}_\text{s} := \text{swipg}$, and with other values of α.

Simulation: Simplified GOR model on real microscopy data

In our second simulation, we consider the simplified GOR model on a cell geometry (Ω, Γ) in $\mathbb{R}^2$ which is given by the microscopy image shown in Figure 4.25a. To extract the cell Ω and its membrane Γ from this image, we employ an image segmentation algorithm based on the level set framework, such as the one that is introduced in (Chan and Vese, 1999, 2001). In the course of this, we interpret the image as a function of gray values on the level set domain Ω_Φ, specifically choosing $\Omega_\Phi := (-2, 2)^2$. As a result, we obtain the level set function Φ which is depicted in Figure 4.25c and the associated geometry (Ω, Γ) shown in Figure 4.25b.

We start with initial values $u_b(\cdot, 0)$ and $u_s(\cdot, 0)$ that are fields of log-normally distributed random numbers (cf. Figure 4.27a). As a feature of the Turing-type polarization mechanism of the GOR model and of its simplified variant which we are considering here, the system's homogeneous steady state is unstable with respect to minute spatial perturbations (cf. Giese, Eigel, Westerheide, Engwer and Klipp, 2015a). Given that our random initial values can be interpreted as such, the system runs into a spatially inhomogeneous, polarized steady state instead of running into the homogeneous steady state.

Again, the above behavior is perfectly reproduced by numerical solutions that are obtained using our globally conservative Scheme 4.2.18 or its stabilized analogue, i.e. Scheme 4.2.19. Figure 4.27 shows an example numerical solution that is obtained using the unstabilized Scheme 4.2.18. As with the above simulation for the WP model on a circular cell, the choices $\text{host}_\text{s} := \text{swipg}$, $\alpha := 0.50$ and the choice of using no stabilization were made arbitrarily in



(a) Microscopy image. (b) Extracted geometry. (c) Level set function Φ and its level set $\Gamma_0 = \Gamma$ (depicted in white).

Figure 4.25.: Part (a) shows a transmission electron microscopy image of a two-dimensional slice of a yeast cell. To simplify working with the microscopy image, it has been embedded in a square frame of reference. Image (b) shows the extracted geometry (Ω, Γ) which we use in our computations. The associated level set function is shown in (c), together with its zero level set, which represents Γ. The microscopy image depicted in (a) is reprinted by permission from Genetics Society of America: Genetics (Rainey et al., 2010, Figure S6), © 2010.

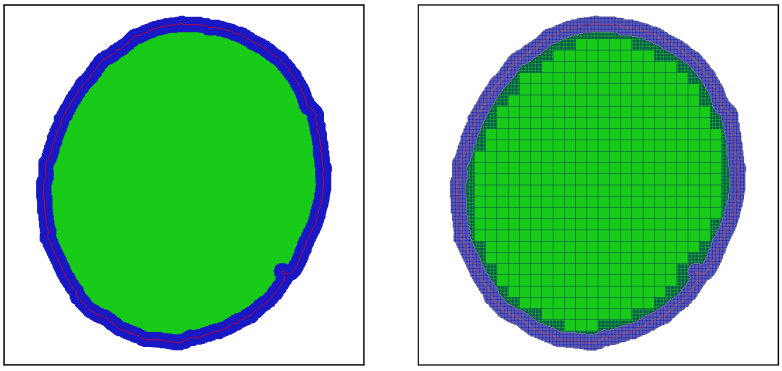
Figure 4.27. We do not show concrete evidence here in this thesis, but would like to point out that essentially the same results are obtained with full gradient stabilization, with $\text{host}_\text{s} := \text{sipg}$, and with other values of α.

Please note that we employ a relatively coarse fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ in our simulation, while we use a geometry mesh $\mathcal{T}_{h}(\Omega_\Phi)$ that is fine enough to obtain an accurate geometry reconstruction. Figure 4.26 illustrates the discrete reconstruction $(\Omega_h, \Gamma_h, \Omega_{\delta,h})$ of the geometry and of the associated narrow band Ω_δ which we are using.

Numerical study: Evolution of masses

Working with the latter simulation for the simplified GOR model on real microscopy data, we now investigate numerically how well conservation properties are reflected by the discrete systems which result from applying the fully discrete schemes from Section 4.2.5. In particular, we compare Scheme 4.2.17 and Scheme 4.2.18 while performing α-refinement. Given a fundamental mesh with a fixed h, we consider a sequence of values $\alpha \to 0$ and observe the evolution of the amounts of the fully discrete system's quantities. These are given by $m^n_{b,h} := \int_{\Omega_h} u^n_{b,h} \, \mathrm{d}x$, $m^n_{s,h} := \int_{\Gamma_h} u^n_{s,h}|_{\Gamma_h} \, \mathrm{d}\sigma$ and $m^n_h = m^n_{b,h} + m^n_{s,h}$, and are masses in the model which we are considering.

Results for different values of α are depicted in Figure 4.28 and Figure 4.29. These results reproduce what we have discussed and shown from a theoretical

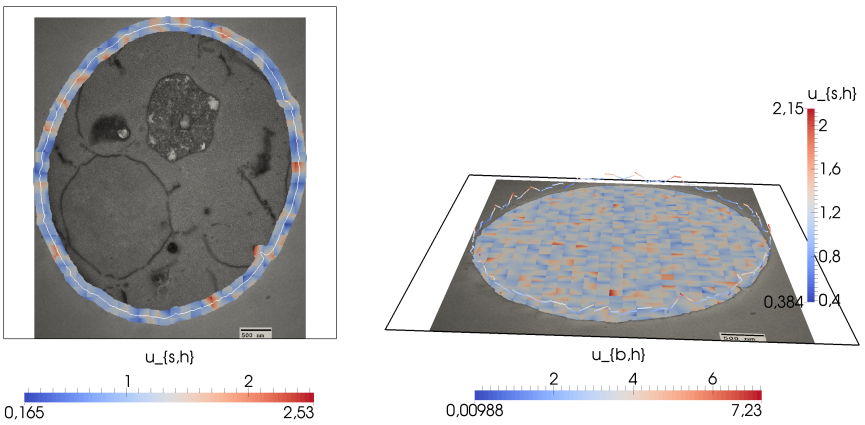


(a) Discrete reconstruction. (b) Local triangulations.

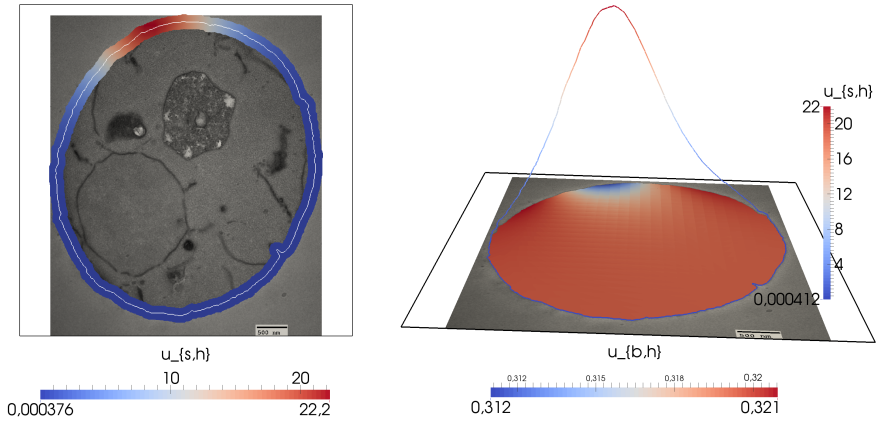
Figure 4.26.: Part (a) shows the discrete reconstruction $(\Omega_\hbar, \Gamma_\hbar, \Omega_{\delta,\hbar})$ of the geometry (Ω, Γ) in Figure 4.25b and of the associated narrow band Ω_δ with $\alpha := 0.50$ which we use in our computations. $\Omega_\hbar$, $\Gamma_\hbar$ and $\Omega_{\delta,\hbar}$ are depicted in green, red and blue, respectively. Part (b) illustrates the corresponding local triangulations $\mathcal{T}_\hbar(K)$ that are solely used for numerical integration over the cut cells $K \in \mathcal{T}_h(\Omega_\hbar) \cup \mathcal{T}_h(\Omega_{\delta,\hbar})$ and their faces $E \subset \partial K$, as discussed at the end of Section 4.2.2. Large cells correspond to cut cells for which no local triangulation is generated since they are cut by neither $\Gamma_\hbar$ nor $\Omega_{\delta,\hbar}$ and hence match full elements in the Cartesian fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$. Information on the mesh widths h and $\hbar$ is given in Figure 4.27.

point of view in Section 4.2.3 and in Theorem 4.2.20. While Scheme 4.2.18 is globally conservative, independent of the choice of α, global mass conservation is achieved only approximatively using Scheme 4.2.17. This approximation is the more accurate, the smaller we choose α. We note that the evolution in Figure 4.28e almost equals the one in Figure 4.28f that has been obtained using Scheme 4.2.18. Still, we obtain a total mass with a relative error of magnitude 10^{-2}, whereas a relative error near machine precision is achieved using Scheme 4.2.18. In this context, it should be mentioned that parts of the code which has been used to assemble the system of linear equations in each step of Newton's method uses numerical differentiation. Therefore, we can in no way expect to achieve full machine precision.

We note that these differences between both schemes also affect the quality of the discrete solutions which they produce. In case of the globally conservative Scheme 4.2.18, different values of α yield a qualitatively equal polarization behavior, see Figure 4.28. The discrete system reaches a polarized steady state at $t^n \approx 600$ for each α. On the contrary, qualitatively different polarization behaviors are obtained using Scheme 4.2.17. As shown in Figure 4.28, the

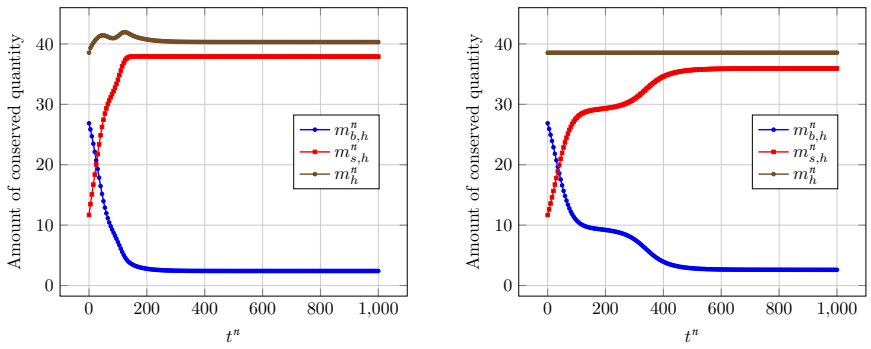


(a) Numerical solution at discrete time $t^n = t^0 = 0$ (discrete initial values).



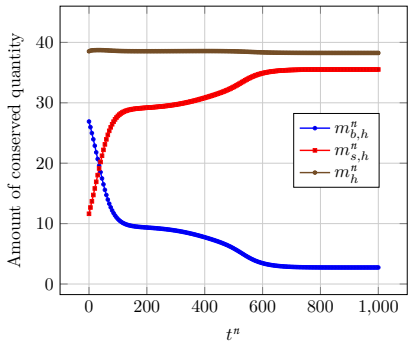
(b) Numerical solution at discrete time $t^n = t^{400} = 1000 =: T$.

Figure 4.27.: Numerical solution $\left(u^n_{b,h}, u^n_{s,h}\right)$ of the simplified GOR model on the geometry that is shown in Figure 4.25b, obtained using Scheme 4.2.18, time step size $\tau^n := 2.5$, $k = 1$, $\mathrm{host}_s := \mathrm{swipg}$, $\mathfrak{r} = 5$ ($h \approx 1.77 \cdot 10^{-1}$), $\alpha := 0.50$, and a separate geometry mesh with $\hbar = h/8$. Left: Function values of $u^n_{s,h}$ in the discrete narrow band $\Omega_{\delta,\hbar}$, visualized by colors according to the color bar that is shown. Right: Function values of $u^n_{b,h}$ and graph of $u^n_{s,h}$ on $\Gamma_\hbar$, visualized using a similar color scheme. Please see Figure 4.25 for copyright information.

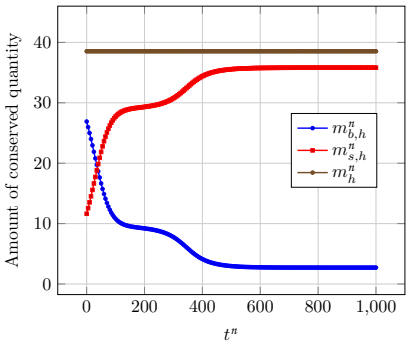


(a) Scheme 4.2.17, $\alpha := 1.00$.

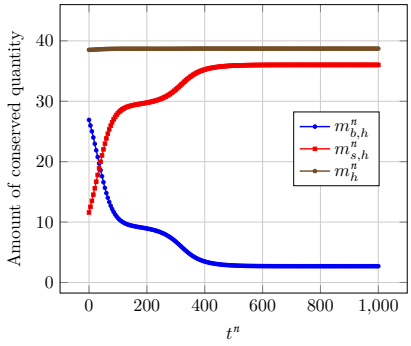
(b) Scheme 4.2.18, $\alpha := 1.00$.



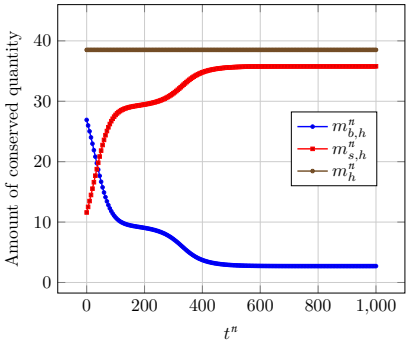
(c) Scheme 4.2.17, $\alpha := 0.50$.



(d) Scheme 4.2.18, $\alpha := 0.50$.

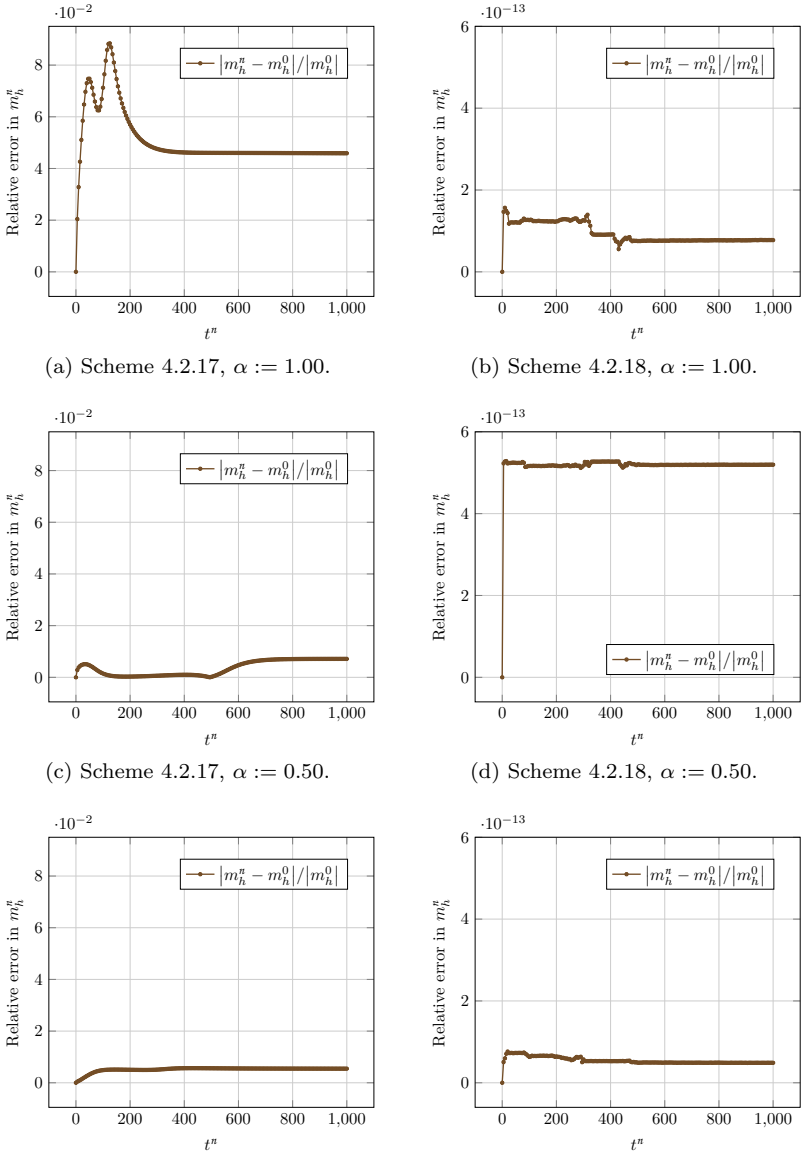


(e) Scheme 4.2.17, $\alpha := 0.05$.



(f) Scheme 4.2.18, $\alpha := 0.05$.

Figure 4.28.: Comparison of the evolutions of masses that are obtained using different values of α and either Scheme 4.2.17 (left), which conserves mass only approximately, or the globally conservative Scheme 4.2.18 (right). The simulation that is performed is the same as the one in Figure 4.27, with identical scheme parameters except for α.



(a) Scheme 4.2.17, $\alpha := 1.00$.

(b) Scheme 4.2.18, $\alpha := 1.00$.

(c) Scheme 4.2.17, $\alpha := 0.50$.

(d) Scheme 4.2.18, $\alpha := 0.50$.

(e) Scheme 4.2.17, $\alpha := 0.05$.

(f) Scheme 4.2.18, $\alpha := 0.05$.

Figure 4.29.: Comparison of the evolutions of the relative error in total mass m_h^n that are obtained using different values of α and either Scheme 4.2.17 (left), which conserves mass only approximately, or the globally conservative Scheme 4.2.18 (right). The simulation that is performed is the same as the one in Figure 4.27, with identical scheme parameters except for α.

discrete system reaches a polarized steady state at $t^n \approx 400$ in (a), at $t^n \approx 800$ in (c), and at $t^n \approx 600$ in (e).

Our study shows that recovering a model's underlying conservation properties can be crucial in order to obtain reliable simulation results. If conservation properties are known to be an essential ingredient of the given model, we therefore recommend to avoid schemes like Scheme 4.2.17 and to use globally conservative schemes instead, such as Scheme 4.2.18 or its stabilized analogue, i.e. Scheme 4.2.19.

4.4. Discussion

In this chapter, we constructed and analyzed a new type of UDG schemes for two example classes of bulk–surface models that comprise continuity equations on static geometries. These schemes are specially designed for mapping properties of continuity equations, and for geometries of arbitrarily complex shape that are represented using the level set framework. They particularly build upon transferring the UDG method to PDEs on hypersurfaces.

As a first step, we introduced an extension process which exploits the level set description of the geometry to obtain bulk extensions of the models' surface PDEs. Our particular approach yields extensions that allow for discretizations which still recover discrete analogues to the original conservation properties that are embedded in the considered type of surface continuity equations. The properties of those extensions, e.g. the fact that they are based on narrow bands which serve as surface extension domains, suggest numerical treatment using geometrically unfitted methods for bulk PDEs, such as the UDG method.

As a second step, we applied the UDG method to perform spatial discretization of the systems of coupled bulk PDEs resulting from the extension process. The method generally provides all benefits of DG approaches. However, in our particular setting, it inherently results in basic semidiscretizations and schemes which suffer from the nature of all extension-based discretization methods for surface PDEs. We developed a strategy which deals with this deficiency by replacing selected terms in the basic semidiscretizations and schemes by suitable analogues that are known from sharp interface FEMs for surface PDEs, applying a proper scaling as well. In order to increase numerical robustness of the resulting variants, we introduced mechanisms that stabilize the surface part of the solution by penalizing variations of its normal component. We particularly considered full gradient stabilization and normal penalty stabilization.

To obtain fully discrete schemes for time-dependent bulk–surface models, we finally performed discretization in time by employing the backward Euler method.

By investigating our approaches theoretically, we proved that they allow for constructing globally conservative schemes. Similar to other DG approaches, those schemes additionally recover discrete analogues to local conservation properties that are embedded in models comprising continuity equations. Our

theoretical investigations demonstrated that the latter also holds true for the models' surface PDEs. The hybrid nature of our way of dealing with surface PDEs hence successfully remedies the shortcomings of pure extension-based methods regarding the recovery of conservation properties. At the same time, our approach stays as easily implementable[2] as pure extension-based methods, instead of posing the implementational difficulties which DG analogues to genuine sharp interface FEMs would pose. Note that implementing such a DG analogue would require evaluating fluxes over codimension 2 faces of elements in the cut cell mesh $\mathcal{T}_h(\Omega_f)$.

Numerical investigation of our schemes indicated good convergence properties, as well as well-posedness and well-conditionedness of the corresponding systems of algebraic equations that need to be solved. In particular, we could observe optimal order convergence for $k = 1$ in the usual H^1-norms and L^2-norms. Furthermore, we have shown that mechanisms which stabilize the surface part of the solution by penalizing variations of its normal component indeed increase numerical robustness. The two considered stabilization mechanisms performed equally well in terms of errors and condition numbers. Employing separate geometry meshes to obtain a geometry description which is more accurate helped improving results that were obtained using coarse fundamental meshes. Moreover, we demonstrated that employing host DG formulations that are designed to handle equations with heterogeneous diffusivity, such as the SWIPG formulation, helps improving the condition of the resulting systems of algebraic equations. Last but not least, we were able to see higher order convergence for $k = 2$ by combining stabilization of the surface part of the solution with a geometry description which is more accurate.

By applying our schemes to nonlinear parabolic models for cell polarization, we verified that they yield reasonable results in real-world applications and that they are capable of replacing simulation frameworks like the one which has been presented in Section 1.4. Those schemes that are globally conservative produced reliable results that can help answering questions from cell biology. A similar scheme which recovers conservation properties only approximatively delivered results that are less reliable. We could show that the quality of these results depends on the particular choice of a discretization parameter which controls how well the scheme reflects conservation properties.

4.4.1. Future perspectives

More general classes of bulk–surface models on static geometries

Since the extension process from Section 4.2.1 is not limited to the diffusive surface flux which we have chosen in this work, the approaches that are presented in Section 4.2 can be applied to more general classes of bulk–surface models on static geometries, particularly those which employ a different surface flux $\mathbf{q_s}$. For instance, they can be applied to problems that incorporate conservative

[2] Provided that an implementation of the UDG method for bulk PDEs is available.

transport in terms of advective bulk/surface fluxes that are intrinsic to Ω and Γ, respectively. Naturally, for every particular choice of bulk/surface fluxes, host DG formulations need to be employed that are adapted to the specific needs of the resulting bulk PDEs.

Higher order time-stepping

In this work, we employ the backward Euler method to give a basic example of how discretization in time can be performed and to ease discussing conservation properties of fully discrete solutions. While the numerical studies in Section 4.3.2 show that backward Euler time-stepping allows for schemes with good convergence properties, it is advisable, for reasons of efficiency, to perform time-stepping using more advanced methods which achieve higher order accuracy in time. Such methods are, for example, the second order time integrator that is known as the implicit trapezoidal rule (and also called Crank–Nicolson method by some authors), the second order time integrator known as the implicit midpoint method, and other higher order members of the family of Runge–Kutta methods (see e.g. Hairer et al., 2006, Section II.1.1).

The resulting schemes still recover fully discrete analogues to the model's underlying global conservation properties. Those analogues are similar to the ones presented in Theorem 4.2.20. The same applies to the model's underlying local conservation properties.

Higher order geometry approximation

In our schemes, we employ piecewise (multi-)linear level set functions on geometry meshes to obtain discrete reconstructions Γ_h, Ω_h and $\Omega_{\delta,h}$ (see Section 4.2.2), and subsequently consider piecewise linear approximations of those objects to perform numerical integration (see Section 4.2.2). In the numerical convergence studies for polynomial degree $k = 2$ that were performed in Section 4.3, we increased the accuracy of the geometry approximation by constructing a separate geometry mesh which is finer than the fundamental mesh, starting from the fundamental mesh and performing a fixed number of additional uniform mesh refinements. Increasing the geometrical accuracy in this way showed to be one of two essential ingredients for observing higher order convergence.

It should be mentioned, however, that the number of refinements had to be chosen large enough in each study to observe higher order convergence throughout the parameter regime which we were considering. This is due to the fact that the approximation quality of piecewise linear approximations is only second order accurate (cf. Lehrenfeld, 2016; Engwer and Nüßing, 2017). In computations with discrete spaces of polynomial degree $k > 1$, this low order geometry approximation error asymptotically dominates the overall error with respect to the width h of the fundamental mesh.

This drawback can only be removed in an efficient manner by applying approaches for higher order geometry approximation in geometrically unfitted FEMs. Such an approach is introduced, e.g., in Lehrenfeld (2016). It is also based on piecewise linear geometry approximations and can hence be incorporated into our schemes with manageable effort. It effectively results in an additional geometry mapping which needs to be taken into account similar to the mappings in fitted isoparametric FEMs.

Nevertheless, we would like to emphasize that the concept which we are using in this thesis to benefit from higher order convergence for polynomial degree $k > 1$, i.e. employing piecewise linear geometry approximations based on separate geometry meshes of sufficient precision, is sufficient if an exact representation of the geometry is not available. This is the case, for instance, if the geometry is given via imaging data. Here, the resolution of an image determines the available level of geometric detail.

Alternative stabilization mechanisms for UDG discretizations of bulk PDEs

In Section 4.3, we investigated the convergence and conditioning properties of the schemes which have been introduced in Section 4.2. However, we did not analyze to what extent the condition of the resulting systems of algebraic equations depends on the relative position of the bulk domain Ω in the level set domain Ω_Φ. Each relative position leads to a different cut cell configuration. In cut cell methods, particularly those configurations which bring along small cut cells are known to potentially be troublesome regarding the well-posedness and well-conditionedness of the corresponding systems of algebraic equations, cf. Section 1.3.2.

The implementation of the UDG method which we are using employs the basis function rescaling technique that is analyzed in Engwer (2009, Section 5.2). Each basis function is rescaled according to the bounding box of its associated cut cell so that all cut cells contribute to the algebraic equations in a manner that is largely independent of their size. In addition to the numerical studies that are performed in Section 4.3, it should be investigated whether this technique is already sufficient to ensure that no ill-conditioned or nearly singular system of algebraic equations arises from any particular cut cell configuration.

Potential deficiencies of the current basis function rescaling technique should be curable by using cell merging techniques, which associate very small cut cells to a neighboring mesh element or to some cut cell that has a sufficiently large intersection with the considered bulk domain (cf. Johansson and Larson, 2013; Heimann et al., 2013; Müller et al., 2017; Kummer, 2017; Kummer et al., 2018), or by adding ghost penalties for UDG discretizations to the formulation. The latter has only just been investigated in Massing and Gürkan (2018).

Generating extended initial values and data functions for surface PDEs

The approaches presented in this chapter, particularly the extension process in Section 4.2.1, result in fully discrete schemes for the class of parabolic model problems (4.1) which employ extended initial values $u_{s,h}^0 \in V_{s,h}(\Omega_{\delta,\mathfrak{h}})$ and an extended surface diffusivity tensor $\mathcal{D}_s^{\text{ext}}$ living in the discrete narrow band $\Omega_{\delta,\mathfrak{h}}$. Schemes that are based on our approaches hence require generating suitable extensions from initial values or from data functions of the surface part of a problem, if those entities are defined only on the discrete reconstruction $\Gamma_{\mathfrak{h}}$ of the given hypersurface.

Motivated by approaches which construct normally constant extensions of quantities (see e.g. Xu and Zhao, 2003, Section 2), we propose to compute required extensions by performing a sharp interface L^2-projection which is stabilized using the normal penalty stabilization mechanism from Section 4.2.4, i.e., we propose the following UDG scheme.

Scheme 4.4.1 (Stabilized sharp interface L^2-projection)**.** *Given some data $f \in L^2(\Gamma_{\mathfrak{h}})$, we seek a discrete function $u_{s,h} \in V_{s,h}(\Omega_{\delta,\mathfrak{h}})$, such that*

$$\int_{\Gamma_{\mathfrak{h}}} \left(u_{s,h}\big|_{\Gamma_{\mathfrak{h}}} - f \right) \varphi_{s,h}\big|_{\Gamma_{\mathfrak{h}}} \, \mathrm{d}\sigma + \frac{1}{\delta} j_s^{\text{np}}(u_{s,h}, \varphi_{s,h}) = 0 \qquad \forall \varphi_{s,h} \in V_{s,h}(\Omega_{\delta,\mathfrak{h}}).$$

Here, $V_{s,h}(\Omega_{\delta,\mathfrak{h}})$ and $j_s^{\text{np}}(u_{s,h}, \varphi_{s,h})$ denote the discrete spaces that are defined in Section 4.2.2 and the stabilization term that is defined in equation (4.33)*, respectively. Regarding $V_{s,h}(\Omega_{\delta,\mathfrak{h}})$, we choose some polynomial degree $k \in \mathbb{N}$ suitable for representing f in a discrete sense.*

Please note that Scheme 4.4.1 is globally conservative in the sense that a solution $u_{s,h}$ satisfies $\int_{\Gamma_{\mathfrak{h}}} u_{s,h}|_{\Gamma_{\mathfrak{h}}} \, \mathrm{d}\sigma = \int_{\Gamma_{\mathfrak{h}}} f \, \mathrm{d}\sigma$. It is hence a consistent tool for the type of UDG schemes which has been developed in this chapter. It has not yet been thoroughly investigated, though.

5. Toward unfitted DG schemes for coupled bulk–surface PDEs on evolving geometries

In this chapter, we show how the approaches from Chapter 4 can be extended to obtain UDG schemes for bulk–surface models that comprise continuity equations on evolving geometries. As a first step toward such schemes, we address restricting the overall problem to the static geometry case and additional transport problems by presenting an operator splitting approach for evolving geometry problems. Combining this approach with suitable numerical schemes for the resulting subproblems yields overall schemes which recover discrete analogues to conservation properties that are embedded in the considered type of continuity equations.

As a second step toward UDG schemes for evolving geometry bulk–surface models, we construct and analyze a novel UDG scheme for an essential type of continuity equations on evolving hypersurfaces, targeting hypersurfaces that are represented using the level set framework. This scheme can be employed to treat the surface equations in those transport problems which need to be dealt with when applying the operator splitting approach mentioned above. Although surface equations are rewritten by exploiting the level set description of the geometry in a way similar to the extension process in Section 4.2.1, the scheme is specially designed for recovering discrete analogues to the surface equations' original conservation properties. We show promising first numerical results for a selected geometrical special case. These results are complemented by discussing limitations of the proposed formulation of the scheme, which need to be overcome to obtain a scheme which is generally usable.

We begin in Section 5.1 by deriving a class of bulk–surface models that will serve as an example in this chapter, and by discussing its associated conservation properties. The operator splitting approach for evolving geometry problems is introduced in Section 5.2. It is illustrated by constructing one first order operator splitting method and one second order operator splitting method for our example class of model problems. We examine how these methods affect conservation properties and we discuss numerical schemes that are suitable to treat the resulting subproblems. Section 5.3 focusses on our new UDG scheme which can be employed to handle surface equations in those subproblems which deal with material transport. Step by step, we derive an approximate reformulation of surface equations as sequences of stationary bulk

advection problems, and we describe the actual UDG discretization which is applied on top. Subsequently, we investigate the resulting scheme from a theoretical point of view, mainly considering a one-dimensional setting, and we illustrate its practical performance by first numerical results. In Section 5.4, we summarize our findings and discuss future perspectives.

The material in Section 5.3 is joint work with Christian Engwer (WWU Münster) and Thomas Ranner (University of Leeds). To a large extent, it has been published in the *Proceedings of the conference ALGORITMY 2016*, see Engwer, Ranner and Westerheide (2016).

5.1. A class of evolving geometry model problems

To present and investigate the ideas that are developed in this chapter, we consider the class of bulk–surface models from Section 1.2 on some given evolving geometry. This geometry is assumed to be represented by means of the level set framework that has been described in Section 3.3. In this light, let us briefly recall the entities of the setting from Section 1.2 and the entities of the level set framework. Both will be used throughout this chapter.

Let $\Omega(t)$ be an evolving bulk domain in $\mathbb{R}^d$ which is bounded by an evolving, potentially complex-shaped, smooth $(d-1)$-dimensional hypersurface $\Gamma(t)$. Given an observation period $[0,T]$, let both be represented by a time-dependent level set function $\Phi\colon \operatorname{cl}(\Omega_\Phi) \times [0,T] \to \mathbb{R}$ which is defined on the closure of some larger, static bulk domain $\Omega_\Phi \subset \mathbb{R}^d$ that contains $\Omega(t) \cup \Gamma(t)$ at each fixed time t. Moreover, let $\boldsymbol{\nu}(\cdot,t)\colon \Gamma(t) \to \mathbb{R}^d$ denote the field of outward-pointing unit normal vectors to $\Gamma(t)$ and let $\mathbf{v}(\cdot,t)\colon \Omega(t) \cup \Gamma(t) \to \mathbb{R}^d$ be a field which describes the material velocity of $\Omega(t) \cup \Gamma(t)$. As defined in Section 3.3, let $\Gamma_l(t)$, $\partial\Gamma_l(t)$ and $\operatorname{cl}\big(\Gamma_l(t)\big)$ with $l \in \mathbb{R}$ be the level sets of Φ, their boundaries and their closures, respectively, and assume $\Gamma(t) = \Gamma_0(t)$. Furthermore, let the fields $u_b(\cdot,t)\colon \Omega(t) \to \mathbb{R}$ and $u_s(\cdot,t)\colon \Gamma(t) \to \mathbb{R}$ denote the concentrations of a scalar bulk quantity in $\Omega(t)$ and a surface-bound scalar quantity on $\Gamma(t)$, let $\mathcal{D}_b$ and $\mathcal{D}_s$ be bulk and surface diffusivity tensors, and let $f_b(u_b)$, $f_s(u_s)$, $f_{b,s}(u_b,u_s)$ and $f_{s,b}(u_b,u_s)$ be source/sink densities in $\Omega(t)$ or on $\Gamma(t)$, respectively.

5.1.1. Reminder and derivation

The class of model problems which we consider in this chapter is the class of bulk–surface models that has been introduced and discussed in Section 1.2. Given some initial values $u_b(\cdot,0)$ and $u_s(\cdot,0)$, we seek $u_b(\cdot,t)\colon \Omega(t) \to \mathbb{R}$ and $u_s(\cdot,t)\colon \Gamma(t) \to \mathbb{R}$ with

$$\partial_t u_b + \nabla \cdot (u_b \mathbf{v}) - \nabla \cdot (\mathcal{D}_b \nabla u_b) = f_b(u_b) \qquad \text{in } \Omega(t), \quad (5.1a)$$

$$-\mathcal{D}_b \nabla u_b \cdot \boldsymbol{\nu} = -f_{b,s}(u_b,u_s) \qquad \text{on } \Gamma(t), \quad (5.1b)$$

$$\partial^\bullet u_s + u_s(\nabla_\Gamma \cdot \mathbf{v}) - \nabla_\Gamma \cdot (\mathcal{D}_s \nabla_\Gamma u_s) = f_{s,b}(u_b,u_s) + f_s(u_s) \qquad \text{on } \Gamma(t). \quad (5.1c)$$

For details on these equations and their components, please refer to the explanations on equations (1.8).

A corresponding class of parabolic model problems that exactly matches the static geometry special case $\mathbf{v} \equiv \underline{\mathbf{0}}$ of the above class of evolving geometry model problems was introduced and derived in a mathematically rigorous way in Section 4.1.1. Model equations of the form (5.1) can be derived completely analogously using the theory which has been discussed in Section 3.1. They particularly develop from the theory in Section 3.1.1, choosing $\mathcal{M}(t) := \Gamma(t)$ and $\mathbf{v}_{\boldsymbol{\mathcal{M}}}(\cdot, t) := \mathbf{v}(\cdot, t)\big|_{\Gamma(t)}$, and arise directly from the theory in Section 3.1.2. More precisely, we get continuity equation (5.1c) for the surface-bound quantity with concentration u_s by choosing the tangential diffusive surface flux $\mathbf{q_s} := -\mathcal{D}_s \nabla_\Gamma u_s$ according to Fick's first law of diffusion, and by choosing the source/sink density $g_s := f_{s,b}(u_b, u_s) + f_s(u_s)$. Choosing the diffusive flux $\mathbf{q_b} := -\mathcal{D}_b \nabla u_b$ and the source/sink density $g_b := f_b(u_b)$ for the bulk quantity with concentration u_b leads to continuity equation (5.1a). Furthermore, seeking a particular solution whose bulk quantity satisfies the outflux $\mathbf{q_b} \cdot \boldsymbol{\nu} = -f_{b,s}(u_b, u_s)$ on the boundary of $\Omega(t)$ corresponds to supplementing continuity equation (5.1a) with boundary condition (5.1b).

As with the derivation in Section 4.1.1, the above derivation by means of the theory from Section 3.1 reveals that continuity equations (5.1a) and (5.1c) represent underlying equivalent conservation laws. Holding for arbitrary portions $R(t) \subseteq \Omega(t)$ and $M(t) \subseteq \Gamma(t)$ which move with the material velocity $\mathbf{v}$, those conservation laws can be formulated as

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{R(t)} u_b \,\mathrm{d}x = \int_{\partial R(t)} \mathcal{D}_b \nabla u_b \cdot \mathbf{n}_{\boldsymbol{\partial R(t)}} \,\mathrm{d}\sigma \quad + \int_{R(t)} f_b(u_b) \,\mathrm{d}x, \tag{5.2a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{M(t)} u_s \,\mathrm{d}\sigma = \int_{\partial M(t)} \mathcal{D}_s \nabla_\Gamma u_s \cdot \boldsymbol{\mu}_{\boldsymbol{\partial M(t)}} \,\mathrm{d}\varsigma + \int_{M(t)} f_{s,b}(u_b, u_s) + f_s(u_s) \,\mathrm{d}\sigma, \tag{5.2b}$$

(cf. equation (3.3) and equation (3.1), respectively). Since $R(t) := \Omega(t)$ and $M(t) := \Gamma(t)$ are admissible choices and since $\partial\Gamma(t) = \emptyset$, those conservation laws and boundary condition (5.1b) imply that solutions (u_b, u_s) of model problems from class (5.1) satisfy global conservation properties

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} u_b \,\mathrm{d}x = \int_{\Gamma(t)} f_{b,s}(u_b, u_s) \,\mathrm{d}\sigma + \int_{\Omega(t)} f_b(u_b) \,\mathrm{d}x, \tag{5.3a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Gamma(t)} u_s \,\mathrm{d}\sigma = \int_{\Gamma(t)} f_{s,b}(u_b, u_s) \,\mathrm{d}\sigma + \int_{\Gamma(t)} f_s(u_s) \,\mathrm{d}\sigma. \tag{5.3b}$$

Therefore, the derivative of the total amount

$$m(t) := \int_{\Omega(t)} u_b \,\mathrm{d}x + \int_{\Gamma(t)} u_s \,\mathrm{d}\sigma$$

of the system's quantities fulfills

$$\frac{\mathrm{d}}{\mathrm{d}t}\, m(t) = \int_{\Gamma(t)} f_{b,s}(u_b, u_s) + f_{s,b}(u_b, u_s)\ \mathrm{d}\sigma + \int_{\Omega(t)} f_b(u_b)\ \mathrm{d}x + \int_{\Gamma(t)} f_s(u_s)\ \mathrm{d}\sigma. \quad (5.4)$$

Note that properties (5.3) and (5.4) generalize global conservation properties (4.3) and (4.4), respectively. The latter hold in the static geometry special case $\mathbf{v} \equiv \underline{\mathbf{0}}$. Again, the value $m(t)$ is an invariant with respect to time if the model parameters are chosen accordingly, e.g., for models with $f_b \equiv 0$, $f_s \equiv 0$ and $f_{b,s} = -f_{s,b}$.

5.2. Simplifying the problem using operator splitting

Treating evolving geometry problems directly can be quite demanding, given the challenges which have been discussed in Section 1.5. As general approaches to developing efficient numerical schemes for unwieldy initial–boundary value problems or initial value problems, several concepts for constructing operator splitting methods have been introduced in the literature, see e.g. Hundsdorfer and Verwer (2003, Section IV.1) and Maday et al. (1990). Based on an arbitrary but fixed operator splitting, these concepts decompose the problem into a sequence of simpler subproblems that can be treated individually using efficient, problem-specific numerical schemes.

Operator splittings for initial–boundary value problems can be divided into two classes. *Differential* operator splittings decouple different physical effects in the undiscretized problem by splitting spatial differential operators. *Algebraic* operator splittings split discrete operators resulting from semidiscretization of spatial differential operators. For further details on these two techniques, we refer to Kuzmin and Hamalainen (2014, Section 8.2.1). In both cases, often the same concepts for constructing operator splitting methods can be applied, with an error analysis that uses the same arguments, see e.g. Jahnke and Lubich (2000).

5.2.1. Operator splitting for PDEs on evolving geometries

To deal with problems that model conserved bulk/surface quantities on an evolving geometry, such as the class of problems (5.1), we propose a new family of 2-term differential operator splittings. It is based on the following observation.

Mathematical modeling which leads to the class of problems (5.1) features two distinguished physical effects. On the one hand, we have plain conservation of a bulk/surface quantity on an evolving geometry, i.e. conservative material transport which is driven by the geometrical evolution. On the other

hand, we have additional processes which also conserve these quantities, particularly diffusive bulk/surface fluxes as well as reactive processes. The above derivation of equations (5.1) by means of the theory in Section 3.1 shows that plain conservation on the evolving geometry corresponds to the advective terms in the equations. See Section 1.1.2 for supplementary information.

Motivated by the origin of those advective terms, the idea of the family of differential operator splittings which we propose is to split into corresponding advection operators and remaining operators, analogous to classical differential operator splitting of advection and diffusion operators for equations in static bulk domains. Together with a concept for constructing operator splitting methods that is suitable for differential operator splittings, this decomposes the considered class of problems into two types of subproblems.

The first type of subproblems comprises the plainest possible continuity equations which can be formulated on evolving geometries, namely advection equations accounting for conservative material transport driven by geometrical evolution, cf. Section 1.1.2. It is a well-posed problem type without the need for any boundary condition. The second type of subproblems matches the static geometry special case $\mathbf{v} \equiv \underline{\mathbf{0}}$ of the given evolving geometry problem, i.e. it consists of bulk PDEs and surface PDEs on static geometries and the problem's original boundary conditions.

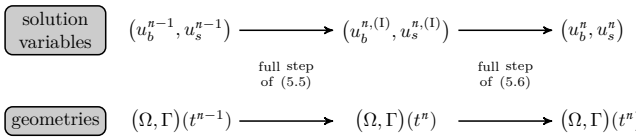
Considering the class of evolving geometry model problems (5.1), the conservative transport problems which make up the first type of subproblems take the form

$$\forall t \in \left(t^{\text{old}}, t^{\text{new}}\right] : \begin{cases} \partial_t u_b + \nabla \cdot (u_b \mathbf{v}) = 0 & \text{in} \quad \Omega(t), \\ \partial^\bullet u_s + u_s (\nabla_\Gamma \cdot \mathbf{v}) = 0 & \text{on} \quad \Gamma(t). \end{cases} \tag{5.5}$$

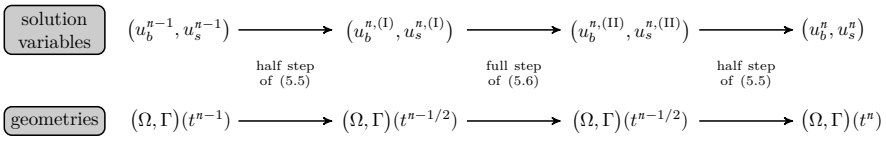
Here, the time interval $\left[t^{\text{old}}, t^{\text{new}}\right]$ is a parameter that takes different values. These values depend on the specific operator splitting method which is applied and will be specified later on. The second type of subproblems matches the class of parabolic model problems from Section 4.1.1. Given again some time interval $\left[t^{\text{old}}, t^{\text{new}}\right]$ and given the geometry at some time t^{geo}, we need to consider coupled, static geometry bulk–surface PDEs of the form

$$\begin{aligned} \partial_t u_b - \nabla \cdot (\mathcal{D}_b \nabla u_b) &= f_b(u_b) && \text{in} \quad \Omega\left(t^{\text{geo}}\right) \times \left(t^{\text{old}}, t^{\text{new}}\right], \\ -\mathcal{D}_b \nabla u_b \cdot \boldsymbol{\nu} &= -f_{b,s}(u_b, u_s) && \text{on} \quad \Gamma\left(t^{\text{geo}}\right) \times \left(t^{\text{old}}, t^{\text{new}}\right], \\ \partial_t u_s - \nabla_\Gamma \cdot (\mathcal{D}_s \nabla_\Gamma u_s) &= f_{s,b}(u_b, u_s) + f_s(u_s) && \text{on} \quad \Gamma\left(t^{\text{geo}}\right) \times \left(t^{\text{old}}, t^{\text{new}}\right]. \end{aligned} \tag{5.6}$$

Instead of having to treat an evolving geometry problem directly, the proposed operator splitting hence allows for dealing with the corresponding type of static geometry problems and an additional type of transport problems. Next, we look at particular sequences of subproblems which result in suitable approximations of solutions to the original evolving geometry problem.



(a) One step of the first order operator splitting method (Lie–Trotter splitting).



(b) One step of the second order operator splitting method (Strang splitting).

Figure 5.1.: Illustrations of the two operator splitting methods for PDEs on evolving geometries which are proposed in Section 5.2.2. To shorten notation, we write $\big(\Omega,\Gamma\big)(t^{\mathrm{geo}}) := \big(\Omega(t^{\mathrm{geo}}),\Gamma(t^{\mathrm{geo}})\big)$ in both illustrations.

5.2.2. Specific operator splitting methods for PDEs on evolving geometries

The basic concept of operator splitting methods is the same as with time-stepping schemes. Time is discretized by dividing the considered observation period $[0,T]$ into subintervals $\big[t^{n-1},t^n\big]$ of length $\tau^n := t^n - t^{n-1}$, $n = 1,\dots,N$, with $t^0 = 0$, $t^N = T$ and $t^n > t^{n-1}$ for $n = 1,\dots,N$. On this basis, every specific operator splitting method performs a sequence of N steps. In the n-th step, the method yields an approximation $\big(u_b^n, u_s^n\big)$ of the solution pair evaluated at $t = t^n$. Each step comprises several stages, at least one stage for each type of subproblems. The sequence of stages corresponds to a certain combination of subproblems. Solving those subproblems one after another introduces an error known as *splitting error*, which can be estimated by some power of the corresponding step size τ^n.

A first order operator splitting method

The most basic operator splitting method we propose results from applying the operator splitting from Section 5.2.1, together with the common concept for constructing first order operator splitting methods which is abstractly known as *Lie–Trotter splitting*. The latter stems back to Trotter (1959). We obtain a method whose steps comprise two stages, precisely one stage for each type of subproblems. In both stages, the respective subproblem that needs to be solved uses the full step size. In the course of this, the order of dealing with subproblem types can be chosen arbitrarily. To present the method in detail, we just fix one specific order.

In particular, considering the class of evolving geometry model problems (5.1) and the step that is associated with the time interval $\left[t^{n-1}, t^n\right]$, the first stage can be described as

$$\text{(I)} \begin{cases} \text{choose initial values} \begin{cases} u_b\big(\cdot, t^{n-1}\big) := u_b^{n-1} & \text{in} \quad \Omega\big(t^{n-1}\big), \\ u_s\big(\cdot, t^{n-1}\big) := u_s^{n-1} & \text{on} \quad \Gamma\big(t^{n-1}\big), \end{cases} \\ \text{solve (5.5) choosing } t^{\text{old}} := t^{n-1} \text{ and } t^{\text{new}} := t^n, \\ \text{set} \begin{cases} u_b^{n,(\text{I})} := u_b(\cdot, t^n) & \text{in} \quad \Omega(t^n), \\ u_s^{n,(\text{I})} := u_s(\cdot, t^n) & \text{on} \quad \Gamma(t^n). \end{cases} \end{cases}$$

This stage deals with conservative material transport between static geometries $\big(\Omega(t^{n-1}), \Gamma(t^{n-1})\big)$ and $\big(\Omega(t^n), \Gamma(t^n)\big)$. The second stage deals with the model's remaining processes. It can be described as

$$\text{(II)} \begin{cases} \text{choose initial values} \begin{cases} u_b\big(\cdot, t^{n-1}\big) := u_b^{n,(\text{I})} & \text{in} \quad \Omega(t^n), \\ u_s\big(\cdot, t^{n-1}\big) := u_s^{n,(\text{I})} & \text{on} \quad \Gamma(t^n), \end{cases} \\ \text{solve (5.6) choosing } t^{\text{geo}} := t^n,\ t^{\text{old}} := t^{n-1} \text{ and } t^{\text{new}} := t^n, \\ \text{set} \begin{cases} u_b^n := u_b(\cdot, t^n) & \text{in} \quad \Omega(t^n), \\ u_s^n := u_s(\cdot, t^n) & \text{on} \quad \Gamma(t^n). \end{cases} \end{cases}$$

The interplay of both stages is illustrated in Figure 5.1a.

Motivated by a formal error analysis of the Lie–Trotter splitting concept (see e.g. Hundsdorfer and Verwer, 2003, Section IV.1.1 and Section IV.1.4), we expect the splitting error with respect to the maximum step size to be of order 1.

A second order operator splitting method

By applying the operator splitting from Section 5.2.1 together with a concept for constructing operator splitting methods which is abstractly known as *Strang splitting*, since its basic idea dates back to Strang (1968), we obtain an advanced operator splitting method which is expected to be second order accurate (cf. Jahnke and Lubich, 2000). The resulting method performs steps comprising three stages, the first and third of which deal with the same type of subproblems. The second stage deals with the other type of subproblems. Just as with the first order operator splitting method above, the subproblem in the second stage uses the full step size, whereas the subproblems in the first stage and in the third stage only use half the step size, respectively. As long as both stages with half steps target the same subproblem type, the order of dealing with subproblem types can be chosen arbitrarily. Next, we fix one specific order to present details.

Again, we particularly consider the class of evolving geometry model prob-

lems (5.1) and the step that is associated with the time interval $\big[t^{n-1}, t^n\big]$. The method's first stage performs one half step of conservative material transport that is solely driven by geometrical evolution. Defining a fractional time $t^{n-1/2} := t^{n-1} + \frac{1}{2}\tau^n$ which corresponds to the midpoint of the considered time interval, it can be described as

$$\text{(I)} \quad \begin{cases} \text{choose initial values } \begin{cases} u_b\big(\cdot, t^{n-1}\big) := u_b^{n-1} & \text{in} \quad \Omega\big(t^{n-1}\big), \\ u_s\big(\cdot, t^{n-1}\big) := u_s^{n-1} & \text{on} \quad \Gamma\big(t^{n-1}\big), \end{cases} \\ \text{solve (5.5) choosing } t^{\text{old}} := t^{n-1} \text{ and } t^{\text{new}} := t^{n-1/2}, \\ \text{set } \begin{cases} u_b^{n,(\text{I})} := u_b\big(\cdot, t^{n-1/2}\big) & \text{in} \quad \Omega\big(t^{n-1/2}\big), \\ u_s^{n,(\text{I})} := u_s\big(\cdot, t^{n-1/2}\big) & \text{on} \quad \Gamma\big(t^{n-1/2}\big). \end{cases} \end{cases}$$

Here, material is transported between two static geometries $\big(\Omega(t^{n-1}), \Gamma(t^{n-1})\big)$ and $\big(\Omega(t^{n-1/2}), \Gamma(t^{n-1/2})\big)$. Fixing the latter geometry, the second stage of the method performs one full step which deals with the model's remaining processes. It can be described as

$$\text{(II)} \quad \begin{cases} \text{choose initial values } \begin{cases} u_b\big(\cdot, t^{n-1}\big) := u_b^{n,(\text{I})} & \text{in} \quad \Omega\big(t^{n-1/2}\big), \\ u_s\big(\cdot, t^{n-1}\big) := u_s^{n,(\text{I})} & \text{on} \quad \Gamma\big(t^{n-1/2}\big), \end{cases} \\ \text{solve (5.6) choosing } t^{\text{geo}} := t^{n-1/2},\ t^{\text{old}} := t^{n-1} \text{ and } t^{\text{new}} := t^n, \\ \text{set } \begin{cases} u_b^{n,(\text{II})} := u_b(\cdot, t^n) & \text{in} \quad \Omega\big(t^{n-1/2}\big), \\ u_s^{n,(\text{II})} := u_s(\cdot, t^n) & \text{on} \quad \Gamma\big(t^{n-1/2}\big). \end{cases} \end{cases}$$

In the method's third stage, material is finally transported between two static geometries $\big(\Omega(t^{n-1/2}), \Gamma(t^{n-1/2})\big)$ and $\big(\Omega(t^n), \Gamma(t^n)\big)$ by performing another half step of conservative transport which is solely driven by geometrical evolution:

$$\text{(III)} \quad \begin{cases} \text{choose initial values } \begin{cases} u_b\big(\cdot, t^{n-1/2}\big) := u_b^{n,(\text{II})} & \text{in} \quad \Omega\big(t^{n-1/2}\big), \\ u_s\big(\cdot, t^{n-1/2}\big) := u_s^{n,(\text{II})} & \text{on} \quad \Gamma\big(t^{n-1/2}\big), \end{cases} \\ \text{solve (5.5) choosing } t^{\text{old}} := t^{n-1/2} \text{ and } t^{\text{new}} := t^n, \\ \text{set } \begin{cases} u_b^n := u_b(\cdot, t^n) & \text{in} \quad \Omega(t^n), \\ u_s^n := u_s(\cdot, t^n) & \text{on} \quad \Gamma(t^n). \end{cases} \end{cases}$$

The interplay of those stages is illustrated in Figure 5.1b.

Global conservation properties

Next, we examine how global conservation properties are reflected by the two operator splitting methods proposed above. As usual, we consider our class

of evolving geometry model problems (5.1) and the corresponding types of subproblems as an example for this purpose.

Solutions to subproblems of type (5.5) fulfill global conservation properties

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} u_b \,\mathrm{d}x = 0 \quad \forall t \in \left(t^{\mathrm{old}}, t^{\mathrm{new}}\right], \tag{5.7a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Gamma(t)} u_s \,\mathrm{d}\sigma = 0 \quad \forall t \in \left(t^{\mathrm{old}}, t^{\mathrm{new}}\right]. \tag{5.7b}$$

They follow from equations (5.3) since the type of subproblems corresponds to the special case

$$\begin{aligned} &\mathcal{D}_b \equiv 0, \quad f_{b,s}(u_b, u_s) \equiv 0, \quad f_b(u_b) \equiv 0, \\ &\mathcal{D}_s \equiv 0, \quad f_{s,b}(u_b, u_s) \equiv 0, \quad f_s(u_s) \equiv 0, \end{aligned}$$

of the class of evolving geometry model problems (5.1). In addition, recalling that subproblems of type (5.6) are members of the class of parabolic model problems from Section 4.1.1, solutions to subproblems of type (5.6) bring along global conservation properties

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t^{\mathrm{geo}})} u_b \,\mathrm{d}x = \int_{\Gamma(t^{\mathrm{geo}})} f_{b,s}(u_b, u_s) \,\mathrm{d}\sigma + \int_{\Omega(t^{\mathrm{geo}})} f_b(u_b) \,\mathrm{d}x, \tag{5.8a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Gamma(t^{\mathrm{geo}})} u_s \,\mathrm{d}\sigma = \int_{\Gamma(t^{\mathrm{geo}})} f_{s,b}(u_b, u_s) \,\mathrm{d}\sigma + \int_{\Gamma(t^{\mathrm{geo}})} f_s(u_s) \,\mathrm{d}\sigma, \tag{5.8b}$$

which hold for all $t \in \left(t^{\mathrm{old}}, t^{\mathrm{new}}\right]$, cf. equations (4.3). If subproblems of both types are solved using numerical schemes which recover discrete analogues to global conservation properties (5.7) and (5.8), respectively, our operator splitting methods recover discrete analogues to conservation properties (5.3) and (5.4).

The latter statement is not hard to verify. Let us particularly consider the first order operator splitting method, for instance, and the step that is associated with the time interval $\left[t^{n-1}, t^n\right]$. Provided that the scheme dealing with the subproblem in the first stage yields discrete analogues to properties (5.7), those discrete analogues usually are of the form

$$\begin{aligned} \int_{\Omega_h(t^n)} u_{b,h}^{n,(\mathrm{I})} \,\mathrm{d}x &= \int_{\Omega_h(t^{n-1})} u_{b,h}^{n-1} \,\mathrm{d}x, \\ \int_{\Gamma_h(t^n)} u_{s,h}^{n,(\mathrm{I})} \,\mathrm{d}\sigma &= \int_{\Gamma_h(t^{n-1})} u_{s,h}^{n-1} \,\mathrm{d}\sigma, \end{aligned}$$

since they are supposed to account for the fact that the amounts of the continuous system's quantities remain constant over time. Moreover, it is reasonable to assume that, if the scheme which deals with the subproblem in the second

stage yields discrete analogues to properties (5.8), those discrete analogues are similar to

$$\begin{aligned}\int_{\Omega_{\mathfrak{h}}(t^n)} u_{b,h}^n \,\mathrm{d}x &= \int_{\Omega_{\mathfrak{h}}(t^n)} u_{b,h}^{n,(\mathrm{I})} \,\mathrm{d}x \\ &\quad + \tau^n \Big(\int_{\Gamma_{\mathfrak{h}}(t^n)} f_{b,s}^n\big(u_{b,h}^n, u_{s,h}^n\big) \,\mathrm{d}\sigma + \int_{\Omega_{\mathfrak{h}}(t^n)} f_b^n\big(u_{b,h}^n\big) \,\mathrm{d}x \Big), \\ \int_{\Gamma_{\mathfrak{h}}(t^n)} u_{s,h}^n \,\mathrm{d}\sigma &= \int_{\Gamma_{\mathfrak{h}}(t^n)} u_{s,h}^{n,(\mathrm{I})} \,\mathrm{d}\sigma + \tau^n \int_{\Gamma_{\mathfrak{h}}(t^n)} f_{s,b}^n\big(u_{b,h}^n, u_{s,h}^n\big) + f_s^n\big(u_{s,h}^n\big) \,\mathrm{d}\sigma,\end{aligned}$$

cf. Theorem 4.2.20. In the case which we are interested in, the first order operator splitting method hence recovers equations similar to

$$\begin{aligned}\int_{\Omega_{\mathfrak{h}}(t^n)} u_{b,h}^n \,\mathrm{d}x &= \int_{\Omega_{\mathfrak{h}}(t^{n-1})} u_{b,h}^{n-1} \,\mathrm{d}x, \\ &\quad + \tau^n \Big(\int_{\Gamma_{\mathfrak{h}}(t^n)} f_{b,s}^n\big(u_{b,h}^n, u_{s,h}^n\big) \,\mathrm{d}\sigma + \int_{\Omega_{\mathfrak{h}}(t^n)} f_b^n\big(u_{b,h}^n\big) \,\mathrm{d}x \Big), \\ \int_{\Gamma_{\mathfrak{h}}(t^n)} u_{s,h}^n \,\mathrm{d}\sigma &= \int_{\Gamma_{\mathfrak{h}}(t^{n-1})} u_{s,h}^{n-1} \,\mathrm{d}\sigma, + \tau^n \int_{\Gamma_{\mathfrak{h}}(t^n)} f_{s,b}^n\big(u_{b,h}^n, u_{s,h}^n\big) + f_s^n\big(u_{s,h}^n\big) \,\mathrm{d}\sigma.\end{aligned}$$

The latter equations are indeed discrete analogues to properties (5.3), and can be combined into a discrete analogue to property (5.4). Note that it is possible to proceed similarly for the second order operator splitting method which is proposed above.

5.2.3. Related splitting approaches

Various splitting approaches for treating problems that comprise PDEs on evolving geometries have been discussed in the literature so far. To the best knowledge of the author of this thesis, those splitting approaches are different from the operator splitting approach which is proposed above. However, many of them show features that are similar to those of the proposed approach, such as keeping the geometry fixed during selected splitting stages, or being based on some differential operator splitting.

For instance, splitting approaches can be found in the literature that employ a fixed geometry during whole time steps. While keeping the geometry fixed during each full time step, the result is extrapolated to the next geometrical setup to obtain initial values for the next time step. A splitting approach of this kind is investigated by Kummer et al. (2018). Their study shows that the approach does not perform well in the context of UDG discretizations of problems which comprise continuity equations with moving interfaces that partition a static bulk domain into two evolving subdomains. In this context, it should be mentioned that extrapolating solution variables between time steps does not account for the fact that geometrical evolution actually drives

material transport. Performing geometrical evolution together with extrapolation instead of real material transport impedes obtaining conservative schemes and, as observed in Kummer et al. (2018), it calls for extremely small time steps. Splitting approaches of this kind are fundamentally different from the approach which we are proposing, given that the design of the latter allows for globally conservative methods.

Another example can be found in Heimann et al. (2013). Here, an operator splitting approach is employed to develop a UDG method for incompressible Navier–Stokes two-phase flow. In this method, the Strang splitting concept is also applied to a differential operator splitting. Unlike the differential operator splitting from Section 5.2.1, the latter does not target conservative material transport driven by geometrical evolution, but rather focusses on allowing for an individual numerical treatment of level set transport for interface movement and of the instationary Navier–Stokes equations.

5.2.4. Treating the resulting subproblems

As advertised, each subproblem type which arises in our operator splitting approach can be treated with a specialized numerical scheme. Henceforth, we refer to such schemes as subproblem schemes. Each subproblem scheme should meet two different kinds of requirements. On the one hand, the order of the specific operator splitting method which is employed calls for a subproblem scheme with certain convergence properties. On the other hand, the scheme needs to deal with the given subproblem type in an adequate manner. The first requirement is considered next, adequate schemes for the types of subproblems associated with our class of evolving geometry model problems (5.1) will be discussed further down below.

Focussing on the first order operator splitting method, subproblem schemes can be considered as having suitable convergence properties if they are first order accurate with respect to time. In this case, the resulting overall scheme is expected to yield discretization errors of order 1 with respect to time.

Considering the second order operator splitting method, each subproblem scheme needs to be at least second order accurate with respect to time to benefit from the higher order splitting error of the method. Otherwise the lower order accuracy of a subproblem scheme will dominate discretization errors of the overall scheme. Conversely, if higher order subproblem schemes are used, the second order splitting error of the second order operator splitting method will dominate overall discretization errors with respect to time. For the second order operator splitting method, it is thus sufficient to employ subproblem schemes that are exactly second order accurate with respect to time.

Treating static geometry subproblems

Subproblems of type (5.6) can be solved, in particular, using the UDG schemes for bulk–surface models on static geometries which we developed in Chapter 4. These schemes are globally conservative according to Theorem 4.2.20. As we have seen in Section 5.2.2, they are hence suitable for obtaining overall schemes that recover discrete analogues to global conservation properties.

In case of the second order operator splitting method, the backward Euler time-stepping which is employed by our UDG schemes should be replaced by some suitable second order time integrator, such as the implicit midpoint method (see e.g. Hairer et al., 2006, Section I.1.2 and Section II.1.1).

Treating subproblems which deal with conservative material transport

Next, we want to consider conservative numerical schemes for treating the conservative material transport problems of type (5.5). As motivated in Sections 1.5 and 1.6, we deliberately aim at UDG schemes.

Our splitting approach successfully reduces the complexity of the transport problem, such that the material velocity material velocity $\mathbf{v}$ of $\Omega(t) \cup \Gamma(t)$ is the only data term remaining. Therefore, both equations in (5.5) can be treated independently of each other, provided that $\mathbf{v}$ does not depend on the solution pair (u_b, u_s). This applies if $\mathbf{v}$ is an externally given field.

For the bulk advection equation in (5.5) the characteristic–Galerkin method by Pironneau et al. (1992) can be employed. It was designed for the more general class of advection–diffusion equations with evolving bulk domains. Performing advective transport along the characteristics, the method yields unconditionally stable schemes which do not require stabilization mechanisms such as upwinding. The essential part of the method, the discretization in time, can be combined with a UDG discretization in space. By default, the method unfortunately does not result in schemes which recover discrete analogues to conservation properties that are embedded in the considered type of continuity equations. However, it is claimed in Pironneau et al. (1992) that the method allows for conservative modifications. Recently, the essential idea of the characteristic–Galerkin method was combined with a piecewise linear cutFEM for surface PDEs (sharp interface FEM) to treat advection–diffusion equations on an evolving hypersurface (Hansbo et al., 2015).

The surface advection equation in subproblems of type (5.5) can be treated by extending the equation to some evolving bulk domain which contains the evolving hypersurface $\Gamma(t)$, and by subsequently applying a method for bulk advection equations. In this way, the same method can be used for both types of advection equations in (5.5). We emphasize that Theorem 2.6.3 allows for an extension process analogous to the one which we introduced in Section 4.2.1. In particular, an evolving narrow band around $\Gamma(t)$ can be chosen as a surface extension domain. It is possible to combine this choice with the ideas which yield conservative and stable UDG schemes in Section 4.2.

In the next section, we introduce a new kind of conservative UDG scheme for the surface advection equation in subproblems of type (5.5). Similar to the extension process which is described above, this scheme is based on a strategy which allows for treating the surface equation using methods for bulk advection equations. As opposed to the extension process, the surface equation is not rewritten by means of Theorem 2.6.3 and extended afterwards. Still, the basis of our scheme is comparable with the extension process in the sense that the level set description of the geometry is exploited to reformulate the surface equation in terms of equations with bulk advection terms.

5.3. An unfitted DG scheme for an essential type of continuity equations on evolving hypersurfaces

Keeping in mind the operator splitting methods introduced in Section 5.2.2, in the following, we focus on the surface part of those stages which deal with conservative material transport driven by geometrical evolution. In particular, we consider the surface advection equation in subproblems of type (5.5). Given some time interval $\left[t^{\mathrm{old}}, t^{\mathrm{new}}\right]$ and initial values $u_s\big(\cdot, t^{\mathrm{old}}\big)$, we wish to find a time-dependent surface field $u_s(\cdot, t)\colon \Gamma(t) \to \mathbb{R}$ with

$$\forall t \in \left(t^{\mathrm{old}}, t^{\mathrm{new}}\right]\colon \quad \partial^\bullet u_s + u_s(\nabla_\Gamma \cdot \mathbf{v_s}) = 0 \quad \text{on} \quad \Gamma(t), \tag{5.9}$$

where we write $\mathbf{v_s} := \mathbf{v}|_{\Gamma(t)}$ for the material velocity of $\Gamma(t)$ to clarify notation.

The scheme which we propose is based on performing discretization in time, reformulating each time step in an approximate manner as a stationary bulk advection problem with singular source and sink terms, and applying the UDG method to this problem. In the course of this, a static computational mesh can be used which does not explicitly track the evolving hypersurface.

Without loss of generality, we restrict the following considerations to one single time step of size $\tau := t^{\mathrm{new}} - t^{\mathrm{old}}$. Recalling the operator splitting methods from Section 5.2.2 in this light, this setting corresponds to the one which is usually found in operator splitting methods, if no substepping is used to deal with the subproblems that arise in each step of such methods.

5.3.1. Approximate reformulation of surface equations

We begin by reformulating the given problem in an exact manner. Proceeding formally, we multiply problem (5.9) by a smooth test function $\varphi_s\colon \Omega_\Phi \to \mathbb{R}$, integrate over $\Gamma(t)$ for $t \in \left[t^{\mathrm{old}}, t^{\mathrm{new}}\right]$ and apply the transport relation from Section 2.6 (Theorem 2.6.2) to see the weak formulation

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{\Gamma(t)} u_s\, \varphi_s \,\mathrm{d}\sigma\right) - \int_{\Gamma(t)} u_s\, \partial^\bullet \varphi_s \,\mathrm{d}\sigma = 0.$$

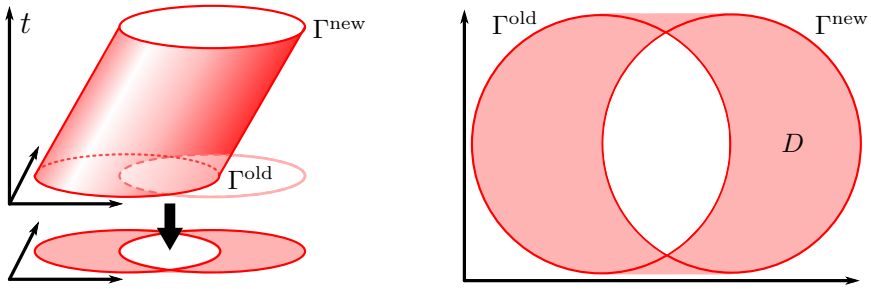


Figure 5.2.: A circle which is translated with a horizontal velocity in the time step associated with $[t^{\mathrm{old}}, t^{\mathrm{new}}]$. Left: Projection of the space–time representation to spatial-only coordinates. Right: The resulting static bulk domain D in detail. Note that a time step with a large time step size τ is illustrated.

Since the test function φ_s does not depend on time, we have $\partial^\bullet \varphi_s = \mathbf{v_s} \cdot \nabla \varphi_s$ in the second integral. Integrating in time over the considered time interval $[t^{\mathrm{old}}, t^{\mathrm{new}}]$ subsequently yields

$$\int_{\Gamma(t^{\mathrm{new}})} u_s(\cdot, t^{\mathrm{new}})\, \varphi_s \,\mathrm{d}\sigma - \int_{\Gamma(t^{\mathrm{old}})} u_s(\cdot, t^{\mathrm{old}})\, \varphi_s \,\mathrm{d}\sigma - \int_{t^{\mathrm{old}}}^{t^{\mathrm{new}}} \left(\int_{\Gamma(t)} u_s \mathbf{v_s} \cdot \nabla \varphi_s \,\mathrm{d}\sigma \right) \mathrm{d}t = 0. \quad (5.10)$$

Our approximate reformulation of problem (5.9) is based on approximating the space–time integral in the third term of equation (5.10) by a properly scaled integral over the static bulk domain

$$D := \mathrm{int}\left(\bigcup\nolimits_{t \in [t^{\mathrm{old}}, t^{\mathrm{new}}]} \Gamma(t) \right) \subset \mathbb{R}^d, \quad (5.11)$$

while searching for a bulk field $u_s^{\mathrm{ext}} \colon D \to \mathbb{R}$ that is an extension of some approximation of the surface field $u_s(\cdot, t^{\mathrm{new}})$ which we are interested in. The static bulk domain D results from projecting the space–time representation

$$\bigcup\nolimits_{t \in [t^{\mathrm{old}}, t^{\mathrm{new}}]} \Gamma(t) \times \{t\} \subset \mathbb{R}^{d+1}$$

of the evolving hypersurface $\Gamma(t)$ to spatial-only coordinates and considering the interior of the resulting set in $\mathbb{R}^d$. Examples of possible geometrical setups are shown in Figure 5.2 and Figure 5.3.

Employing the right-hand rectangle method to approximate the outer integral in the third term of equation (5.10) as with backward Euler time-stepping,

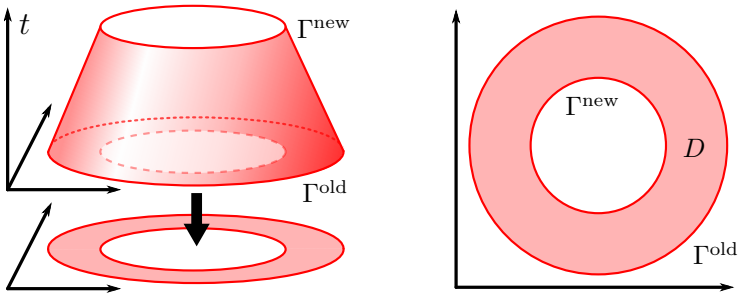


Figure 5.3.: A circle which is shrinking with a velocity that points in normal direction in the time step which is associated with $[t^{\text{old}}, t^{\text{new}}]$. Left: Projection of the space–time representation to spatial-only coordinates. Right: The resulting static bulk domain D in detail.

we have

$$-\int_{t^{\text{old}}}^{t^{\text{new}}} \left(\int_{\Gamma(t)} u_s \mathbf{v_s} \cdot \nabla \varphi_s \, \mathrm{d}\sigma \right) \mathrm{d}t \approx -\tau \int_{\Gamma(t^{\text{new}})} u_s \mathbf{v_s} \cdot \nabla \varphi_s \, \mathrm{d}\sigma. \tag{5.12}$$

This approximation is reasonable if we assume that the time step size τ is small. If the time step size is small, it is also reasonable to assume that the value $\gamma := \gamma^- + \gamma^+ \in \mathbb{R}$ with $\gamma^- := -\inf_D \Phi(\cdot, t^{\text{new}})$ and $\gamma^+ := \sup_D \Phi(\cdot, t^{\text{new}})$ is small. Note that γ corresponds to the travel distance of an individual point on the surface if the material velocity $\mathbf{v_s}$ is constant and Φ is a signed distance function, i.e. if $|\nabla \Phi| \equiv 1$. Assuming that γ is indeed small, we approximate the surface integral on the right-hand side of equation (5.12) using the level sets $\Gamma_l(t^{\text{new}}) := \{\underline{\mathbf{x}} \in D \mid \Phi(\underline{\mathbf{x}}, t^{\text{new}}) = l\}$ of $\Phi(\cdot, t^{\text{new}})\big|_D$ and apply the coarea formula (see e.g. Federer, 1959, Theorem 3.1). This gives

$$\begin{aligned} -\tau \int_{\Gamma(t^{\text{new}})} u_s \mathbf{v_s} \cdot \nabla \varphi_s \, \mathrm{d}\sigma &\approx -\frac{\tau}{\gamma} \int_{-\gamma^-}^{\gamma^+} \left(\int_{\Gamma_l(t^{\text{new}})} u_s^{\text{ext}} \mathbf{v_s^{ext}} \cdot \nabla \varphi_s \, \mathrm{d}\sigma \right) \mathrm{d}l \\ &= -\frac{\tau}{\gamma} \int_D u_s^{\text{ext}} \mathbf{v_s^{ext}} \big| \nabla \Phi(\cdot, t^{\text{new}}) \big| \cdot \nabla \varphi_s \, \mathrm{d}x, \end{aligned}$$

where $u_s^{\text{ext}} \colon D \to \mathbb{R}$ is an extended solution variable which we are looking for in the following. As announced earlier, it represents approximately an extension of the surface field $u_s(\cdot, t^{\text{new}})$ to the static bulk domain D. Moreover, $\mathbf{v_s^{ext}} \colon D \to \mathbb{R}^d$ denotes some corresponding extension of the material velocity $\mathbf{v_s}(\cdot, t^{\text{new}})$. It needs to be given and determines the extension u_s^{ext}.

As an approximate reformulation of problem (5.9), we therefore obtain the following stationary problem. Given a surface field $u_s(\cdot, t^{\text{old}}) \colon \Gamma(t^{\text{old}}) \to \mathbb{R}$,

we seek a bulk field $u_s^{\mathrm{ext}}\colon D \to \mathbb{R}$ which satisfies

$$\int_{\Gamma^{\mathrm{new}}} u_s^{\mathrm{ext}}\,\varphi_s \,\mathrm{d}\sigma - \int_{\Gamma^{\mathrm{old}}} u_s^{\mathrm{old}}\,\varphi_s \,\mathrm{d}\sigma - \frac{\tau}{\gamma}\int_D u_s^{\mathrm{ext}}\mathbf{v_s^{ext}}\big|\nabla\Phi(\cdot,t^{\mathrm{new}})\big|\cdot\nabla\varphi_s \,\mathrm{d}x = 0$$

for all smooth functions $\varphi_s\colon D \to \mathbb{R}$. Here and in the following, we use the notation $\Gamma^{\mathrm{new}} = \Gamma(t^{\mathrm{new}})$, $\Gamma^{\mathrm{old}} = \Gamma(t^{\mathrm{old}})$ and $u_s^{\mathrm{old}} = u_s(\cdot,t^{\mathrm{old}})$.

For being able to discretize using flux-based approaches, such as the UDG method, we finally use integration by parts to reformulate the integral over the bulk domain D via the divergence of an advective flux tested with φ_s:

$$\begin{aligned}\int_{\Gamma^{\mathrm{new}}} u_s^{\mathrm{ext}}\,\varphi_s \,\mathrm{d}\sigma - \int_{\Gamma^{\mathrm{old}}} u_s^{\mathrm{old}}\,\varphi_s \,\mathrm{d}\sigma + \frac{\tau}{\gamma}\int_D \nabla\cdot\Big(u_s^{\mathrm{ext}}\mathbf{v_s^{ext}}\big|\nabla\Phi(\cdot,t^{\mathrm{new}})\big|\Big)\varphi_s \,\mathrm{d}x \\ - \frac{\tau}{\gamma}\int_{\partial D} u_s^{\mathrm{ext}}\mathbf{v_s^{ext}}\big|\nabla\Phi(\cdot,t^{\mathrm{new}})\big|\cdot\mathbf{n_{\partial D}}\,\varphi_s \,\mathrm{d}\sigma = 0. \quad (5.13)\end{aligned}$$

Here, $\mathbf{n_{\partial D}}$ denotes the field of outward-pointing unit normal vectors to ∂D.

Remark 5.3.1 (Strong formulation)**.** *In the special case where* Γ^{old} *and* Γ^{new} *are disjoint sets with* $\partial D = \Gamma^{\mathrm{old}} \dot\cup \Gamma^{\mathrm{new}}$ *(e.g. in case of the geometrical setup that is shown in Figure 5.3), our reformulated problem given by equation* (5.13) *is consistent with the following bulk advection problem in strong form: given boundary values* $u_s^{\mathrm{old}}\colon \Gamma^{\mathrm{old}} \to \mathbb{R}$*, find* $u_s^{\mathrm{ext}}\colon D \to \mathbb{R}$ *with*

$$\begin{aligned}
&\nabla\cdot\Big(u_s^{\mathrm{ext}}\mathbf{v_s^{ext}}\big|\nabla\Phi(\cdot,t^{\mathrm{new}})\big|\Big) = 0 && \text{in}\quad D,\\
&u_s^{\mathrm{ext}}\mathbf{v_s^{ext}}\big|\nabla\Phi(\cdot,t^{\mathrm{new}})\big|\cdot\mathbf{n_{\partial D}} = -\frac{\gamma}{\tau}u_s^{\mathrm{old}} && \text{on}\quad \Gamma^{\mathrm{old}},\\
&u_s^{\mathrm{ext}}\mathbf{v_s^{ext}}\big|\nabla\Phi(\cdot,t^{\mathrm{new}})\big|\cdot\mathbf{n_{\partial D}} = \frac{\gamma}{\tau}u_s^{\mathrm{ext}} && \text{on}\quad \Gamma^{\mathrm{new}}.
\end{aligned}$$

5.3.2. Unfitted discontinuous Galerkin

In order to discretize the reformulated problem which is given by equation (5.13), we use the UDG method that has been introduced in Section 4.2.2.

Basic meshes, discretization of the computational geometry

We recall that the UDG method which we are considering uses two fixed, simple meshes of the level set domain Ω_Φ. On the one hand, it employs a fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ of width h to construct geometrically unfitted discrete function spaces that are suitable for discretizing PDEs in bulk domains contained in Ω_Φ. On the other hand, a geometry mesh $\mathcal{T}_{\hbar}(\Omega_\Phi)$ of width $\hbar \leq h$ is employed to obtain a discrete analogue to the continuous level set description. The latter provides the basis for some discrete reconstruction of the geometry and for numerical integration over this discrete geometry. Both $\mathcal{T}_h(\Omega_\Phi)$ and $\mathcal{T}_{\hbar}(\Omega_\Phi)$ are expected to comprise shape regular elements that are tetrahedra or hexahedra for $d = 3$, and triangles or quadrilaterals for $d = 2$.

After choosing a specific geometry mesh $\mathcal{T}_\hbar(\Omega_\Phi)$, we start by discretizing the computational geometry similar to what has been presented in Section 4.2.2. Let $X_\hbar(\Omega_\Phi)$ be the space of piecewise linear (for simplices), bilinear (for quadrilaterals) or trilinear (for hexahedra) continuous functions over $\mathcal{T}_\hbar(\Omega_\Phi)$ and let $I_\hbar$ denote an operator which performs interpolation of functions in $C^0\big(\Omega_\Phi\big)$ into $X_\hbar(\Omega_\Phi)$. To obtain a discrete geometry reconstruction, we define some discrete level set function $\Phi_\hbar$ as a time-dependent function over $\mathcal{T}_\hbar(\Omega_\Phi)$ by setting $\Phi_\hbar(\cdot,t) := I_\hbar\big[\Phi(\cdot,t)\big]$. Subsequently, we define discrete analogues to the continuous level set function's level sets $\Gamma_l(t)$ defined in equation (3.23), to the fixed-in-time hypersurfaces Γ^{old} and Γ^{new}, and to the static bulk domain D that has been defined in equation (5.11), particularly by setting

$$\Gamma_{l,\hbar}(t) := \Phi_\hbar(\cdot,t)\big|_{\Omega_\Phi}^{-1}(l) = \big\{\underline{\mathbf{x}} \in \Omega_\Phi \;\big|\; \Phi_\hbar(\underline{\mathbf{x}},t) = l\big\}, \quad l \in \mathbb{R}, \quad t \in [0,T],$$

$$\Gamma_\hbar^{\text{old}} := \Gamma_{0,\hbar}(t^{\text{old}}), \quad \Gamma_\hbar^{\text{new}} := \Gamma_{0,\hbar}(t^{\text{new}}) \;\text{ and }\; D_\hbar := \operatorname{int}\left(\bigcup\nolimits_{t\in[t^{\text{old}},t^{\text{new}}]} \Gamma_{0,\hbar}(t)\right).$$

Note that $D_\hbar$ is an open set in $\mathbb{R}^d$ with $\Gamma_\hbar^{\text{old}}, \Gamma_\hbar^{\text{new}} \subset D_\hbar \cup \partial D_\hbar$. See Figure 5.4a for illustrations.

We complement the discretized computational geometry as in Section 4.2.2 by assigning a piecewise variant of the (transposed) classical gradient operator in $\mathbb{R}^d$ to the space $X_\hbar(\Omega_\Phi)$. Using this piecewise variant, we obtain a discrete analogue to the field $\nabla\Phi(\cdot,t^{\text{new}})$ which is used in the reformulated problem given by equation (5.13). Applied to $\Phi_\hbar(\cdot,t^{\text{new}})$, it can be defined by setting $\nabla_\hbar\Phi_\hbar(\cdot,t^{\text{new}})\big|_K := \nabla\Big[\Phi_\hbar(\cdot,t^{\text{new}})\big|_K\Big]$ for each $K \in \mathcal{T}_\hbar(\Omega_\Phi)$, and by extending this definition to the faces of geometry mesh elements, i.e. to the whole level set domain Ω_Φ and its boundary. On each face of an element, we do the latter by evaluating the gradient in an arbitrarily chosen adjacent element.

Meshes and discrete approximation spaces

Analogous to our way of proceeding in Section 4.2.2, we use the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ and the discretized computational geometry to define an active mesh for the bulk domain $D_\hbar$ by

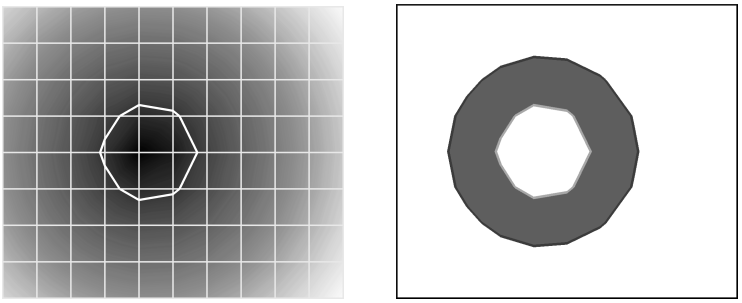
$$\hat{\mathcal{T}}_h(D_\hbar) := \big\{\hat{K} \in \mathcal{T}_h(\Omega_\Phi) \;\big|\; \operatorname{meas}_{\mathbb{R}^d}\big(\hat{K} \cap D_\hbar\big) > 0\big\},$$

and a cut cell mesh of $D_\hbar$ by

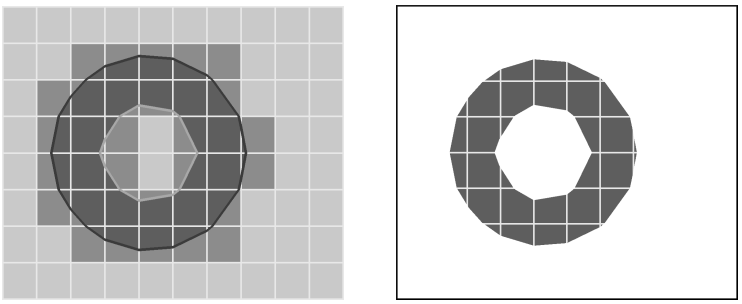
$$\mathcal{T}_h(D_\hbar) := \big\{K = \hat{K} \cap D_\hbar \;\big|\; \hat{K} \in \hat{\mathcal{T}}_h(D_\hbar)\big\}.$$

Furthermore, we define the internal skeleton

$$\mathcal{E}_h^{\text{int}}(D_\hbar) := \big\{E = \partial K_E^+ \cap \partial K_E^- \;\big|\; K_E^+, K_E^- \in \mathcal{T}_h(D_\hbar),\, K_E^+ \neq K_E^-,\, \operatorname{meas}_{\mathbb{R}^{d-1}}(E) > 0\big\}$$



(a) Discrete reconstruction of an example geometry.



(b) Meshes for the reconstructed geometry.

Figure 5.4.: Discrete geometry reconstruction and meshes that are employed by the considered UDG method, illustrated for a circle which is shrinking with a velocity that points in normal direction.
The left image in (a) shows a discrete level set function Φ_h (its values visualized by shades of gray) and its zero level set $\Gamma_{0,h}(t)$ (depicted in white) at a fixed time t, together with the geometry mesh $\mathcal{T}_h(\Omega_\Phi)$ which is used (represented by horizontal and vertical lines). Considering the time step associated with $[t^{\text{old}}, t^{\text{new}}]$, the right image in (a) shows the reconstructed geometry comprising the hypersurface Γ_h^{old}, the bulk domain D_h and the hypersurface Γ_h^{new} (from dark gray to light gray).
The left image in (b) shows the reconstructed geometry again. Moreover, it shows some fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$, together with the corresponding active mesh $\hat{\mathcal{T}}_h(D_h)$ and cut cell mesh $\mathcal{T}_h(D_h)$ (from light gray to dark gray, where pixels which contribute to multiple meshes have the color of the most specialized mesh). Note that we illustrate the special case $\mathcal{T}_h(\Omega_\Phi) = \mathcal{T}_h(\Omega_\Phi)$. The right image in (b) shows the cut cell mesh $\mathcal{T}_h(D_h)$ and its internal skeleton $\mathcal{E}_h^{\text{int}}(D_h)$.

of the cut cell mesh $\mathcal{T}_h(D_{\mathit{fl}})$, i.e. the set which comprises the internal faces of $\mathcal{T}_h(D_{\mathit{fl}})$. See Figure 5.4b for illustrations of the above meshes and of $\mathcal{E}_h^{\mathrm{int}}(D_{\mathit{fl}})$. After assigning the names $K_E^+, K_E^- \in \mathcal{T}_h(D_{\mathit{fl}})$ to the adjacent elements of an arbitrary but fixed internal face $E \in \mathcal{E}_h^{\mathrm{int}}(D_{\mathit{fl}})$ in a fixed manner, we assign a dedicated field of unit normal vectors to E by setting $\mathbf{n_E} := \mathbf{n}_{\boldsymbol{\partial K_E^+}}\big|_E$.

As with the procedure in Section 4.2.2, we use standard DG shape functions on the active mesh $\hat{\mathcal{T}}_h(D_{\mathit{fl}})$, with their support restricted to the cut cells in $\mathcal{T}_h(D_{\mathit{fl}})$, to construct discrete finite element spaces of piecewise polynomial functions over D_{fl} given by

$$V_{s,h}(D_{\mathit{fl}}) := \Big\{ v_{s,h} \in L^2(D_{\mathit{fl}}) \;\Big|\; v_{s,h}\big|_K \in \mathbb{P}(K) \;\forall K \in \mathcal{T}_h(D_{\mathit{fl}}) \Big\}.$$

Here, $\mathbb{P}(K)$ again denotes some space of polynomial functions over an element K, such as the space $P_k(K)$ of polynomial functions of total degree less than or equal to some $k \in \mathbb{N}$, or the space $Q_k(K)$ of polynomial functions with a degree less than or equal to k in each coordinate direction.

In addition, we define two operators $[\![\cdot]\!]$ and $\{ \cdot \}$ which describe the jump and the average of functions that have a reasonable definition on each internal face in $\mathcal{E}_h^{\mathrm{int}}(D_{\mathit{fl}})$, with two potentially different branches on each of those faces. On an internal face E with adjacent elements $K_E^+, K_E^- \in \mathcal{T}_h(D_{\mathit{fl}})$, the action of these two operators on some function $v_{s,h}$ with branches $v_{s,h}|_{\partial K_E^+}$ and $v_{s,h}|_{\partial K_E^-}$ is given by

$$[\![v_{s,h}]\!]\big|_E := v_{s,h}\big|_{\partial K_E^+} - v_{s,h}\big|_{\partial K_E^-}, \quad \{v_{s,h}\}\big|_E := \frac{1}{2}\Big(v_{s,h}\big|_{\partial K_E^+} + v_{s,h}\big|_{\partial K_E^-} \Big).$$

For functions in the discrete spaces $V_{s,h}(D_{\mathit{fl}})$, we also consider a piecewise variant of the (transposed) classical gradient operator in $\mathbb{R}^d$. Given a function $v_{s,h} \in V_{s,h}(D_{\mathit{fl}})$, we set

$$\nabla_h v_{s,h}\big|_K := \nabla\Big[v_{s,h}\big|_K \Big]$$

for each cut cell mesh element $K \in \mathcal{T}_h(D_{\mathit{fl}})$. Evaluations of $\nabla_h v_{s,h}$ on internal faces of $\mathcal{T}_h(D_{\mathit{fl}})$ will not be required.

Remark 5.3.2. *Functions in the discrete spaces $V_{s,h}(D_{\mathit{fl}})$ are defined over the union of elements in the cut cell mesh $\mathcal{T}_h(D_{\mathit{fl}})$ and have a straightforward extension up to the whole active mesh $\hat{\mathcal{T}}_h(D_{\mathit{fl}})$. Although we will seek a discrete solution of the form $u_{s,h} \in V_{s,h}(D_{\mathit{fl}})$, we will only be interested in the values of $u_{s,h}$ over the reconstructed hypersurface $\Gamma_{\mathit{fl}}^{\mathrm{new}}$. Integrals will be computed over the reconstructed hypersurfaces $\Gamma_{\mathit{fl}}^{\mathrm{old}}$ and $\Gamma_{\mathit{fl}}^{\mathrm{new}}$, and also over the cut cell mesh $\mathcal{T}_h(D_{\mathit{fl}})$. The latter can be considered as being a geometrically unfitted mesh with respect to $\Gamma_{\mathit{fl}}^{\mathrm{old}}$ and $\Gamma_{\mathit{fl}}^{\mathrm{new}}$. This is a similar approach to Deckelnick et al. (2014), but different to Olshanskii et al. (2009). Unlike Olshanskii et al. (2009), we avoid difficulties in defining discrete spaces and constructing a natural basis for those spaces, cf. Remark 4.2.4.*

Discretization

We discretize the reformulated problem which is given by equation (5.13) in three steps. First, we replace the geometry $(\Gamma^{\mathrm{old}}, D, \Gamma^{\mathrm{new}})$ in equation (5.13) by its discrete reconstruction $\big(\Gamma_h^{\mathrm{old}}, D_h, \Gamma_h^{\mathrm{new}}\big)$, and the level set function Φ by its discrete analogue Φ_h. In the course of this, we consequently use $\nabla_h \Phi_h$ as a replacement for the classical gradient $\nabla\Phi$. At the same time, we replace the extended material velocity $\mathbf{v}_{\mathbf{s}}^{\mathbf{ext}} \colon D \to \mathbb{R}^d$ by some consistent discrete approximation $\mathbf{v}_{\mathbf{s},h}^{\mathbf{ext}} \colon D_h \to \mathbb{R}^d$ which we will take a closer look at in Section 5.3.3. As a second step, we transform the resulting equation by splitting its bulk integral over D_h into a sum of bulk integrals over the cut cells $K \in \mathcal{T}_h(D_h)$, and by integrating by parts on each cut cell K. Subsequently, we apply *upwind stabilization* (see e.g. Brezzi et al., 2004) by using the classical upwind numerical flux on the internal faces of $\mathcal{T}_h(D_h)$, similar to the approach in Bastian et al. (2011). In the context of DG methods for hyperbolic problems, upwinding is a well-known stabilization mechanism. Since we are dealing with a bulk advection problem according to Remark 5.3.1, we can expect that it leads to a host DG formulation suitable for the problem at hand. As a third step, we restrict the set of admissible functions by considering only functions that are representable using one of the discrete function spaces $V_{s,h}(D_h)$. This corresponds to replacing the extended solution variable u_s^{ext} by some discrete analogue $u_{s,h} \in V_{s,h}(D_h)$, while replacing the smooth test functions φ_s by discrete test functions $\varphi_{s,h} \in V_{s,h}(D_h)$.

As a result, we obtain the following UDG scheme for problem (5.9):

Scheme 5.3.3. *Given a suitable approximation $u_{s,h}^{\mathrm{old}} \in L^2\big(\Gamma_h^{\mathrm{old}}\big)$ of the surface field u_s^{old}, find a discrete function $u_{s,h} \in V_{s,h}(D_h)$, such that*

$$\int_{\Gamma_h^{\mathrm{new}}} u_{s,h}\, \varphi_{s,h} \,\mathrm{d}\sigma + \frac{\tau}{\gamma}\, a_s(u_{s,h}, \varphi_{s,h}) = \int_{\Gamma_h^{\mathrm{old}}} u_{s,h}^{\mathrm{old}}\, \varphi_{s,h} \,\mathrm{d}\sigma \qquad \forall \varphi_{s,h} \in V_{s,h}(D_h), \quad (5.14)$$

where $a_s \colon V_{s,h}(D_h) \times V_{s,h}(D_h) \to \mathbb{R}$ is a bilinear form which is defined along the lines of the classical upwind DG formulation. It is given by

$$a_s(u_{s,h}, \varphi_{s,h}) := a^{\mathrm{upwind}}\Big(D_h, u_{s,h}, \mathbf{v}_{\mathbf{s},h}^{\mathbf{ext}}\big|\nabla_h \Phi_h(\cdot, t^{\mathrm{new}})\big|, \varphi_{s,h}\Big), \quad (5.15)$$

with

$$a^{\mathrm{upwind}}\big(\mathcal{D}, u_h, \mathbf{w}, \varphi_h\big) := -\sum\nolimits_{K \in \mathcal{T}_h(\mathcal{D})} \int_K u_h \mathbf{w} \cdot \nabla_h \varphi_h \,\mathrm{d}x + \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})} \int_E u_h^{\uparrow} \{\mathbf{w} \cdot \mathbf{n}_{\mathbf{E}}\} [\![\varphi_h]\!] \,\mathrm{d}\sigma.$$

Here, we are dealing with a function u_h that is multiply defined on each internal

face $E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})$, and $u_h^{\uparrow}$ denotes the value of u_h on the upwind side of E with respect to the considered velocity field $\mathbf{w}$. For each $E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})$, this value is defined as

$$u_h^{\uparrow}\big|_E := \begin{cases} u_h|_{K_E^+} & \text{if } \{\mathbf{w}\cdot\mathbf{n_E}\} \geq 0, \\ u_h|_{K_E^-} & \text{if } \{\mathbf{w}\cdot\mathbf{n_E}\} < 0. \end{cases}$$

In special cases, it can happen that a section of $\Gamma_\hbar^{\mathrm{new}}$ or a section of $\Gamma_\hbar^{\mathrm{old}}$ is part of some internal face $E \in \mathcal{E}_h^{\mathrm{int}}(D_\hbar)$. On such sections, we consider the velocity field $\mathbf{w} = \mathbf{v}_{\mathbf{s},\hbar}^{\mathbf{ext}}\big|\nabla_\hbar\Phi_\hbar(\cdot,t^{\mathrm{new}})\big|$ while evaluating the first and the last integral in equation (5.14), and replace the integrand $u_{s,h}\,\varphi_{s,h}$ by $[u_{s,h}\,\varphi_{s,h}]^{\uparrow}$ and the integrand $u_{s,h}^{\mathrm{old}}\,\varphi_{s,h}$ by $[u_{s,h}^{\mathrm{old}}\,\varphi_{s,h}]^{\uparrow}$, respectively.

5.3.3. Remarks on choosing extended data functions

There are many different options regarding the choice of the extended material velocity $\mathbf{v}_{\mathbf{s}}^{\mathbf{ext}}\colon D \to \mathbb{R}^d$ and its discrete approximation $\mathbf{v}_{\mathbf{s},\hbar}^{\mathbf{ext}}\colon D_\hbar \to \mathbb{R}^d$.

In geometrical special cases which allow for assuming that the material velocity $\mathbf{v}_{\mathbf{s}}(\cdot,t^{\mathrm{new}})$ has no component tangential to $\Gamma(t^{\mathrm{new}})$, for instance, we might construct $\mathbf{v}_{\mathbf{s}}^{\mathbf{ext}}$ using the normal velocity of the level sets $\Gamma_l(t^{\mathrm{new}})$ of the continuous level set function $\Phi(\cdot,t^{\mathrm{new}})$. Referring to what has been discussed in Section 3.3.1, we can particularly set

$$\mathbf{v}_{\mathbf{s}}^{\mathbf{ext}} := \frac{-\big[\partial_t\Phi\big](\cdot,t^{\mathrm{new}})}{\big|\nabla\Phi(\cdot,t^{\mathrm{new}})\big|}\,\frac{\nabla\Phi(\cdot,t^{\mathrm{new}})}{\big|\nabla\Phi(\cdot,t^{\mathrm{new}})\big|}\bigg|_D.$$

Moreover, exploiting that it is possible to approximate the time derivative $\partial_t\Phi$ using a backward difference, we can extract an approximate normal velocity from the values $\Phi_\hbar(\cdot,t^{\mathrm{new}})$ and $\Phi_\hbar(\cdot,t^{\mathrm{old}})$ of the discrete level set function by setting

$$\mathbf{v}_{\mathbf{s},\hbar}^{\mathbf{ext}} := -\frac{1}{\tau}\frac{\Phi_\hbar(\cdot,t^{\mathrm{new}}) - \Phi_\hbar(\cdot,t^{\mathrm{old}})}{\big|\nabla_\hbar\Phi_\hbar(\cdot,t^{\mathrm{new}})\big|}\,\frac{\nabla_\hbar\Phi_\hbar(\cdot,t^{\mathrm{new}})}{\big|\nabla_\hbar\Phi_\hbar(\cdot,t^{\mathrm{new}})\big|}\bigg|_{D_\hbar}.$$

5.3.4. Global conservation properties

According to what we discussed in Section 5.2.2, exact solutions to problem (5.9) satisfy global conservation property (5.7b). Due to its construction, Scheme 5.3.3 recovers a discrete analogue to that property. This is the subject of the following theorem.

Theorem 5.3.4 (Discrete solutions – global conservation property)**.** *Each discrete solution $u_{s,h} \in V_{s,h}(D_\hbar)$ which is obtained using Scheme 5.3.3 satisfies*

$$\int_{\Gamma_\hbar^{\mathrm{new}}} u_{s,h}\big|_{\Gamma_\hbar^{\mathrm{new}}}\,\mathrm{d}\sigma = \int_{\Gamma_\hbar^{\mathrm{old}}} u_{s,h}^{\mathrm{old}}\,\mathrm{d}\sigma. \tag{5.16}$$

Here, the integrands shall be understood in the same way as in equation (5.14)*, particularly in the special case which is mentioned in Scheme 5.3.3. Property* (5.16) *is a discrete analogue to global conservation property* (5.7b)*.*

Proof. Property (5.16) can be derived in a straightforward way. In particular, the construction of Scheme 5.3.3 implies that the characteristic function $\mathbb{1}_{D_\hbar}$ of $D_\hbar$ is part of the discrete function spaces $V_{s,h}(D_\hbar)$ and hence an admissible test function $\varphi_{s,h}$ in equation (5.14). Moreover, we have $a_s\big(u_{s,h}, \mathbb{1}_{D_\hbar}\big) = 0$ for the bilinear form defined in equation (5.15). Therefore, testing with $\varphi_{s,h} = \mathbb{1}_{D_\hbar}$ in equation (5.14) yields property (5.16). □

We would like to emphasize that Theorem 5.3.4 has practical consequences regarding our operator splitting methods that were introduced in Section 5.2.2. When being employed to treat the surface equation in subproblems of type (5.5), Scheme 5.3.3 allows for obtaining overall schemes which are globally conservative. Please refer to Section 5.2.2 for details on this matter.

5.3.5. Understanding the scheme in one dimension

Next, we want to provide a better interpretation of the scheme and illustrate its limitations. We employ a simple 1d example for this purpose.

In particular, let $\Gamma(t)$ be an evolving 0-dimensional hypersurface that is moving along a one-dimensional axis with some constant material velocity $\mathbf{v_s} := v_s \in \mathbb{R}^{>0}$. In view of the single time step of size $\tau := t^{\text{new}} - t^{\text{old}}$ which we are considering, two points $x^{\text{old}} := \Gamma^{\text{old}} \in \mathbb{R}$ and $x^{\text{new}} := \Gamma^{\text{new}} = x^{\text{old}} + \tau v_s \in \mathbb{R}$ can be associated with $\Gamma(t)$ at the time-discrete level. Fixing an open interval $(x_0, x_M) \subset \mathbb{R}$ which contains $\Gamma(t)$ at all times t at the time-continuous level, we describe the geometry by choosing the level set domain $\Omega_\Phi := (x_0, x_M)$, together with the time-dependent level set function Φ that is defined by $\Phi(x,t) := (x - x^{\text{old}}) - (t - t^{\text{old}})v_s$.

For the geometrical setup and time step which we are considering, the static bulk domain D that has been defined in equation (5.11) takes the form

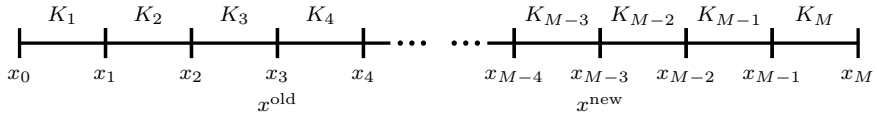
$$D = (x^{\text{old}}, x^{\text{new}}).$$

Accordingly, in view of the level set function Φ chosen above, the parameter γ which arises from the reformulation approach that has been introduced in Section 5.3.1 evaluates to

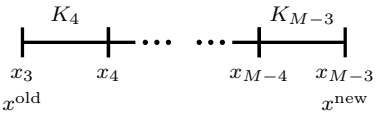
$$\gamma = \sup_D \Phi(\cdot, t^{\text{new}}) - \inf_D \Phi(\cdot, t^{\text{new}}) = x^{\text{new}} - x^{\text{old}} = \tau v_s. \tag{5.17}$$

By extending the material velocity $\mathbf{v_s}(\cdot, t^{\text{new}})$ in the reformulation approach using the $\Phi(\cdot, t^{\text{new}})$-based strategy which we introduced in Section 5.3.3, we obtain the extended material velocity $\mathbf{v_s^{ext}} \equiv v_s$. Note that $\mathbf{v_s}(\cdot, t^{\text{new}})$ has no component tangential to Γ^{new} since the hypersurface comprises a single point.

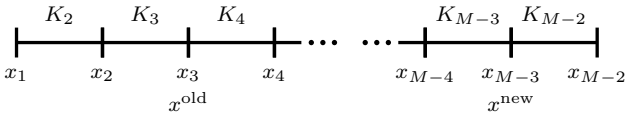
Let the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ be a decomposition of Ω_Φ into $M \in \mathbb{N}$ open subintervals $\{K_i := (x_{i-1}, x_i)\}_{i=1,\dots,M}$ of uniform width $h := \frac{x_M - x_0}{M}$,



(a) Fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$.



(b) Cut cell mesh $\mathcal{T}_h(D_{\mathit{fl}})$ of the static bulk domain D_{fl}.



(c) Cut cell mesh $\mathcal{T}_h(\mathfrak{D})$ of the larger bulk domain $\mathfrak{D} \supset D_{\mathit{fl}}$.

Figure 5.5.: Meshes which appear in the 1d example from Section 5.3.5.

where the number M is supposed to be large enough that the following considerations make sense. Moreover, let the geometry mesh not be a separate mesh, but rather be chosen as $\mathcal{T}_{\mathit{fl}}(\Omega_\Phi) := \mathcal{T}_h(\Omega_\Phi)$. Given our level set function Φ, the latter choice does not affect results that are obtained using Scheme 5.3.3, as we will see next.

The simplicity of the geometrical setup which we are considering in this example has the following practical implications. The level set function Φ and its discrete analogue Φ_{fl} coincide since $\Phi(\cdot, t)$ is a continuous linear function on $\mathrm{cl}\,(\Omega_\Phi)$ at each fixed time t. We hence have the equality

$$\left(\Gamma_{\mathit{fl}}^{\text{old}}, D_{\mathit{fl}}, \Gamma_{\mathit{fl}}^{\text{new}}\right) = (\Gamma^{\text{old}}, D, \Gamma^{\text{new}}),$$

i.e. the exact geometry matches its discrete reconstruction. Furthermore, we have $|\nabla_{\mathit{fl}}\Phi_{\mathit{fl}}| \equiv 1$ and the discrete approximation of the extended material velocity $\mathbf{v}_{\mathbf{s}}^{\mathbf{ext}}$ which we introduced in Section 5.3.3 takes the form $\mathbf{v}_{\mathbf{s},\mathit{fl}}^{\mathbf{ext}} \equiv v_s$.

In the following, we suppose that $u_{s,h}^{\text{old}}$ is a given scalar at x^{old}. In order to further simplify our theoretical considerations, let the points x^{old} and x^{new} be given in such a way that the cut cells in $\mathcal{T}_h(D_{\mathit{fl}})$ match full elements in the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$. In particular, let be $x^{\text{old}} = x_3$ and $x^{\text{new}} = x_{M-3}$, such that $\mathcal{T}_h(D_{\mathit{fl}}) = \hat{\mathcal{T}}_h(D_{\mathit{fl}}) = \{K_4, \ldots, K_{M-3}\} \subset \mathcal{T}_h(\Omega_\Phi)$. See Figure 5.5a and Figure 5.5b for illustrations.

Applying Scheme 5.3.3 in its standard version

For the sake of simplicity, we apply Scheme 5.3.3 using the discrete function space $V_{s,h}(D_{\mathfrak{h}})$ that is associated with polynomial degree $k = 0$, i.e. the piecewise constant discrete space. Doing so yields a finite volume type scheme. We may assume that the dedicated unit normal vector $\mathbf{n_E} := 1$ is assigned to each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(D_{\mathfrak{h}}) = \{x_4, \dots, x_{M-4}\}$ of the cut cell mesh $\mathcal{T}_h(D_{\mathfrak{h}})$, given that the particular choice at each internal face is arbitrary but fixed. Using this assumption and our knowledge about the polynomial degree k, equation (5.14) simplifies to

$$\big[u_{s,h}\,\varphi_{s,h}\big](x_{M-3}) + \frac{\tau v_s}{\gamma}\left(\sum_{m=4}^{M-4} u_{s,h}|_{K_m}\,[\![\varphi_{s,h}]\!](x_m)\right) = u_{s,h}^{\mathrm{old}}\,\varphi_{s,h}(x_3) \qquad \forall \varphi_{s,h} \in V_{s,h}(D_{\mathfrak{h}}). \quad (5.18)$$

By fixing a basis for $V_{s,h}(D_{\mathfrak{h}})$, we obtain a system of linear equations. As a straightforward choice, we employ the basis $\{\varphi_{s,h,i} := \mathbb{1}_{K_i}\}_{i=4,\dots,M-3}$ which consists of the characteristic functions of the elements in the cut cell mesh $\mathcal{T}_h(D_{\mathfrak{h}})$. Denoting the vector of unkowns by $\underline{\mathbf{u}} =: (\mathrm{u}_4, \dots, \mathrm{u}_{M-3})^{\mathrm{tr}}$, we wish to find a discrete function $u_{s,h} = \sum_{j=4}^{M-3} \mathrm{u}_j\,\varphi_{s,h,j}$ with

$$\begin{aligned} \tau v_s \gamma^{-1} \mathrm{u}_4 &= u_{s,h}^{\mathrm{old}}, \\ \tau v_s \gamma^{-1}(\mathrm{u}_i - \mathrm{u}_{i-1}) &= 0 \qquad && \text{for } i = 5, \dots, M-4, \\ \mathrm{u}_{M-3} + \tau v_s \gamma^{-1}(-\mathrm{u}_{M-4}) &= 0. \end{aligned}$$

This linear system results from choosing the basis functions $\{\varphi_{s,h,i}\}_{i=4,\dots,M-3}$ as test functions in equation (5.18). It is uniquely solvable, yielding a piecewise constant discrete solution $u_{s,h}$ with $u_{s,h}|_{K_i} = \mathrm{u}_i$ for $i = 4, \dots, M-3$, and

$$\begin{aligned} \mathrm{u}_i &= \gamma(\tau v_s)^{-1} u_{s,h}^{\mathrm{old}} \qquad && \text{for } i = 4, \dots, M-4, \\ \mathrm{u}_{M-3} &= u_{s,h}^{\mathrm{old}}. \end{aligned}$$

As a consequence, we have $u_{s,h}|_{\Gamma_{\mathfrak{h}}^{\mathrm{new}}} = u_{s,h}|_{K_{M-3}} = u_{s,h}^{\mathrm{old}}$. The discrete solution $u_{s,h} \in V_{s,h}(D_{\mathfrak{h}})$ which is obtained using Scheme 5.3.3 hence takes the correct value on $\Gamma_{\mathfrak{h}}^{\mathrm{new}}$. Please note that computing the exact value of the parameter γ in equation (5.17) is not crucial for gaining the correct solution in this example.

Applying Scheme 5.3.3 using a larger bulk domain

It is an interesting question if the construction of our scheme also allows for applying the scheme using some larger bulk domain. This would enable us to employ bulk domains of reduced geometrical complexity. Sticking with the

above setting, we now illustrate what happens if we replace the static bulk domain D_{fl} in Scheme 5.3.3 by a larger bulk domain $\mathfrak{D} \supset D_{\mathit{fl}}$.

For this purpose, we define an active mesh $\hat{\mathcal{T}}_h(\mathfrak{D})$, a cut cell mesh $\mathcal{T}_h(\mathfrak{D})$, its internal skeleton $\mathcal{E}_h^{\mathrm{int}}(\mathfrak{D})$, and discrete finite element spaces $V_{s,h}(\mathfrak{D})$ for some given bulk domain $\mathfrak{D}$ of this kind. Correspondingly, we define a dedicated unit normal vector $\mathbf{n_E}$ for every internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\mathfrak{D})$, two operators $[\![\cdot]\!]$ and $\{ \cdot \}$ which describe the jump and the average of functions on $\mathcal{E}_h^{\mathrm{int}}(\mathfrak{D})$, and a piecewise variant ∇_h of the (transposed) classical gradient operator. The definitions are analogous to those in Section 5.3.2.

In order to keep our theoretical considerations as simple as before, we choose $\mathfrak{D}$ in such a way that the cut cells in $\mathcal{T}_h(\mathfrak{D})$ match full elements in the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$. In particular, let be $\mathfrak{D} := (x_1, x_{M-2})$, such that we have $\mathcal{T}_h(\mathfrak{D}) = \hat{\mathcal{T}}_h(\mathfrak{D}) = \{K_2, \dots, K_{M-2}\} \subset \mathcal{T}_h(\Omega_\Phi)$, and the internal skeleton takes the form $\mathcal{E}_h^{\mathrm{int}}(\mathfrak{D}) = \{x_2, \dots, x_{M-3}\}$. See Figure 5.5a and Figure 5.5c for illustrations.

Proceeding as described above, with $\mathfrak{D}$ taking the role of the static bulk domain D_{fl}, equation (5.14) simplifies to

$$\big[u_{s,h}\,\varphi_{s,h}\big]^{\uparrow}(x_{M-3}) + \frac{\tau v_s}{\gamma}\left(\sum_{m=2}^{M-3} u_{s,h}|_{K_m}\,[\![\varphi_{s,h}]\!](x_m)\right) = u_{s,h}^{\mathrm{old}}\,\varphi_{s,h}^{\uparrow}(x_3) \qquad \forall \varphi_{s,h} \in V_{s,h}(\mathfrak{D}), \quad (5.19)$$

where the two remaining upwind terms at the beginning and at the end of the equation result from the special case which is mentioned in Scheme 5.3.3. Please note that $\Gamma_{\mathit{fl}}^{\mathrm{new}} = x^{\mathrm{new}} = x_{M-3}$ and $\Gamma_{\mathit{fl}}^{\mathrm{old}} = x^{\mathrm{old}} = x_3$ correspond to some internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\mathfrak{D})$.

We continue as described above and obtain a system of linear equations by choosing the set $\{\varphi_{s,h,i} := \mathbb{1}_{K_i}\}_{i=2,\dots,M-2}$ of characteristic functions of the elements in $\mathcal{T}_h(\mathfrak{D})$ as a basis for $V_{s,h}(\mathfrak{D})$, and by using each such basis function as a test function in equation (5.19). Denoting the vector of unkowns by $\underline{\mathbf{u}} =: (\mathrm{u}_2, \dots, \mathrm{u}_{M-2})^{\mathrm{tr}}$, we seek a discrete function $u_{s,h} = \sum_{j=2}^{M-2} \mathrm{u}_j\,\varphi_{s,h,j}$ with

$$\begin{aligned}
\tau v_s \gamma^{-1} \mathrm{u}_2 &= 0, \\
\tau v_s \gamma^{-1} (\mathrm{u}_3 - \mathrm{u}_2) &= u_{s,h}^{\mathrm{old}}, \\
\tau v_s \gamma^{-1} (\mathrm{u}_i - \mathrm{u}_{i-1}) &= 0 \qquad\qquad \text{for } i = 4, \dots, M-4, \\
\mathrm{u}_{M-3} + \tau v_s \gamma^{-1} (\mathrm{u}_{M-3} - \mathrm{u}_{M-4}) &= 0, \\
\tau v_s \gamma^{-1} (-\mathrm{u}_{M-3}) &= 0.
\end{aligned}$$

Reformulating this linear system shows that we look for a piecewise constant discrete function $u_{s,h}$ with $u_{s,h}|_{K_i} = \mathrm{u}_i$ for $i = 2, \dots, M-2$, and

$$\begin{aligned} \mathrm{u}_2 &= 0, \\ \mathrm{u}_i &= \gamma(\tau v_s)^{-1} u_{s,h}^{\mathrm{old}} \qquad \text{for } i = 3, \dots, M-4, \\ \mathrm{u}_{M-3} &= (1 + \tau v_s \gamma^{-1})^{-1} u_{s,h}^{\mathrm{old}}, \\ \mathrm{u}_{M-3} &= 0. \end{aligned}$$

The latter formulation reveals that the linear system is not uniquely solvable, in general, although it is exactly determined having the same number of equations and unknowns. While the last two equations both determine u_{M-3} and yield an inconsistency if $u_{s,h}^{\mathrm{old}} \neq 0$, the unknown u_{M-2} may take any value.

We note that replacing the last equation by an equation which determines u_{M-2} would render the problem well-posed. However, the discrete analogue to global conservation property (5.7b), which is recovered by the standard version of Scheme 5.3.3 according to Theorem 5.3.4, would still be spoiled. Even using the exact value of the parameter γ that is computed in equation (5.17), we would obtain a discrete solution $u_{s,h} \in V_{s,h}(\mathfrak{D})$ which takes the wrong value on $\Gamma_{\hbar}^{\mathrm{new}}$. We would specifically get $u_{s,h}|_{\Gamma_{\hbar}^{\mathrm{new}}} = u_{s,h}|_{K_{M-3}} = \mathrm{u}_{M-3} = \frac{1}{2} u_{s,h}^{\mathrm{old}}$.

It should be mentioned that both problems are exclusively linked to extending the solution in the direction of transport. Looking upwind, the solution appears to be extended in a suitable manner by the constant value of 0. Hence, if we want to replace the static bulk domain $D_{\hbar}$ in Scheme 5.3.3 by some larger bulk domain $\mathfrak{D} \supset D_{\hbar}$, we are either bound to employ some domain $\mathfrak{D}$ whose boundary resolves $\Gamma_{\hbar}^{\mathrm{new}} \cap \partial D_{\hbar}$, or we require modifications which tackle the two problems mentioned above.

5.3.6. Numerical results

To investigate the convergence properties of Scheme 5.3.3, we implemented the scheme in C++ using the Distributed and Unified Numerics Environment (`DUNE`)[1]. With this implementation, we performed numerical studies which will be presented in the remainder of this section.

Details on the implementation and on how the code can be obtained to reproduce the results which we present in the following can be found in Appendix A.3.

Analytical test problem: constant initial values on a circle shrinking with constant normal speed

Fixing a time step with $t^{\mathrm{old}} := 0$ and $t^{\mathrm{new}} := 0.5 = \tau$, we consider a circle $\Gamma(t)$ in $\mathbb{R}^2$ of initial radius 1.0, which is centered at the origin and shrinking with constant normal velocity given by the choice $\mathbf{v_s} := -\boldsymbol{\nu}$. Using the level set

[1] `https://www.dune-project.org`

framework, this circle $\Gamma(t)$ can be represented as the zero level set of the time-dependent level set function Φ that is defined by $\Phi(\underline{\mathbf{x}}, t) := |\underline{\mathbf{x}}| - (1.0 - t)$, with $\underline{\mathbf{x}} \in \mathbb{R}^2$ and $t \in [0, T]$. The latter is a signed distance function, i.e. $|\nabla\Phi| \equiv 1$.

For the geometrical setup and time step which we are considering, the static bulk domain D that has been defined in equation (5.11) takes the form

$$D = \{\underline{\mathbf{x}} \in \mathbb{R}^2 \mid 0.5 < |\underline{\mathbf{x}}| < 1.0\}. \tag{5.20}$$

Therefore, in view of the level set function Φ defined above, the parameter γ which arises from the reformulation approach that has been introduced in Section 5.3.1 evaluates to

$$\gamma = \sup_D \Phi(\cdot, t^{\text{new}}) - \inf_D \Phi(\cdot, t^{\text{new}}) = 0.5.$$

Given the shrinking circle $\Gamma(t)$, we consider problem (5.9) with constant initial values $u_s(\cdot, t^{\text{old}}) \equiv 1$. In view of the conservation law which is represented by problem (5.9), i.e. the one that results in global conservation property (5.7b) when looking at the full set $\Gamma(t)$, it is clear that the corresponding analytical solution is constant with respect to each fixed time $t \in [t^{\text{old}}, t^{\text{new}}]$. Particularly considering its values on $\Gamma(t^{\text{new}})$, we have $u_s(\cdot, t^{\text{new}}) \equiv 2$.

Common simulation parameters

In our numerical studies, we use the following simulation parameters and related choices, unless otherwise stated.

The geometrical setup which we investigate in each simulation is the one which is considered in the analytical test problem defined above. The geometry is described choosing the level set domain $\Omega_\Phi := (-1.5, 2) \times (-1.5, 1.5)$ and the time-dependent level set function Φ that is associated with our test problem. Always taking into account the test problem's fixed time step, we furthermore use the annular bulk domain D given by equation (5.20), together with the corresponding parameter $\gamma = 0.5$.

To perform the reformulation approach which is described in Section 5.3.1, we extend the material velocity $\mathbf{v_s}(\cdot, t^{\text{new}})$ by choosing the extended material velocity $\mathbf{v_s^{ext}}$ which we introduced in Section 5.3.3. Note that $\mathbf{v_s}$ is purely normal to $\Gamma(t)$ in case of the geometrical setup which we are considering.

For UDG discretization as described in Section 5.3.2, we employ structured fundamental meshes $\mathcal{T}_h(\Omega_\Phi)$ and structured geometry meshes $\mathcal{T}_\hbar(\Omega_\Phi)$ with varying mesh widths h and $\hbar$, respectively, that decompose Ω_Φ into an equal number of quadrilaterals in x_0-direction and in x_1-direction. To obtain each fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$, we construct an initial mesh comprising five quadrilaterals in each direction and perform a certain number $\mathfrak{r} \in \mathbb{N}$ of uniform mesh refinements. For the sake of simplicity, we do not use separate geometry meshes $\mathcal{T}_\hbar(\Omega_\Phi)$, but rather choose $\mathcal{T}_\hbar(\Omega_\Phi) := \mathcal{T}_h(\Omega_\Phi)$. Discrete reconstructions $D_\hbar$ of the bulk domain D for different values of h can be seen in Figure 5.6.

Please note the asymmetry in x_0-direction which is induced by the choice of the level set domain Ω_Φ.

On each fundamental mesh, we choose a discrete space $V_{s,h}(D_{\mathfrak{h}})$ that locally (i.e. on each cut cell K) resembles $\mathbb{P}(K) := P_k(K)$, the space of polynomial functions of total degree less than or equal to some $k \in \mathbb{N}$ over the domain K. For its construction, we use monomial basis functions (i.e. the set $\{1\}$ for $k = 0$, the set $\{1, \mathrm{x}_0, \mathrm{x}_1\}$ for $k = 1$, and the set $\{1, \mathrm{x}_0, \mathrm{x}_1, \mathrm{x}_0\mathrm{x}_1, \mathrm{x}_0^2, \mathrm{x}_1^2\}$ for $k = 2$) on the reference element of the fundamental mesh elements. As polynomial degree, we employ $k = 0$ for the sake of simplicity.

Error measures

In order to analyze the convergence of Scheme 5.3.3, we compute *absolute errors*

$$\mathcal{E}_{s,L^1(\Gamma_{\mathfrak{h}}^{\mathrm{new}})}(h) := \left\| u_s(\cdot, t^{\mathrm{new}}) - u_{s,h} \right\|_{L^1(\Gamma_{\mathfrak{h}}^{\mathrm{new}})} , \tag{5.21a}$$

$$\mathcal{E}_{s,L^2(\Gamma_{\mathfrak{h}}^{\mathrm{new}})}(h) := \left\| u_s(\cdot, t^{\mathrm{new}}) - u_{s,h} \right\|_{L^2(\Gamma_{\mathfrak{h}}^{\mathrm{new}})} , \tag{5.21b}$$

$$\mathcal{E}_{s,L^\infty(\Gamma_{\mathfrak{h}}^{\mathrm{new}})}(h) := \left\| u_s(\cdot, t^{\mathrm{new}}) - u_{s,h} \right\|_{L^\infty(\Gamma_{\mathfrak{h}}^{\mathrm{new}})} , \tag{5.21c}$$

with respect to an a priori known analytical solution u_s on $\Gamma(t^{\mathrm{new}})$. Here, the function $u_{s,h} \in V_{s,h}(D_{\mathfrak{h}})$ again denotes the numerical solution which is obtained using Scheme 5.3.3.

Assuming that the absolute errors defined in equations (5.21) satisfy estimates of the form $\mathcal{E}(h) \in \mathcal{O}(h^z)$, where $z \in \mathbb{R}^{>0}$ is known as the order of convergence with respect to $\mathcal{E}(h)$, we compute experimental orders of convergence as defined in Section 4.3.1.

Numerical convergence study: h-refinement

First, we investigate errors and associated experimental orders of convergence with respect to the width h of the fundamental mesh. As with our numerical studies for linear elliptic model problems with static bulk domains in Section 4.1.2, we employ h-refinement to do this. In particular, we perform computations while considering a sequence of fundamental meshes with $h \to 0$, which results from refinement of some coarse initial mesh.

In our numerical study, we consider the analytical test problem defined above. Correspondingly, we choose discrete initial values $u_{s,h}^{\mathrm{old}} = 1$ on $\Gamma_{\mathfrak{h}}^{\mathrm{old}}$. The associated numerical solution $u_{s,h}$ on $\Gamma_{\mathfrak{h}}^{\mathrm{new}}$ should have properties similar to those of the analytical solution. Results that are obtained using Scheme 5.3.3 are shown in Figure 5.6. As desired, the function values of $u_{s,h}$ on $\Gamma_{\mathfrak{h}}^{\mathrm{new}}$ appear to converge to the correct value of 2 as $h \to 0$.

Actual errors are listed in Table 5.1 and visualized in Figure 5.7. From those errors, it can be observed that the scheme exhibits convergence with respect to all three norms which we are considering. The sequence of numerical solutions

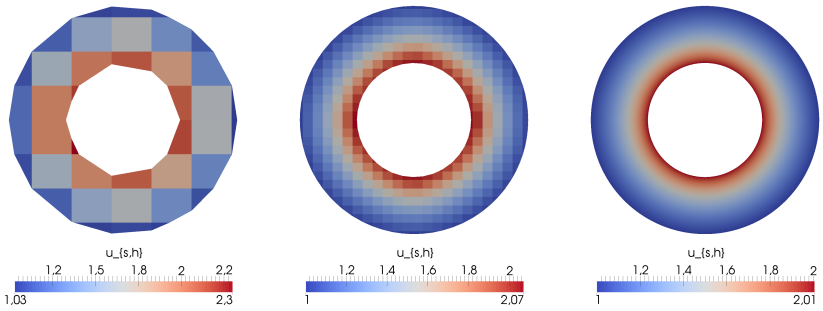


Figure 5.6.: Numerical solutions of the shrinking circle test problem, obtained using Scheme 5.3.3 with $k = 0$, and no separate geometry meshes. Note that, although the full solution $u_{s,h}$ is shown, we are only interested in its values over the reconstructed hypersurface Γ_h^{new}. The depicted sequence of numerical solutions corresponds to the cases $\#\text{cells} = 10^2, 40^2, 640^2$ in Table 5.1.

$\mathfrak{r}$	#cells	h	$\mathcal{E}_{s,L^1(\Gamma_h^{\text{new}})}$	eoc	$\mathcal{E}_{s,L^2(\Gamma_h^{\text{new}})}$	eoc	$\mathcal{E}_{s,L^\infty(\Gamma_h^{\text{new}})}$	eoc	$\int_{\Gamma_h^{\text{new}}} u_{s,h}$
0	5^2	$9.22 \cdot 10^{-1}$	$1.15 \cdot 10^{0}$	–	$9.08 \cdot 10^{-1}$	–	$1.00 \cdot 10^{0}$	–	6.10369
1	10^2	$4.61 \cdot 10^{-1}$	$3.17 \cdot 10^{-1}$	1.86	$2.20 \cdot 10^{-1}$	2.05	$3.00 \cdot 10^{-1}$	1.74	6.24228
2	20^2	$2.31 \cdot 10^{-1}$	$1.95 \cdot 10^{-1}$	0.7	$1.20 \cdot 10^{-1}$	0.87	$1.04 \cdot 10^{-1}$	1.53	6.27299
3	40^2	$1.15 \cdot 10^{-1}$	$1.07 \cdot 10^{-1}$	0.87	$7.18 \cdot 10^{-2}$	0.75	$8.25 \cdot 10^{-2}$	0.34	6.28064
4	80^2	$5.76 \cdot 10^{-2}$	$5.73 \cdot 10^{-2}$	0.89	$3.93 \cdot 10^{-2}$	0.87	$6.63 \cdot 10^{-2}$	0.32	6.28255
5	160^2	$2.88 \cdot 10^{-2}$	$3.08 \cdot 10^{-2}$	0.9	$2.13 \cdot 10^{-2}$	0.88	$3.72 \cdot 10^{-2}$	0.83	6.28303
6	320^2	$1.44 \cdot 10^{-2}$	$1.59 \cdot 10^{-2}$	0.95	$1.08 \cdot 10^{-2}$	0.98	$2.02 \cdot 10^{-2}$	0.88	6.28315
7	640^2	$7.20 \cdot 10^{-3}$	$8.06 \cdot 10^{-3}$	0.98	$5.49 \cdot 10^{-3}$	0.98	$1.01 \cdot 10^{-2}$	1.01	6.28318

Table 5.1.: Errors and amount of the considered quantity in numerical solutions of the shrinking circle test problem, obtained using Scheme 5.3.3 with $k = 0$, and no separate geometry meshes.

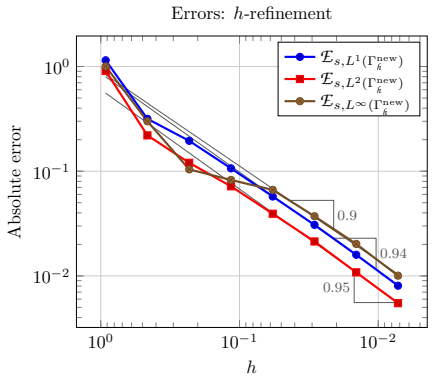


Figure 5.7.: Errors in numerical solutions of the shrinking circle test problem, obtained using Scheme 5.3.3 with $k = 0$, and no separate geometry meshes.

$(u_{s,h})_h$ converges to the analytical solution $u_s(\cdot, t^{\text{new}}) \equiv 2$ with order 1 in $\|\cdot\|_{L^1(\Gamma_h^{\text{new}})}$ and in $\|\cdot\|_{L^2(\Gamma_h^{\text{new}})}$, respectively, slightly affected by geometrical errors for coarse meshes. The convergence rate with respect to $\|\cdot\|_{L^\infty(\Gamma_h^{\text{new}})}$ is not as obvious, but also seems to be approaching order 1 for small values of h. Please note that #cells in Table 5.1 is the number of cells in the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$. It does not refer to the number of cut cells $K \in \mathcal{T}_h(D_h)$.

In addition to errors, Table 5.1 furthermore lists the amount of the quantity which we are considering in problem (5.9). Since Scheme 5.3.3 is globally conservative up to machine precision according to Theorem 5.3.4, the same amount is obtained both from the quantity's concentration $u_{s,h}^{\text{old}}$ on Γ_h^{old} and from the quantity's concentration $u_{s,h}$ on Γ_h^{new}. It should be mentioned that it is an indicator for the geometrical error coming from the reconstruction Γ_h^{old}. The latter determines the amount of quantity entering the discrete system. For the continuous problem with initial values $u_s(\cdot, t^{\text{old}}) \equiv 1$, this amount equals $2\pi \approx 6.283185$, which is nicely reproduced by our numerical results for small values of h.

Simulation study: Binary initial values on a shrinking circle

To illustrate the effect of non-constant initial values, we next perform the same simulations as above, but use initial values that are piecewise constant with a binary nature. More precisely, we choose initial values

$$u_s(\underline{\mathbf{x}}, t^{\text{old}}) := \begin{cases} 1 & \text{if } \ 0.2\pi < \operatorname{arctan2}(\mathrm{x}_1, \mathrm{x}_0) < 0.4\pi, \\ 0 & \text{else,} \end{cases} \quad \text{with} \quad \underline{\mathbf{x}} \in \Gamma(t^{\text{old}}), \tag{5.22}$$

where we employ the function $\operatorname{arctan2}(\mathrm{x}_1, \mathrm{x}_0)$ defined in Remark 4.3.1 to obtain the azimuthal angle ϕ of polar coordinates (r, ϕ). In view of the conservation law which is represented by problem (5.9), i.e. the one that results in global conservation property (5.7b) when looking at the full set $\Gamma(t)$, it is clear that the associated analytical solution on $\Gamma(t^{\text{new}})$ takes the form

$$u_s(\underline{\mathbf{x}}, t^{\text{new}}) = \begin{cases} 2 & \text{if } \ 0.2\pi < \operatorname{arctan2}(\mathrm{x}_1, \mathrm{x}_0) < 0.4\pi, \\ 0 & \text{else,} \end{cases} \quad \text{with} \quad \underline{\mathbf{x}} \in \Gamma(t^{\text{new}}). \tag{5.23}$$

When using corresponding discrete initial values $u_{s,h}^{\text{old}}$ on Γ_h^{old}, the associated numerical solution $u_{s,h}$ on Γ_h^{new} should have properties similar to those of the analytical solution. Results that are obtained using Scheme 5.3.3 are shown in Figure 5.8. As desired, the function values of $u_{s,h}$ on Γ_h^{new} exhibit a binary nature similar to the analytical solution which is given by equation (5.23). The reconstructed hypersurface Γ_h^{new} can be clustered in one region of points associated with vanishing function values and one region of points that are associated with function values greater than zero. Meanwhile, function values greater than zero appear to converge to the correct value of 2 as $h \to 0$.

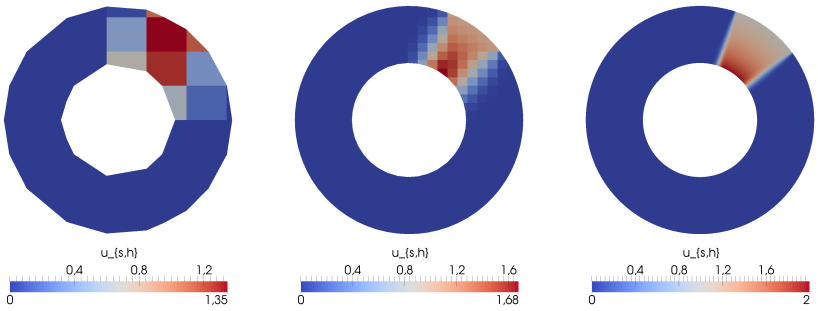


Figure 5.8.: Numerical solutions of problem (5.9) with binary initial values on a shrinking circle, obtained using Scheme 5.3.3 with $k = 0$, and no separate geometry meshes. As with Figure 5.6, the depicted sequence of numerical solutions corresponds to simulations with $\#\text{cells} = 10^2, 40^2, 640^2$.

However, we observe that our numerical solutions do not exactly represent the two discontinuities of the analytical solution. In each numerical solution $u_{s,h}$, the latter discontinuities are blurred and result in two diffuse jumps. These jumps sharpen for small values of h. This shows the effect of what is known as *numerical diffusion*. Numerical diffusion is a typical phenomenon that can be observed when using numerical methods for hyperbolic conservation laws.

Please note that the initial values which we are considering can be seen as a worst case scenario within the scope of our geometrical setup. Numerical diffusion is expected to have the largest possible impact since the transport velocity at each discontinuous front is not aligned with any cut cell boundary which touches the front. It should furthermore be mentioned that numerical diffusion can be reduced by using higher order methods with flux limiters.

Simulation study: Binary initial values on a rotating shrinking circle

Finally, we want to look at the effect of a material velocity with an additional tangential component. For this purpose, we repeat the above simulations with binary initial values (5.22), this time choosing an enriched material velocity $\mathbf{v_s} := -\boldsymbol{\nu} + \mathbf{v_{s,tan}}$, where $\mathbf{v_{s,tan}}(\cdot, t)\colon \Gamma(t) \to \mathbb{R}^2$ is some field of unit vectors tangential to $\Gamma(t)$. Particularly applying the field $\mathbf{v_{s,tan}}(\underline{x}, t) := \frac{1}{|\underline{x}|}(x_1, -x_0)^{\mathrm{tr}}$, the enriched material velocity $\mathbf{v_s}$ still corresponds to a shrinking circle, but the circle additionally rotates clockwise around its center.

Regarding the reformulation approach which is described in Section 5.3.1, we employ the strategy introduced in Section 5.3.3, as before, to extend the normal component of $\mathbf{v_s}(\cdot, t^{\mathrm{new}})$. For the tangential component of $\mathbf{v_s}(\cdot, t^{\mathrm{new}})$, we furthermore use the field $\mathbf{v^{ext}_{s,tan}}\colon D \to \mathbb{R}^2$ with $\mathbf{v^{ext}_{s,tan}}(\underline{x}) := \frac{1}{|\underline{x}|}(x_1, -x_0)^{\mathrm{tr}}$.

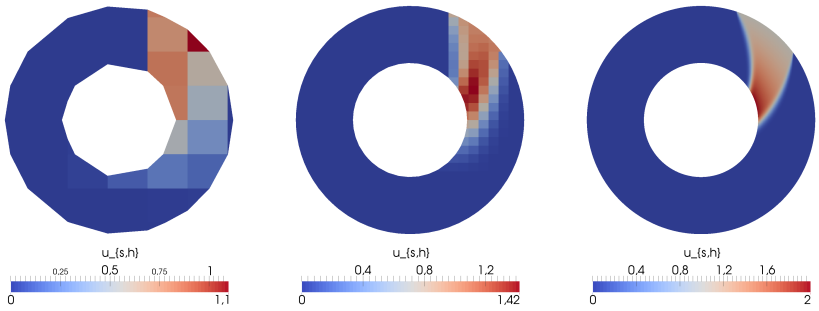


Figure 5.9.: Numerical solutions of problem (5.9) with binary initial values on a rotating and shrinking circle, obtained using Scheme 5.3.3 with $k = 0$, and no separate geometry meshes. As with Figure 5.6, the depicted sequence of numerical solutions corresponds to simulations with $\#\text{cells} = 10^2, 40^2, 640^2$.

The overall extended material velocity $\mathbf{v}_{\mathbf{s}}^{\mathbf{ext}} \colon D \to \mathbb{R}^2$ then takes the form $\mathbf{v}_{\mathbf{s}}^{\mathbf{ext}}(\underline{\mathbf{x}}) := -\frac{\underline{\mathbf{x}}}{|\underline{\mathbf{x}}|} + \mathbf{v}_{\mathbf{s,tan}}^{\mathbf{ext}}(\underline{\mathbf{x}})$.

Results that are obtained using Scheme 5.3.3 are shown in Figure 5.9. It can be observed that each numerical solution $u_{s,h}$ on Γ_h^{new} has similar properties as the corresponding one in the above simulations with binary initial values (5.22). In particular, it has the same binary nature and function values greater than zero seem to converge to the correct value of 2 as $h \to 0$. Moreover, due to numerical diffusion (which should be diminished when using higher order methods with flux limiters), the two discontinuities that would appear in an analytical solution get blurred and result in two diffuse jumps in $u_{s,h}$ which sharpen for small values of h. With the enriched material velocity, each numerical solution $u_{s,h}$ is rotated clockwise around the center of the shrinking circle, as desired.

5.4. Discussion

In this chapter, we dealt with the question how the approaches from Chapter 4 can be extended to obtain UDG schemes for an example class of bulk–surface models comprising continuity equations on evolving geometries. We introduced an operator splitting approach which allows for recovering discrete analogues to the conservation properties that are embedded in the considered type of continuity equations, while disassembling the overall problem into a sequence of subproblems of two different types. Both types correspond to special cases of the considered class of bulk–surface models. Accordingly, the corresponding subproblems are easier to handle than the overall problem and they can be treated with specialized numerical schemes.

Subproblems of the first type are static geometry problems which can be treated with the UDG schemes introduced in Chapter 4. As a first step toward UDG schemes for subproblems of the second type, we introduced a UDG scheme which can be used for the surface part of those subproblems, namely conservative material transport driven by geometrical evolution of a hypersurface. Although dealing with continuity equations on evolving geometries, the scheme does not require space–time meshes. It is globally conservative and yields promising first numerical results for a selected geometrical special case.

5.4.1. Future perspectives

Generalizing Scheme 5.3.3

The numerical scheme which has been presented in Section 5.3 requires exact knowledge of the domain $D_{\hbar}$. Unfortunately, the latter can not be constructed in a fully automated manner. Only if the material velocity $\mathbf{v_s}$ has a dominant component normal to $\Gamma(t)$ at each point, $D_{\hbar}$ can be constructed from the values of the level set function $\Phi_{\hbar}$, evaluated at $t = t^{\mathrm{old}}$ and $t = t^{\mathrm{new}}$ in each time step. This is the case, e.g., in the setting in Figure 5.3, whereas in the setting in Figure 5.2 both the northern tip and the southern tip of the circle have a material velocity which is purely tangential to $\Gamma(t)$. The latter results in the two triangle-like parts of the domain $D_{\hbar}$ depicted in the right image in Figure 5.2. Reconstructing those parts requires knowledge of all intermediate curves $\Gamma(t)$, i.e., information which is not provided by $\Phi_{\hbar}(\cdot, t^{\mathrm{old}})$ and $\Phi_{\hbar}(\cdot, t^{\mathrm{new}})$ alone.

When designing Scheme 5.3.3, we originally aimed at a scheme that employs an unfitted bulk domain which only *contains* the domain $D_{\hbar}$. As we have seen in Section 5.3.5, directly employing such a larger bulk domain $\mathfrak{D} \supset D_{\hbar}$ in our scheme unfortunately does not result in a well-posed problem in general. Unique solvability requires that the boundary of $\mathfrak{D}$ resolves $\Gamma_{\hbar}^{\mathrm{new}} \cap \partial D_{\hbar}$. If $\mathfrak{D}$ does not meet this requirement, we can furthermore no longer guarantee that the scheme recovers conservation property (5.16).

However, we have good evidence that a generalized scheme which allows for utilizing either the domain $\mathcal{D} := D_{\hbar}$ or some easily-constructible larger bulk domain $\mathcal{D} := \mathfrak{D} \supset D_{\hbar}$ is available. In first tests, both issues encountered in Section 5.3.5 could be cured by two minor modifications of equation (5.14), which can be expressed by replacing the term $a^{\mathrm{upwind}}\big(\mathcal{D}, u_h, \mathbf{w}, \varphi_h\big)$ in equation (5.15) by a modified term

$$\begin{aligned} a^{\mathrm{upwind},*}\big(\mathcal{D}, u_h, \mathbf{w}, \varphi_h\big) := &- \sum\nolimits_{K \in \mathcal{T}_h(\mathcal{D})} \int_K u_h \mathbf{w} \cdot \nabla_h \varphi_h \,\mathrm{d}x \\ &+ \sum\nolimits_{E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D}) \setminus \mathcal{E}_h^{\mathrm{int}}\left(\mathcal{N}(\Gamma_{\hbar}^{\mathrm{new}} \cap \partial D_{\hbar})\right)} \int_E u_h^{\uparrow}\{\mathbf{w} \cdot \mathbf{n_E}\}[\![\varphi_h]\!] \,\mathrm{d}\sigma \\ &+ \int_{\mathcal{M}} u_h (\mathbf{w} \cdot \mathbf{n}_{\partial \mathcal{D}}) \varphi_h \,\mathrm{d}\sigma. \end{aligned} \tag{5.24}$$

This modified term contains an additional integral over a boundary section given by

$$\mathcal{M} := \big\{ \underline{\mathbf{x}} \in \partial\mathcal{D} \setminus (\Gamma_{\hbar}^{\mathrm{new}} \cap \partial\mathcal{D}) \;\big|\; (\mathbf{w} \cdot \mathbf{n}_{\partial\mathcal{D}})(\underline{\mathbf{x}}) > 0 \big\}.$$

Moreover, in comparison with the set of internal faces that are considered in the term $a^{\mathrm{upwind}}\big(\mathcal{D}, u_h, \mathbf{w}, \varphi_h\big)$ in equation (5.15), the modified term (5.24) ignores all members of the subset

$$\mathcal{E}_h^{\mathrm{int}}\big(\mathcal{N}(\Gamma_{\hbar}^{\mathrm{new}} \cap \partial D_{\hbar})\big) := \big\{ E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D}) \;\big|\; \mathrm{meas}_{\mathbb{R}^{d-1}}\big((K_E^{\uparrow} \cup E) \cap \Gamma_{\hbar}^{\mathrm{new}}\big) > 0,\ \mathrm{meas}_{\mathbb{R}^{d-1}}\big(K_E^{\downarrow} \cap \Gamma_{\hbar}^{\mathrm{new}}\big) = 0 \big\}.$$

Here, for each internal face $E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})$, $K_E^{\uparrow} \in \mathcal{T}_h(\mathcal{D})$ and $K_E^{\downarrow} \in \mathcal{T}_h(\mathcal{D})$ denote the cut cell mesh element on the upwind side of E and on the downwind side, respectively, with respect to the considered velocity field $\mathbf{w}$. For each $E \in \mathcal{E}_h^{\mathrm{int}}(\mathcal{D})$, they are defined as the components of the ordered pair

$$(K_E^{\uparrow}, K_E^{\downarrow}) := \begin{cases} (K_E^{+}, K_E^{-}) & \text{if } \big\{\mathbf{w} \cdot \mathbf{n}_{\mathbf{E}}\big\} \geq 0, \\ (K_E^{-}, K_E^{+}) & \text{if } \big\{\mathbf{w} \cdot \mathbf{n}_{\mathbf{E}}\big\} < 0. \end{cases}$$

Not considering the subset $\mathcal{E}_h^{\mathrm{int}}\big(\mathcal{N}(\Gamma_{\hbar}^{\mathrm{new}} \cap \partial D_{\hbar})\big)$ aims at restoring property (5.16) by removing the coupling between suitable pairs of cut cells in some neighborhood of $\Gamma_{\hbar}^{\mathrm{new}} \cap \partial D_{\hbar}$. Similar to following the trajectories of material points in $\mathcal{D}$ with respect to the velocity field $\mathbf{w}$ and capturing internal faces closest to $\Gamma_{\hbar}^{\mathrm{new}} \cap \partial D_{\hbar}$, we search for internal faces whose adjacent cut cell on the upwind side intersects or touches $\Gamma_{\hbar}^{\mathrm{new}}$ and whose adjacent cut cell in the direction of transport does not intersect $\Gamma_{\hbar}^{\mathrm{new}}$. By ignoring these internal faces in the 1d example from Section 5.3.5, for instance, the upper bound of summation in the term $\tau v_s \gamma^{-1} \sum_{m=2}^{M-3} u_{s,h}|_{K_m} [\![\varphi_{s,h}]\!](x_m)$ in equation (5.19) is replaced by $M-4$. This removes the two terms of the form $\pm\tau v_s \gamma^{-1} \mathrm{u}_{M-3}$ in the associated system of linear equations. As a result, we obtain a single linear equation determining the unknown u_{M-3}, which takes the desired form $\mathrm{u}_{M-3} = u_{s,h}^{\mathrm{old}}$.

Adding the integral over the boundary section $\mathcal{M}$ corresponds to imposing an outflow boundary condition on $\mathcal{M}$. The latter restores unique solvability of the resulting system of linear equations. Together with the above restriction on the set of internal faces, the discrete solution gets extended in the direction of transport by the constant value of 0. In the 1d example from Section 5.3.5, for instance, the last linear equation which results from equation (5.19) changes to $\tau v_s \gamma^{-1} \mathrm{u}_{M-2} = 0$. This determines the unknown u_{M-2}, precisely giving $\mathrm{u}_{M-2} = 0$. Please note that it is important to exclude the set $\Gamma_{\hbar}^{\mathrm{new}} \cap \partial\mathcal{D}$ from $\mathcal{M}$ since the effect of the outflow boundary condition on $\Gamma_{\hbar}^{\mathrm{new}} \cap \partial\mathcal{D}$ would spoil property (5.16), particularly in the case $\mathcal{D} := D_{\hbar}$.

In our experience, the generalized scheme which results from replacing the term $a^{\mathrm{upwind}}\big(\mathcal{D}, u_h, \mathbf{w}, \varphi_h\big)$ in equation (5.15) by the modified term (5.24)

works as intended. Choosing some suitable narrow band around Γ_h^{new} as a bulk domain $\mathcal{D} := \mathfrak{D} \supset D_h$ that can be constructed easily, we successfully tested the geometrical setups considered in this chapter, including the one in Figure 5.2. Nevertheless, we would like to emphasize that the approach requires further investigation to make sure that it is really general enough.

Practical application of Scheme 5.3.3 or of some generalization

Given its favorable properties and the promising results in Section 5.3.6, Scheme 5.3.3 should be tested further by performing numerical convergence studies with multiple time steps. For this purpose, test problems with a known analytical solution are sufficient again. Provided that studies with multiple time steps show positive results, it is justifiable to subsequently employ the scheme in real applications.

However, as stated above, the scheme in its current state is not yet general enough to be practically usable and generalizations, such as the one proposed above, still need to be investigated thoroughly. Further convergence studies and practical applications of the scheme should hence be postponed until a reliable generalization is available.

Dealing with bulk–surface models from class (5.1)

Once a generalization of Scheme 5.3.3 proves to perform well, it is reasonable to apply the operator splitting approach from Section 5.2 and this generalization to the class of evolving geometry bulk–surface models (5.1).

It should be noted, though, that we still require some appropriate scheme for the bulk advection equation in subproblems of type (5.5) to obtain a full scheme for bulk–surface models on evolving geometries. If the material velocity $\mathbf{v}$ of $\Omega(t) \cup \Gamma(t)$ is not an externally given field and the model itself drives the evolution of the geometry instead, such a full scheme furthermore needs to be supplemented with an appropriate solver dealing with level set equation (3.25).

More general classes of bulk–surface models on evolving geometries

As discussed in Section 4.4.1, the numerical approaches which we employ for dealing with subproblems on static geometries can be used for a wider range of bulk–surface models. They are not limited to the static geometry special case of the example class of bulk–surface models (5.1) chosen in this work. As a result, overall schemes for bulk–surface models on evolving geometries which arise from the approaches presented in this chapter can be transferred to more general classes of such models.

6. Conclusion

Motivated by applications from cell biology and by drawbacks of classical mesh-based simulation frameworks for bulk–surface models, in this thesis, we developed numerical schemes that build upon transferring the UDG method for bulk PDEs to PDEs on hypersurfaces. These schemes are particularly suitable for problems comprising continuity equations with hypersurfaces and bulk domains of complex shape. Our developments are based on

- careful consideration of the theory of conservation laws and their PDE analogues,
- the ideas of classical, fitted DG methods and of geometrically unfitted DG methods for this kind of equations in bulk domains,
- the representation of bulk domains and hypersurfaces in an implicit way by means of the level set framework,
- and on a detailed analysis of differential operators for surface fields, with a particular focus on their representation in the level set context.

Since our schemes use extensions of surface PDEs to suitable bulk domains and concepts of sharp interface FEMs for surface PDEs in a hybrid manner, they can be implemented easily using existing implementations of the UDG method for bulk PDEs, while discrete analogues to the models' original conservation properties are still recovered.

We were able to show that our approaches for static geometries in Chapter 4 offer very good properties. These properties particularly justify employing the approaches in practical applications, exploiting that they can be transferred to other classes of model equations in a straightforward manner. As an overall result, our UDG schemes have the potential to replace classical mesh-based simulation frameworks for bulk–surface models. Our results also motivate further investigations and enhancements, such as incorporating the ideas of isoparametric unfitted FEMs with regard to higher order convergence. Please refer to the discussion in Section 4.4 for a detailed conclusion on this part of the thesis.

The extensions which we investigated in Chapter 5 are first steps toward conservative UDG schemes for bulk–surface models that comprise continuity equations on evolving geometries. Particular steps are the operator splitting approach proposed in Section 5.2, which allows for conservative schemes that recycle our approaches from Chapter 4, and the development of Scheme 5.3.3. The latter successfully deals with conservative material transport driven by an evolving hypersurface, without requiring moving meshes or space–time meshes.

Obtaining practically relevant schemes using the operator splitting approach from Section 5.2 requires further investigating generalizations of Scheme 5.3.3, as well as finding UDG schemes that are suitable for dealing with conservative material transport driven by an evolving bulk domain. Details are given in the discussion in Section 5.4.

A. Software

In this appendix, we describe the numerical software which has been created as part of this thesis, aiming at reproducibility of the results from Section 4.3 and Section 5.3.6. The numerical schemes from Chapter 4 and Chapter 5 have been implemented and tested in C++ using a framework for numerical software development called `DUNE`.

We begin in Section A.1 by providing an overview of `DUNE` and its modular design in general. In Section A.2, we continue by introducing a `DUNE` module which implements the schemes from Chapter 4 and supplies tools to analyze those schemes. Section A.3 focusses on a `DUNE` module which has been created for the approach from Section 5.3.

A.1. `DUNE`

The Distributed and Unified Numerics Environment (`DUNE`)[1] is a modular toolbox comprising a set of C++ libraries for solving PDEs using mesh-based numerical methods. It is free software based on the idea of using abstract interfaces to separate data structures and algorithms. Due to its design, it supports different implementations of the same numerical concepts, such as meshes or solvers, and allows for reusing existing software packages. Interfaces are implemented efficiently using modern C++ programming techniques which effectively remove interfaces at compile time, resulting in very low overhead. The framework consists of a number of modules that are downloadable as separate packages. Modules known as *core modules* are used by most other modules, such as *discretization modules* and *application modules* that are built on top.

The set of core modules which are relevant to the code written as part of this thesis comprises `dune-common`, `dune-grid`, `dune-geometry`, `dune-istl`, and `dune-localfunctions`. A brief description of these modules and of all other `DUNE` modules upon which our code is built is given below.

`dune-common`:
Basic infrastructure for all `DUNE` modules, such as classes for exception handling and dense matrix/vector classes, is provided by the `dune-common`[2] module. The module also contains the `DUNE` build system, which is used to set up `DUNE` installations.

[1] `https://www.dune-project.org`
[2] `https://gitlab.dune-project.org/core/dune-common`

dune-grid:
In DUNE, meshes have a hierarchical structure and are known as grids (see Bastian et al., 2008b). Meshes in the sense of our definitions in Section 3.2.1 and Section 4.2.2 are subsets of such grids. The module dune-grid[3] contains some grid implementations and provides the underlying interfaces. The latter allow for using additional grid managers that are available as extra modules. The module also provides infrastructure for grid input and output.

dune-geometry:
Reference elements in DUNE, as well as the corresponding geometry transformations and quadrature rules are implemented in the dune-geometry[4] module.

dune-localfunctions:
The module dune-localfunctions[5] specifies interfaces for shape functions defined on the DUNE reference elements (see dune-geometry), for local interpolation operators, and for additional information which allows for assembling global finite element spaces. Implementations of these interfaces are provided for typical finite elements, e.g. Lagrangian shape functions for Lagrange elements and monomial shape functions for DG elements.

dune-istl:
The iterative solver template library dune-istl[6] comprises generic sparse matrix/vector classes supporting a recursive block structure, related infrastructure, and a variety of solvers based on these classes, including Krylov methods and aggregation-based algebraic multigrid. Bindings to libraries for the direct solution of sparse linear systems are provided as well.

As part of this thesis, eigenvalue solvers were built into the module, which allows for computing the spectral condition number of system matrices numerically. See Section B.5 for details.

dune-pdelab:
The module dune-pdelab[7] is a discretization module providing high-level abstractions to allow for a reasonably quick implementation of discretization schemes and of solvers for systems of PDEs. Its design offers flexibility regarding features of finite element spaces and targets a wide range of discretization schemes. The latter is achieved by employing an abstract problem formulation known as the weighted residual formulation (Bastian et al., 2010). Operators are implemented from an element-local point of view.

[3] https://gitlab.dune-project.org/core/dune-grid
[4] https://gitlab.dune-project.org/core/dune-geometry
[5] https://gitlab.dune-project.org/core/dune-localfunctions
[6] https://gitlab.dune-project.org/core/dune-istl
[7] https://gitlab.dune-project.org/pdelab/dune-pdelab

Focussing on systems of PDEs, finite element spaces in `dune-pdelab` use a tree-based abstraction that is based on the module `dune-typetree`[8], a template library for constructing and operating on statically-typed trees of objects.

`dune-udg`:
The module `dune-udg`[9] (Engwer and Heimann, 2012) aims at complementing the `dune-pdelab` discretization module by adding infrastructure that is required for implementing cut cell methods, particularly the UDG method which we are using in this thesis. In large parts, this infrastructure is summarized in Section A.2 below. Most prominently, the module provides implementations of different routines for constructing local triangulations, which allows to perform numerical integration over cut cells. The core of all routines for constructing local triangulations is provided by the `dune-mc` module described below, which is used to generate the contributions of single elements of the geometry mesh $\mathcal{T}_h(\Omega_\Phi)$ in each routine.

`dune-mc`:
The module `dune-mc`[10] provides an implementation of the topology preserving marching cubes and marching simplex algorithm that is described in Engwer and Nüßing (2017). Particularly for `DUNE` reference elements, the latter allows for constructing a partition with respect to the zero level set of a given discrete Q_1 or P_1 level set function Φ_h. The partition is retrieved as a collection of simple geometric objects corresponding to the set of points with $\Phi_h < 0$, a collection of simple geometric objects corresponding to the set of points with $\Phi_h > 0$, and a collection of simple geometric objects corresponding to the codimension 1 boundaries of these two sets. Geometric objects are used that are suitable for performing numerical integration using standard quadrature rules. The partition is generated solely from the vertex values of the level set function in a highly efficient manner using lookup tables.

The module was renamed to `dune-tpmc` in a version of the module that is more recent than those versions which we are using in this thesis.

For further reading on `DUNE` and its core modules, we refer to the `DUNE` website[11] and to Bastian et al. (2008b,a); Blatt et al. (2016). The website also contains information on how to obtain a working installation using the build system in `dune-common`. Please also refer to Blatt and Bastian (2007) with regard to `dune-istl`, to Bastian et al. (2010) regarding `dune-pdelab`, and to Engwer and Heimann (2012) regarding `dune-udg`.

[8] `https://gitlab.dune-project.org/staging/dune-typetree`
[9] `https://gitlab.dune-project.org/cutcell/dune-udg`
[10] `https://gitlab.dune-project.org/extensions/dune-tpmc`
[11] `https://www.dune-project.org`

A.2. The dune-udg-bulksurface module

As part of this thesis, an application module `dune-udg-bulksurface`[12] has been implemented to investigate the approaches for PDEs on static geometries which we introduce in Chapter 4. Besides providing its own infrastructure, the module heavily uses infrastructure that is provided by `dune-pdelab` and by `dune-udg`. Our code particularly uses the following `dune-pdelab` facilities:

- its infrastructure for assembling global finite element spaces, such as the spaces $V_{s,h}(\Omega_{\delta,\hbar})$ and $V_{b,h}(\Omega_\hbar)$ (see Section 4.2.2), from local finite element spaces, such as those provided by `dune-udg` (see below),
- its matrix/vector backends and DOF management features,
- its time-stepping schemes,
- its bindings to linear solvers from `dune-istl`,
- and its solver for nonlinear systems based on Newton's method.

Infrastructure of `dune-udg` is used for:

- defining cut cell meshes of bulk domain reconstructions that are defined using multiple level set functions, such as the discrete narrow band $\Omega_{\delta,\hbar}$,
- constructing local triangulations for numerical integration over cut cells that are defined via multiple discrete level set functions,
- constructing local finite element spaces on cut cell meshes using the UDG shape functions,
- assembling the global discrete systems of algebraic equations from element-local contributions, in a way that is analogous to the way of proceeding in `dune-pdelab`,
- data output over cut cell meshes.

The numerical results that are presented in Section 4.3 were obtained using the module version of `dune-udg-bulksurface` that is specified in Table A.1. The table also lists compatible versions of the `DUNE` modules it builds upon.

Running the code

Code that is relevant to this thesis is contained in the folder

`./src/udg-isdg-coupling`.

In particular, the approaches and corresponding numerical schemes from Section 4.2 are implemented in a single C++ source code file `./src/udg-isdg-coupling/bulksurface.cc`. The latter is compiled to different executables, one for each polynomial degree k that is employed for the discrete spaces $V_{s,h}(\Omega_{\delta,\hbar})$ and $V_{b,h}(\Omega_\hbar)$ in our UDG schemes. Simulations can be performed by running those executables with some parameter file which configures the

[12] `https://gitlab.dune-project.org/sebastian.westerheide/dune-udg-bulksurface`

model, its parameters, and other simulation parameters, such as settings related to the particular UDG scheme which shall be run. The code allows for simulations in 2D and for simulations in 3D ($d = 2$ or $d = 3$).

An example parameter file with explanations can be found in `./src/udg-isdg-coupling/bulksurface.ini`. Further details are given in the documentation of the source code file `./src/udg-isdg-coupling/bulksurface.cc` and in the C++ headers included therein.

It is worth noting that the implementation in `./src/udg-isdg-coupling/-bulksurface.cc` is structured in a way which prepares implementing UDG schemes for PDEs on evolving geometries on the basis of the operator splitting approach that is proposed in Section 5.2. Example code for the first order operator splitting method that is sketched in Figure 5.1a can be found in the C++ source code file `./src/udg-isdg-coupling/evolving_bulksurface.cc`.

Reproducing the numerical results from Section 4.3

The results of the numerical studies in Section 4.3.1 can be reproduced using the four shell scripts

- `elliptic_2d_testproblem2_h_refinement.sh`,
- `elliptic_2d_testproblem3_h_refinement.sh`,
- `elliptic_2d_testproblem2_alpha_refinement.sh`,
- `elliptic_2d_testproblem3_alpha_refinement.sh`,

which are contained in the folder

`./src/udg-isdg-coupling/simulations/scripts/bulksurface`. (A.1)

They expect precisely one argument, which shall specify the path to the *build directory* of the module, i.e. the path to the root directory of compiled executables of the entire module. Output is written to subfolders of the directory

`./src/udg-isdg-coupling/simulations/output/bulksurface/current`. (A.2)

Folder (A.1) also contains the two shell scripts

- `parabolic_2d_testproblem3_h_refinement.sh`,
- `parabolic_2d_testproblem4_h_refinement.sh`.

These can be called in the same way to reproduce the results of the numerical studies in Section 4.3.2. The results of the simulations for nonlinear parabolic models for cell polarization in Section 4.3.3 and the results of the related numerical study of discrete conservation properties can be reproduced using two shell scripts

- `wavepinning.sh`,
- `goryachev_microscopy_image.sh`.

Again, these are contained in folder (A.1). The call signature and the output mechanism is analogous to the shell scripts above. Please note that the shell script `goryachev_microscopy_image.sh` expects the current directory to be the folder (A.1).

The results written to the corresponding subfolders of directory (A.2) can be visualized using ParaView[13]. Furthermore, the data for the error plots, error tables, and condition number plots in Section 4.3 and the data for the mass plots in Section 4.3.3 can be found in a collection of text files that are part of the output.

A.3. The `dune-udg-evolving` module

A second application module `dune-udg-evolving`[14] has been implemented as part of this thesis to investigate the approach which is introduced in Section 5.3, i.e., our approach for dealing with conservative material transport driven by the geometrical evolution of a hypersurface. Besides providing its own infrastructure, the module uses the infrastructure that is provided by `dune-pdelab` and by `dune-udg` in a similar way to `dune-udg-bulksurface`, see Section A.2.

The numerical results that are presented in Section 5.3.6 were obtained using the module version of `dune-udg-evolving` that is specified in Table A.2. The table also lists compatible versions of the `DUNE` modules it builds upon.

Running the code

Code that is relevant to this thesis is contained in the folder

`./src/udg`.

In particular, the UDG scheme from Section 5.3, i.e. Scheme 5.3.3, is implemented in a single C++ source code file `./src/udg/udg_geometry.cc`. The latter is compiled to a single executable. Simulations can be performed by running this executable with some parameter file which configures the model parameters, i.e. the velocity field and the initial values, and other simulation parameters, such as settings related to the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ which shall be employed. The code allows for simulations in 2D ($d = 2$) with a discrete space $V_{s,h}(D_\hbar)$ of polynomial degree $k = 0$. By using a different `DUNE` grid object, the implementation can be extended in a straightforward way to allow for simulations in 3D, but this has not been done yet since our studies in Section 5.3.6 were only performed in 2D. For polynomial degrees $k > 0$, the operator in

`./dune/udg-evolving/udg/convectiondiffusionccfv.hh`

[13] `https://www.paraview.org`
[14] `https://gitlab.dune-project.org/sebastian.westerheide/dune-udg-evolving`

needs to be generalized slightly, which has not been done up to now to ease changing Scheme 5.3.3 during its development (please see the discussion of future perspectives in Section 5.4.1).

An example parameter file with first explanations can be found in `./src/dune-udg/udg_geometry.ini`. Particular details are given in the documentation of the source code file `./src/udg/udg_geometry.cc` and in the C++ headers included therein.

Reproducing the numerical results from Section 5.3.6

The results of the numerical studies in Section 5.3.6 can be reproduced using the three shell scripts

- `annulus_h_refinement_tau=0.5/annulus.sh`,
- `annulus_h_refinement_tau=0.5_stepinitialvalues/annulus.sh`,
- `annulus_h_refinement_tau=0.5_rotation/annulus.sh`,

which are contained in the folder

$$\texttt{./src/udg/simulations}\,. \tag{A.3}$$

In order to run these shell scripts, the folder (A.3) needs to be copied to the subfolder `./src/udg` of the build directory of the module. Please recall that the term *build directory* refers to the root directory of compiled executables of the entire module.

Each of the above shell scripts expects no arguments and directly writes its output to corresponding subfolders of the form `n_cells_*`. The results written to these subfolders can again be visualized using ParaView. Furthermore, Table 5.1 can be generated from the output files by running the Python script

`annulus_h_refinement_tau=0.5/eoc.py`,

which is also contained in folder (A.3).

Module	Version	Git branch	Git hash	Date
dune-udg-bulksurface	0.1	releases/2.3-compatible	085111f61b16fc6beec6c03de05ddf69f963bb57	Apr 28 20:00:48 2018 +0200
dune-common	2.3.1	releases/2.3	8cfcea54a78fee457993729535b7c63b3497be5e	Jul 17 09:32:33 2014 +0200
dune-grid	2.3.1	releases/2.3	e2b991f1ccfe5b74a82da581eab5472a1778f334	Jun 26 14:05:25 2015 +0200
dune-geometry	2.3.1	releases/2.3	8e5075fc4c58e116d4a0f003a4d1f0482a3ab618	Jun 3 16:12:54 2014 +0200
dune-localfunctions	2.3.1	releases/2.3	eedfe2a7639be4a7e6dc457fc5181454088aebe0	Jun 3 16:12:54 2014 +0200
dune-istl	2.3.1	releases/2.3	b5979486b15ad529251b815fb63a9b4d2c765b49	May 28 10:54:45 2015 +0200
dune-pdelab	2.0.0	releases/2.0	f7a7ff4a23e6f43967b006318480ffbf894ff63f	Feb 23 10:43:59 2015 +0100
dune-typetree	2.3.1	releases/2.3	ecffa10c59fa61a0071e7c788899464b0268719f	Jul 23 17:49:24 2014 +0200
dune-udg	0.1	releases/2.3-compatible	8687465e642857c4bdcc70fe4b9a26c82b6feb97	Jun 7 14:36:00 2017 +0200
dune-mc	0.1	master	af8b6acbf35c4e83bdf0440edceff2fbba063ad9	Aug 26 11:39:24 2014 +0200

Table A.1.: The dune-udg-bulksurface module version which has been used to produce the results in this thesis, and compatible versions of the DUNE modules it builds upon.

Module	Version	Git branch	Git hash	Date
dune-udg-evolving	0.1	master	741dc7721f80ae36e5bc723123ff12c6b02f8600	Apr 30 11:23:11 2018 +0200
dune-common	2.4	releases/2.4	7c4f09c116e0f7a053e84935cc5c4d9e0e0adb5a	Sep 24 14:45:19 2015 +0200
dune-grid	2.4	releases/2.4	34ce5171ae6fb97180310682a9759369607807e6	Sep 23 11:07:19 2015 +0200
dune-geometry	2.4	releases/2.4	5807479ea23e5501ae5f057027f37bb19376ffc6	Oct 12 14:32:57 2015 +0200
dune-localfunctions	2.4	releases/2.4	d00f8be5a05a775febe572f30681120b6f626acd	Aug 25 17:24:34 2015 +0200
dune-istl	2.4	releases/2.4	f40f33096c811f21e2760b3b6b8bb0c67ac1da28	Oct 9 13:40:48 2015 +0200
dune-pdelab	2.4-dev	releases/2.4	90b0fd4627de43c48f160a377de5aec975a4d541	Sep 17 18:37:16 2015 +0200
dune-typetree	2.4.0-rc1	releases/2.4	cdb5184745fb61af4ee8d690e890d7a772dd968d	Oct 13 15:16:52 2015 +0200
dune-udg	0.1	releases/2.4-compatible	03363eb610ecb8e1633023ef9aa4918f19eaf8ec	Nov 23 15:24:48 2015 +0100
dune-mc	0.1	master	6548608dd8ee607673343b86333ad8f81d91172c	Jun 3 16:31:13 2015 +0200

Table A.2.: The dune-udg-evolving module version which has been used to produce the results in this thesis, and compatible versions of the DUNE modules it builds upon.

B. The condition number of a matrix

When solving mathematical problems using a computer, the description of the problem's input data often includes perturbations. These perturbations can not be avoided, in general. They can already arise from round-off errors entering the data description due to the limited precision of the floating point arithmetic of the machine. As a result, the problem's solution will contain an unavoidable error whose origin solely lies in perturbed input data and the mathematical problem itself. This error is particularly not caused by the numerical scheme that is applied to solve the problem. However, other numerical methods might contribute to the definition of the problem, yielding perturbed input data.

While the pertubation behavior of the scheme which is employed to solve a mathematical problem is described by a concept known as *the stability of the scheme*, the dependency of the problem's solution from perturbed input data, independent of the choice of the solution scheme, is described by a concept known as *the condition of the problem*. Both concepts are orthogonal to each other.

In this appendix, we focus on the condition of an important problem which arises from all discretization methods that are similar to those introduced in this thesis, namely the problem of solving a system of linear equations. In particular, we recall indicators allowing for the measurement of the condition of this problem and look at their interpretation. Furthermore, we deal with the computation of a specific indicator, both from a theoretical and a numerical point of view.

We begin in Section B.1 by introducing a general definition of condition numbers, deriving condition numbers of the problem of solving a system of linear equations, and looking at a related estimate for error propagation. In Section B.2, we continue with theory from linear algebra which leads to a formula for computing a particular condition number of our specific problem. This formula is presented in Section B.3. It is known as the *spectral condition number of a matrix*, but often just called *the condition number of a matrix*. Section B.4 deals with the question of how the spectral condition number can be computed using numerical schemes. Finally, we discuss the practical implementation of those schemes in the `dune-istl` module in Section B.5.

B.1. Basic definitions and facts

The condition of a mathematical problem is described by numbers which measure the factor by which input errors are amplified in the worst case. These factors are known as the problem's *condition numbers*.

In particular, let $f\colon \mathbb{C}^n \to \mathbb{C}^m$ with $n, m \in \mathbb{N}$ be a function which represents an abstract mathematical problem, let $\|\cdot\|$ represent arbitrary norms on $\mathbb{C}^n$ and $\mathbb{C}^m$, respectively, and let $\tilde{\underline{\mathbf{x}}} \in \mathbb{C}^n$ denote a slightly perturbed version of input data represented by a point $\underline{\mathbf{x}} \in \mathbb{C}^n$. Then, measuring perturbations relative to the norm of the object being perturbed, namely $\underline{\mathbf{x}}$, condition numbers (specifically, relative condition numbers) are given by the following definitions.

Definition B.1.1 (Condition number)**.** The condition number of the problem f at a point $\underline{\mathbf{x}}$ with respect to a particular choice of the norms $\|\cdot\|$ is defined by

$$\begin{aligned}\kappa(f, \underline{\mathbf{x}}) &:= \lim_{\varepsilon \to 0} \sup_{\|\tilde{\underline{\mathbf{x}}} - \underline{\mathbf{x}}\| \leq \varepsilon} \frac{\|f(\tilde{\underline{\mathbf{x}}}) - f(\underline{\mathbf{x}})\|}{\|f(\underline{\mathbf{x}})\|} \;/\; \frac{\|\tilde{\underline{\mathbf{x}}} - \underline{\mathbf{x}}\|}{\|\underline{\mathbf{x}}\|} \\ &= \lim_{\varepsilon \to 0} \sup_{\|\tilde{\underline{\mathbf{x}}} - \underline{\mathbf{x}}\| \leq \varepsilon} \frac{\|f(\tilde{\underline{\mathbf{x}}}) - f(\underline{\mathbf{x}})\|}{\|\tilde{\underline{\mathbf{x}}} - \underline{\mathbf{x}}\|} \;\cdot\; \frac{\|\underline{\mathbf{x}}\|}{\|f(\underline{\mathbf{x}})\|},\end{aligned}$$

which can be interpreted as the maximum ratio of relative changes in $f(\underline{\mathbf{x}})$ to relative changes in $\underline{\mathbf{x}}$, in the limit where changes in $\underline{\mathbf{x}}$ become infinitesimally small.

If f is differentiable in $\underline{\mathbf{x}}$, this expression is equivalent to

$$\kappa(f, \underline{\mathbf{x}}) = \|D\mathbf{f}(\underline{\mathbf{x}})\| \cdot \frac{\|\underline{\mathbf{x}}\|}{\|f(\underline{\mathbf{x}})\|}, \tag{B.1}$$

where now $\|\cdot\|$ also represents the matrix norm induced by the two chosen vector norms and $D\mathbf{f}(\underline{\mathbf{x}})$ denotes the Jacobian matrix of f at $\underline{\mathbf{x}}$. See Trefethen and Bau (1997, Lecture 12) for more information.

Definition B.1.2 (Global condition number)**.** In addition to the condition number $\kappa(f, \underline{\mathbf{x}})$ which has a local nature, a global condition number of the problem f with respect to $\|\cdot\|$ can be defined as

$$\kappa(f) := \sup_{\underline{\mathbf{x}} \in \mathcal{D}} \kappa(f, \underline{\mathbf{x}}).$$

Here, $\mathcal{D} \subseteq \mathbb{C}^n$ denotes the set of points for which the condition of the problem shall be analyzed.

Clearly, the choice of the set of points that are considered in Definition B.1.2 depends on the focus of the analysis of a specific problem's condition. Given some a priori knowledge about the problem f, it can also be reasonable to systematically exclude points from $\mathcal{D}$. For instance, it can be suitable to

exclude points $\underline{\mathbf{x}}$ for which it is known that $f(\underline{\mathbf{x}}) = 0$. Note that if f has an isolated zero at a point $\underline{\mathbf{x}}$, its condition number $\kappa(f, \underline{\mathbf{x}})$ is infinite due to the zero in the denominator. This accounts for the fact that infinitesimal changes in the input yield infinite relative change in the output in this case.

A problem of fundamental importance is the problem of matrix–vector multiplication. It can be described by a linear function $f(\underline{\mathbf{x}}) := A\underline{\mathbf{x}}$ with $A \in \mathbb{C}^{m\times n}$. We can assume that A is not the zero matrix (otherwise the problem is trivial). For this specific problem, applying Definition B.1.1 in terms of equation (B.1) yields

$$\kappa(f, \underline{\mathbf{x}}) = \|A\| \cdot \frac{\|\underline{\mathbf{x}}\|}{\|A\underline{\mathbf{x}}\|}.$$

Furthermore, a global condition number (with respect to the chosen vector norms $\|\cdot\|$) of matrix–vector multiplication can be defined as

$$\begin{aligned}\kappa(f) &:= \sup_{\underline{\mathbf{x}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \kappa(f, \underline{\mathbf{x}}) \\ &= \|A\| \cdot \sup_{\underline{\mathbf{x}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \frac{\|\underline{\mathbf{x}}\|}{\|A\underline{\mathbf{x}}\|} \\ &= \sup_{\underline{\mathbf{x}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \frac{\|A\underline{\mathbf{x}}\|}{\|\underline{\mathbf{x}}\|} \Big/ \inf_{\underline{\mathbf{x}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \frac{\|A\underline{\mathbf{x}}\|}{\|\underline{\mathbf{x}}\|} \in [1, \infty],\end{aligned}$$

where we have used the equality

$$\sup_{\underline{\mathbf{x}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \frac{\|\underline{\mathbf{x}}\|}{\|A\underline{\mathbf{x}}\|} = \left(\inf_{\underline{\mathbf{x}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \frac{\|A\underline{\mathbf{x}}\|}{\|\underline{\mathbf{x}}\|}\right)^{-1}$$

in the last step. This definition follows Definition B.1.2, where the point $\underline{\mathbf{0}}$ has been excluded since matrix–vector multiplication for $\underline{\mathbf{x}} = \underline{\mathbf{0}}$ is trivial. It is known a priori that $f(\underline{\mathbf{0}}) = A\underline{\mathbf{0}} = \underline{\mathbf{0}}$. Including the point $\underline{\mathbf{0}}$ in the definition would render $\kappa(f)$ meaningless to problems of practical relevance.

We will now write down first properties of the global condition number of this problem. If $n, m \in \mathbb{N}$ are arbitrary numbers and f is not injective, there is a point $\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$ with $f(\underline{\mathbf{x}}) = A\underline{\mathbf{x}} = \underline{\mathbf{0}}$ and hence $\kappa(f) = \infty$. On the other hand, if f is injective, f does not have any zeros in $\mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$ such that $\kappa(f) < \infty$. Moreover, if $m = n$ and f is bijective which means that the matrix A is square and invertible, we find

$$\kappa(f) = \|A\| \cdot \|A^{-1}\|. \tag{B.2}$$

Note that

$$\sup_{\underline{\mathbf{x}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \frac{\|\underline{\mathbf{x}}\|}{\|A\underline{\mathbf{x}}\|} = \sup_{\underline{\mathbf{z}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \frac{\|A^{-1}\underline{\mathbf{z}}\|}{\|\underline{\mathbf{z}}\|} = \|A^{-1}\|$$

in this case.

Instead of writing $\kappa(f)$ and talking about the condition number of matrix–

vector multiplication, it is common to denote the condition number of this problem by $\kappa(A)$ and to call $\kappa(A)$ *the condition number of* A with respect to the chosen vector norms $\|\cdot\|$. Note that for an invertible matrix A, we have

$$\kappa(A) = \|A\| \cdot \|A^{-1}\| = \kappa(A^{-1}) \tag{B.3}$$

according to equation (B.2). Hence, the condition number of A is not only an indicator for the accuracy of results from matrix–vector multiplication as described above, i.e. the problem of computing $A\underline{\mathbf{x}}$ from an input $\underline{\mathbf{x}}$ for a given matrix A. It is also, and this is what we are interested in in this thesis, an indicator for the accuracy of results from the associated inverse problem of computing $A^{-1}\underline{\mathbf{b}}$ from an input $\underline{\mathbf{b}}$ for a given invertible matrix A. This inverse problem corresponds solving the system of linear equations

$$A\underline{\mathbf{x}} = \underline{\mathbf{b}}$$

which is represented by the given matrix A.

A theorem which allows to actually estimate the relative error which results from solving a system of linear equations with a perturbed right-hand side using the condition number of the system matrix is the following.

Theorem B.1.3 (Estimation of the relative error)**.** *Let $A \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}$, be an invertible matrix. Furthermore, let $\underline{\mathbf{x}} \in \mathbb{C}^n$ and $\tilde{\underline{\mathbf{x}}} \in \mathbb{C}^n$ be the solutions of $A\underline{\mathbf{x}} = \underline{\mathbf{b}}$ and $A\tilde{\underline{\mathbf{x}}} = \tilde{\underline{\mathbf{b}}}$, respectively, where $\tilde{\underline{\mathbf{b}}} \in \mathbb{C}^n$ is a slightly perturbed version of a vector $\underline{\mathbf{b}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$. Then*

$$\frac{\|\tilde{\underline{\mathbf{x}}} - \underline{\mathbf{x}}\|}{\|\underline{\mathbf{x}}\|} \leq \kappa(A)\, \frac{\|\tilde{\underline{\mathbf{b}}} - \underline{\mathbf{b}}\|}{\|\underline{\mathbf{b}}\|}.$$

Proof. Due to the linearity, subtracting both sides of the systems of linear equations yields $A(\tilde{\underline{\mathbf{x}}} - \underline{\mathbf{x}}) = \tilde{\underline{\mathbf{b}}} - \underline{\mathbf{b}}$ and hence $\tilde{\underline{\mathbf{x}}} - \underline{\mathbf{x}} = A^{-1}(\tilde{\underline{\mathbf{b}}} - \underline{\mathbf{b}})$. Therefore, using the property $\|A\underline{\mathbf{z}}\| \leq \|A\|\,\|\underline{\mathbf{z}}\|$ of the matrix norm and equation (B.3) we find

$$\begin{aligned}\frac{\|\tilde{\underline{\mathbf{x}}} - \underline{\mathbf{x}}\|}{\|\underline{\mathbf{x}}\|} &= \frac{1}{\|\underline{\mathbf{x}}\|}\left\|A^{-1}\big(\tilde{\underline{\mathbf{b}}} - \underline{\mathbf{b}}\big)\right\| \leq \|A^{-1}\|\, \frac{\|\tilde{\underline{\mathbf{b}}} - \underline{\mathbf{b}}\|}{\|\underline{\mathbf{x}}\|} \\ &= \kappa(A)\, \frac{\|\tilde{\underline{\mathbf{b}}} - \underline{\mathbf{b}}\|}{\|A\|\|\underline{\mathbf{x}}\|} \leq \kappa(A)\, \frac{\|\tilde{\underline{\mathbf{b}}} - \underline{\mathbf{b}}\|}{\|A\underline{\mathbf{x}}\|} = \kappa(A)\, \frac{\|\tilde{\underline{\mathbf{b}}} - \underline{\mathbf{b}}\|}{\|\underline{\mathbf{b}}\|}. \quad \square\end{aligned}$$

As announced at the beginning of this section, the condition number of the system matrix thus measures the factor by which the relative error in the right-hand side of a system of linear equations is amplified in the worst case.

To compute the condition number $\kappa(A)$ of A with respect to a particular choice of the vector norms $\|\cdot\|$, namely the Euclidean norm $|\cdot|$ which is also known as 2-norm, we need some theory from linear algebra.

B.2. Theory from linear algebra

In large part, the following presentation of required theory from linear algebra is based on Stoer and Bulirsch (2002, Section 6.4). In Section B.2.1, we begin by recalling theory about Hermitian matrices, focussing on their eigenvalues. On this basis, we continue in Section B.2.2 with the definition of singular values of arbitrary matrices and an important characterization of extremal singular values which will take us back to computing the condition number of a matrix.

B.2.1. Eigenvalues of Hermitian matrices

Theorem B.2.1 (Schur decomposition)**.** *For every square matrix $B \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}$, there is a unitary matrix $U \in \mathbb{C}^{n\times n}$ (i.e. $U^* = U^{-1}$, where $U^* = \bar{U}^{\mathrm{tr}}$) with*

$$U^{-1}BU = U^*BU = \begin{pmatrix} \lambda_1(B) & * & \cdots & * \\ & \lambda_2(B) & \ddots & \vdots \\ & & \ddots & * \\ 0 & & & \lambda_n(B) \end{pmatrix}.$$

Here, the entries $\lambda_i(B)$, $i = 1, \ldots, n$, are the eigenvalues of B. They are (not necessarily distinct) complex numbers, in general.

Proof. A proof which is based on induction with respect to n and easy to read can be found in Stoer and Bulirsch (2002, Theorem 6.4.1). □

For Hermitian matrices (i.e. for complex square matrices B with $B^* = B$), Theorem B.2.1 immediately yields the following corollary.

Corollary B.2.2 (Diagonalizability of Hermitian matrices)**.** *Every Hermitian matrix $B \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}$, is diagonalizable by a unitary matrix. This means that given B, there is a unitary matrix $U = (\underline{\mathbf{v_1}}, \ldots, \underline{\mathbf{v_n}}) \in \mathbb{C}^{n\times n}$ with*

$$U^{-1}BU = U^*BU = \operatorname{diag}\big(\lambda_1(B), \ldots, \lambda_n(B)\big).$$

The eigenvalues $\lambda_i(B)$ of B, $i = 1, \ldots, n$, are (not necessarily distinct) real numbers. The j-th column $\underline{\mathbf{v_j}}$ of U is an eigenvector belonging to the eigenvalue $\lambda_j(B)$. B thus has n linearly independent pairwise orthonormal eigenvectors.

Proof. Given that B is a Hermitian matrix,

$$U^*BU = U^*B^*(U^*)^* = U^*(U^*B)^* = (U^*BU)^*$$

is a Hermitian matrix as well. The first part of the corollary thus directly

follows from Theorem B.2.1. For a column $\underline{\mathbf{v_j}}$ of U, we subsequently find

$$\begin{aligned} B\underline{\mathbf{v_j}} &= U \operatorname{diag}\big(\lambda_1(B), \ldots, \lambda_n(B)\big)\, U^* \underline{\mathbf{v_j}} \\ &= U \operatorname{diag}\big(\lambda_1(B), \ldots, \lambda_n(B)\big)\, \underline{\mathbf{e_j}} = \lambda_j(B)\, U \underline{\mathbf{e_j}} = \lambda_j(B)\, \underline{\mathbf{v_j}}, \end{aligned}$$

where we have used that the vectors $\underline{\mathbf{v_1}}, \ldots, \underline{\mathbf{v_n}}$ form an orthonormal set since U is a unitary matrix. □

Definition B.2.3 (Rayleigh quotient)**.** Let $B \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}$, be a Hermitian matrix. Then the mapping $\mathrm{rq}_B \colon \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\} \to \mathbb{R}$ with

$$\mathrm{rq}_B(\underline{\mathbf{x}}) := \frac{\underline{\mathbf{x}}^* B\, \underline{\mathbf{x}}}{\underline{\mathbf{x}}^* \underline{\mathbf{x}}}$$

is called *Rayleigh quotient* of B. For each vector $\underline{\mathbf{x}}$, its value $\mathrm{rq}_B(\underline{\mathbf{x}})$ is called the Rayleigh quotient of B with respect to $\underline{\mathbf{x}}$.

It is easy to see that the Rayleigh quotient of a Hermitian matrix B is indeed a mapping to $\mathbb{R}$ which maps each eigenvector of B to the associated eigenvalue. Moreover, the following theorem will show that computing the largest and smallest eigenvalue of B is equivalent to finding the maximum and minimum value of the Rayleigh quotient of B, respectively.

Theorem B.2.4 (Characterization of extremal eigenvalues by extremal properties of the Rayleigh quotient)**.** *Let $\lambda_1(B) \geq \lambda_2(B) \geq \ldots \geq \lambda_n(B)$ be the eigenvalues of a Hermitian matrix $B \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}$, arranged in decreasing order. Then*

$$\lambda_1(B) = \max_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \mathrm{rq}_B(\underline{\mathbf{x}}) \qquad \textit{and} \qquad \lambda_n(B) = \min_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \mathrm{rq}_B(\underline{\mathbf{x}}).$$

Proof. Following Corollary B.2.2, there is a unitary $U = (\underline{\mathbf{v_1}}, \ldots, \underline{\mathbf{v_n}}) \in \mathbb{C}^{n\times n}$ with $U^* B U = \operatorname{diag}\big(\lambda_1(B), \ldots, \lambda_n(B)\big) =: D$. For all $\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$, we thus have the equality

$$\mathrm{rq}_B(\underline{\mathbf{x}}) = \frac{(\underline{\mathbf{x}}^* U)\, U^* B U\, (U^*\, \underline{\mathbf{x}})}{(\underline{\mathbf{x}}^* U)(U^*\, \underline{\mathbf{x}})} = \frac{\underline{\mathbf{y}}^* D\, \underline{\mathbf{y}}}{\underline{\mathbf{y}}^* \underline{\mathbf{y}}} = \frac{\sum_i \lambda_i(B)\, |\mathrm{y}_i|^2}{\sum_i |\mathrm{y}_i|^2},$$

where $\underline{\mathbf{y}} := U^*\, \underline{\mathbf{x}} = \big(\mathrm{y}_1, \ldots, \mathrm{y}_n\big)^{\mathrm{tr}} \neq \underline{\mathbf{0}}$. Hence, for an arbitrary $\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$, we get

$$\mathrm{rq}_B(\underline{\mathbf{x}}) \leq \lambda_1(B) \qquad \text{and} \qquad \mathrm{rq}_B(\underline{\mathbf{x}}) \geq \lambda_n(B).$$

Furthermore, the particular choices $\underline{\mathbf{x}} = \underline{\mathbf{v_1}} \neq \underline{\mathbf{0}}$ and $\underline{\mathbf{x}} = \underline{\mathbf{v_n}} \neq \underline{\mathbf{0}}$ yield

$$\mathrm{rq}_B\big(\underline{\mathbf{v_1}}\big) = \lambda_1(B) \qquad \text{and} \qquad \mathrm{rq}_B\big(\underline{\mathbf{v_n}}\big) = \lambda_n(B)$$

which completes the proof. □

Corollary B.2.5 (Definiteness of Hermitian matrices)**.** *A Hermitian matrix $B \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}$, is positive definite if and only if all of its eigenvalues are positive. It is positive semidefinite if and only if all of its eigenvalues are nonnegative.*

Proof. Let $B \in \mathbb{C}^{n\times n}$ be a Hermitian matrix. Noting that

$$\begin{aligned} & \underline{\mathbf{x}}^* B\, \underline{\mathbf{x}} \left\{ \begin{matrix} > \\ \geq \end{matrix} \right\} 0 \quad \text{for all } \underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\} \\ \Leftrightarrow \quad & \mathrm{rq}_B(\underline{\mathbf{x}}) \left\{ \begin{matrix} > \\ \geq \end{matrix} \right\} 0 \quad \text{for all } \underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\} \\ \Leftrightarrow \quad & \min_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \mathrm{rq}_B(\underline{\mathbf{x}}) \left\{ \begin{matrix} > \\ \geq \end{matrix} \right\} 0, \end{aligned}$$

the corollary directly follows from Theorem B.2.4 which states that

$$\lambda_1(B) \geq \lambda_2(B) \geq \ldots \geq \lambda_n(B) = \min_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \mathrm{rq}_B(\underline{\mathbf{x}}). \qquad \square$$

Lemma B.2.6 (Invertibility of square matrices)**.** *A square matrix $B \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}$, is invertible if and only if $\lambda_i(B) \neq 0$ for $i = 1, \ldots, n$. Here, the $\lambda_i(B)$, $i = 1, \ldots, n$, are the (not necessarily distinct) eigenvalues of B. Furthermore, if B is Hermitian and invertible, then*

$$B^{-1} = U \operatorname{diag}\big(\lambda_1(B)^{-1}, \ldots, \lambda_n(B)^{-1}\big)\, U^*,$$

where $U \in \mathbb{C}^{n\times n}$ is the unitary matrix from Corollary B.2.2.

Proof. According to Theorem B.2.1, there is a unitary matrix $U \in \mathbb{C}^{n\times n}$ with $U^*BU = T$, where $T \in \mathbb{C}^{n\times n}$ is an upper triangular matrix with diagonal entries $\lambda_i(B)$, $i = 1, \ldots, n$. Therefore,

$$\begin{aligned} & B \text{ invertible} \\ \Leftrightarrow\ & \prod_i \lambda_i(B) = \det(T) = \det(U^*) \cdot \det(B) \cdot \det(U) = \det(B) \neq 0 \\ \Leftrightarrow\ & \lambda_i(B) \neq 0 \quad \text{for all } i \in \{1, \ldots, n\}, \end{aligned}$$

where we have used that $\det(U^*) = \det(U) = 1$ since U is a unitary matrix. The second part of the lemma is a direct consequence of Corollary B.2.2. $\square$

B.2.2. Singular values

Theorem B.2.7 (Properties of A^*A)**.** *For every matrix $A \in \mathbb{C}^{m\times n}$, $n, m \in \mathbb{N}$, the matrix $B := A^*A \in \mathbb{C}^{n\times n}$ has the following properties:*

1. *B is Hermitian and positive semidefinite.*

2. *B is positive definite if and only if* $\ker(A) = \{0\}$*, i.e. if the corresponding linear map is injective.*
3. *B is invertible if and only if it is positive definite.*

Proof. We start by proving property 1. B is a Hermitian matrix since

$$B = A^*A = A^*(A^*)^* = (A^*A)^* = B^*,$$

and it is positive semidefinite since

$$\underline{\mathbf{x}}^* B\,\underline{\mathbf{x}} = \underline{\mathbf{x}}^* A^* A\,\underline{\mathbf{x}} = |A\underline{\mathbf{x}}|^2 \geq 0$$

for all $\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$. For an arbitrary $\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$, furthermore the following equivalence holds which proves property 2:

$$|A\underline{\mathbf{x}}|^2 > 0 \;\Leftrightarrow\; A\underline{\mathbf{x}} \neq \underline{\mathbf{0}} \;\Leftrightarrow\; \underline{\mathbf{x}} \notin \ker(A).$$

Property 3 follows from Property 1, Lemma B.2.6 and Corollary B.2.5. □

Definition B.2.8 (Singular values)**.** We consider a matrix $A \in \mathbb{C}^{m\times n}$ with $n, m \in \mathbb{N}$, and define $B := A^*A$. Let $\lambda_1(B) \geq \lambda_2(B) \geq \ldots \geq \lambda_n(B) \geq 0$ be the eigenvalues of B, arranged in decreasing order, which are nonnegative according to Theorem B.2.7. Then we can write

$$\lambda_i(B) = \sigma_i(A)^2 \quad \text{with} \quad \sigma_i(A) \geq 0,$$

$i = 1, \ldots, n$. The numbers $\sigma_1(A) \geq \sigma_2(A) \geq \ldots \geq \sigma_n(A) \geq 0$ are called the *singular values* of A.

According to Theorem B.2.7 the singular values of a matrix A are strictly positive if and only if $\ker(A) = \{0\}$. This property also follows from the following theorem which expresses the largest and smallest singular value of A in terms of the 2-norm $|\cdot|$. More importantly, this characterization will take us back to the condition number of a matrix.

Theorem B.2.9 (Characterization of extremal singular values by extremal properties of the 2-norm)**.** *Let $\sigma_1(A) \geq \sigma_2(A) \geq \ldots \geq \sigma_n(A)$ be the singular values of a matrix $A \in \mathbb{C}^{m\times n}$, $n, m \in \mathbb{N}$, arranged in decreasing order. Then*

$$\sigma_1(A) = \max_{\underline{\mathbf{x}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \frac{|A\underline{\mathbf{x}}|}{|\underline{\mathbf{x}}|} \qquad \textit{and} \qquad \sigma_n(A) = \min_{\underline{\mathbf{x}}\in\mathbb{C}^n\setminus\{\underline{\mathbf{0}}\}} \frac{|A\underline{\mathbf{x}}|}{|\underline{\mathbf{x}}|}.$$

Proof. Let $B := A^*A$. Then $\lambda_i(B) = \sigma_i(A)^2$, $i = 1, \ldots, n$, are the eigenvalues of B, arranged in decreasing order. Hence, noting that

$$\mathrm{rq}_B(\underline{\mathbf{x}}) = \frac{\underline{\mathbf{x}}^* B\,\underline{\mathbf{x}}}{\underline{\mathbf{x}}^*\underline{\mathbf{x}}} = \frac{\underline{\mathbf{x}}^* A^* A\,\underline{\mathbf{x}}}{\underline{\mathbf{x}}^*\underline{\mathbf{x}}} = \frac{|A\underline{\mathbf{x}}|^2}{|\underline{\mathbf{x}}|^2},$$

the statement of the theorem is a direct consequence of Theorem B.2.4:

$$\sigma_1(A)^2 = \lambda_1(B) = \max_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \frac{|A\underline{\mathbf{x}}|^2}{|\underline{\mathbf{x}}|^2} = \left(\max_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \frac{|A\underline{\mathbf{x}}|}{|\underline{\mathbf{x}}|} \right)^2,$$

$$\underbrace{\sigma_n(A)}_{\geq 0}^2 = \lambda_n(B) = \min_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \frac{|A\underline{\mathbf{x}}|^2}{|\underline{\mathbf{x}}|^2} = \Bigg(\underbrace{\min_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \frac{|A\underline{\mathbf{x}}|}{|\underline{\mathbf{x}}|}}_{\geq 0} \Bigg)^2. \quad \square$$

Theorem B.2.10 (Singular values of Hermitian matrices)**.** *Let $A \in \mathbb{C}^{n \times n}$, $n \in \mathbb{N}$, be a Hermitian matrix. Let $\sigma_1(A) \geq \sigma_2(A) \geq \ldots \geq \sigma_n(A)$ be the singular values of A, arranged in decreasing order, and let $\lambda_i(A) \in \mathbb{R}$, $i = 1, \ldots, n$, be the eigenvalues of A. Then*

$$\sigma_i(A) = |\lambda_i(A)| \quad \textit{with} \quad |\lambda_1(A)| \geq \ldots \geq |\lambda_n(A)|.$$

Proof. The proof follows from our definitions, particularly Definition B.2.8. Let $B := A^*A$. Then $\lambda_i(B) = \sigma_i(A)^2$, $i = 1, \ldots, n$, are the eigenvalues of B, arranged in decreasing order. Since A is a Hermitian matrix and thus $B = A^2$, these eigenvalues correspond to the square of the eigenvalues of A, arranged in decreasing order with respect to magnitude, i.e.

$$\sigma_i(A)^2 = \lambda_i(A)^2 \quad \text{with} \quad |\lambda_1(A)| \geq \ldots \geq |\lambda_n(A)|.$$

We conclude the proof by using that $\sigma_i(A) \geq 0$ and taking the square root on both sides of the equation. $\square$

B.3. The spectral condition number of a matrix

Let $A \in \mathbb{C}^{m \times n}$, $n, m \in \mathbb{N}$, be a given matrix. The theory from Section B.2 shows that the condition number of A with respect to the Euclidean norm $|\cdot|$ can be expressed in terms of the singular values of A. It is called the *spectral condition number of* A and we write $\kappa_2(A)$.

In particular, according to Theorem B.2.9, Definition B.2.8 and the definitions from Section B.1, we have

$$\kappa_2(A) = \sup_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \frac{|A\underline{\mathbf{x}}|}{|\underline{\mathbf{x}}|} \Big/ \inf_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \frac{|A\underline{\mathbf{x}}|}{|\underline{\mathbf{x}}|} = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} = \frac{\sqrt{\lambda_{\max}(A^*A)}}{\sqrt{\lambda_{\min}(A^*A)}}. \quad \text{(B.4)}$$

Here, $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ denote the largest and the smallest singular value of A, respectively. They correspond to the square roots of the largest and the smallest eigenvalue of the matrix A^*A, which we denote by $\lambda_{\max}(A^*A)$ and $\lambda_{\min}(A^*A)$, respectively. The latter matrix is Hermitian and has only nonnegative eigenvalues according to Theorem B.2.7 and Corollary B.2.5.

Theorem B.2.7 and Corollary B.2.5 furthermore reproduce a property which

has been shown in Section B.1 for arbitrary vector norms, namely that the condition number $\kappa_2(A) \in [1, \infty]$ is smaller than ∞ if and only if $\ker(A) = \{0\}$. If $n = m$ this is exactly the case if A is invertible. As shown in Section B.1, we find

$$\|A^{-1}\| = \left(\inf_{\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}} \frac{\|A\underline{\mathbf{x}}\|}{\|\underline{\mathbf{x}}\|} \right)^{-1}$$

in this case, which yields

$$|A^{-1}| = \frac{1}{\sigma_{\min}(A)} = \frac{1}{\sqrt{\lambda_{\min}(A^*A)}}.$$

If $n = m$ and the matrix A is Hermitian itself, Theorem B.2.10 enables us to rewrite formula (B.4) as

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} = \frac{\left|\lambda_{\text{absmax}}(A)\right|}{\left|\lambda_{\text{absmin}}(A)\right|}, \tag{B.5}$$

where $\lambda_{\text{absmax}}(A)$ and $\lambda_{\text{absmin}}(A)$ denote the largest magnitude eigenvalue of A and its smallest magnitude eigenvalue, respectively.

Summing up, the spectral condition number of a matrix A can be obtained by computing the extremal eigenvalues of a Hermitian matrix B (either A^*A or A itself), particularly its dominant (i.e. largest magnitude) eigenvalue and its least dominant (i.e. smallest magnitude) eigenvalue. To obtain these eigenvalues, we need to solve the basic eigenvalue problem of finding numbers $\lambda(B) \in \mathbb{R}$ and nontrivial vectors $\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$ such that

$$B\underline{\mathbf{x}} = \lambda(B)\,\underline{\mathbf{x}}. \tag{B.6}$$

In this thesis, we compute $\kappa_2(A)$ using numerical approaches to solve problem (B.6) for the extremal eigenvalues of B.

B.4. Numerical computation of eigenvalues

Matrices which result from discretization methods like those introduced in this thesis are square and generally large. Moreover, they have the property that they are sparse which means that there is a limited number of nonzero entries in each row of the matrix. This number is independent of the matrix size. We note that, given a sparse matrix A, the matrix B from Section B.3 is again a sparse matrix.

In the following, let $B \in \mathbb{C}^{n \times n}$ hence be some sparse, square matrix whose eigenvalues shall be examined, possibly with a large $n \in \mathbb{N}$. If not otherwise stated, we do not require B to be a Hermitian matrix, such that its eigenvalues are complex numbers in general. Furthermore, let $\|\cdot\|$ represent an arbitrary vector norm on $\mathbb{C}^n$.

Modern numerical methods for finding eigenvalues of large, sparse matrices

avoid operations other than matrix–vector products which can be performed efficiently for sparse matrices, and operations on the resulting vectors. Such methods belong to the class of iterative eigenvalue algorithms. They are based on the idea that when computing the matrix–vector product $B\underline{\mathbf{x}}$, the eigenvalues of B will amplify the components of $\underline{\mathbf{x}}$ in the direction of eigenvectors with a magnitude greater than 1 and dampen those components in the direction of eigenvectors with a magnitude smaller than 1. This amplification/dampening effect is strongest/weakest for the dominant eigenvalue of B (depending on its magnitude) such that repeated matrix–vector multiplication which is performed on one and the same vector $\underline{\mathbf{x}}$ will pull this vector toward an eigenvector associated with the dominant eigenvalue. To obtain a method which converges to a valid eigenvector, normalization of $\underline{\mathbf{x}}$ is necessary. Without normalization, the magnitude of $\underline{\mathbf{x}}$ would tend to ∞ or 0, depending on whether the magnitude of the dominant eigenvalue of B is greater or smaller than 1. Finally, the computed eigenvector can be used to extract the dominant eigenvalue of B.

B.4.1. Power iteration

The most basic method that is following this idea is the *von Mises vector iteration* which is also known as *power iteration method*. In fact, it follows precisely the idea described above. Starting with a given initial vector $\underline{\mathbf{x^0}} \in \mathbb{C}^n$ with the property that $B\underline{\mathbf{x^0}} \neq \underline{\mathbf{0}}$, it computes sequences

$$\underline{\mathbf{x^k}} := \frac{B\underline{\mathbf{x^{k-1}}}}{\left\|B\underline{\mathbf{x^{k-1}}}\right\|},$$

$$\lambda^k_{\mathrm{absmax}}(B) := \mathrm{rq}_B\left(\underline{\mathbf{x^k}}\right) = \frac{\left(\underline{\mathbf{x^k}}\right)^* B\,\underline{\mathbf{x^k}}}{\left(\underline{\mathbf{x^k}}\right)^* \underline{\mathbf{x^k}}},$$

where rq_B denotes the Rayleigh quotient of B which has been introduced in Definition B.2.3 for Hermitian matrices. Note that extending this Definition to arbitrary square matrices yields a mapping to $\mathbb{C}$ which maps each eigenvector of B to the associated, generally complex eigenvalue. Provided that B has a distinct dominant eigenvalue $\lambda_{\mathrm{absmax}}(B) \in \mathbb{C}$, i.e. an eigenvalue that is strictly greater in magnitude than all of its other eigenvalues, and provided that $\underline{\mathbf{x^0}}$ has a nonzero component in the direction of an eigenvector associated with this eigenvalue, the sequence $\left(\lambda^k_{\mathrm{absmax}}(B)\right)_k$ of Rayleigh quotients of $\underline{\mathbf{x^k}}$ converges to $\lambda_{\mathrm{absmax}}(B)$, while the sequence $\left(\underline{\mathbf{x^k}}\right)_k$ converges to an eigenvector associated with $\lambda_{\mathrm{absmax}}(B)$.

The power iteration method can be described by Algorithm 1. In each iteration, the current vector iterate is multiplied by the matrix B and normalized, and the Rayleigh quotient of B with respect to the resulting vector is computed. In particular, the method does not require solving a system of linear equations or computing the characteristic polynomial of B and finding its roots. It only uses matrix–vector products with our sparse matrix B and

Algorithm 1: Power iteration

Input: matrix $B \in \mathbb{C}^{n\times n}$, initial vector $\underline{\mathbf{x}^{\mathbf{0}}} \in \mathbb{C}^n$ with $B\underline{\mathbf{x}^{\mathbf{0}}} \neq \underline{\mathbf{0}}$
Output: approximation $\lambda_{\text{absmax}}(B) \in \mathbb{C}$ of the dominant eigenvalue, approximation $\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$ of an associated eigenvector

initialize $\underline{\mathbf{x}^{\text{old}}} = \underline{\mathbf{x}^{\mathbf{0}}}$;
while *stopping criterion not satisfied* **do**
  compute

$$\underline{\mathbf{x}} = \frac{B\underline{\mathbf{x}^{\text{old}}}}{\left\|B\underline{\mathbf{x}^{\text{old}}}\right\|};$$

  compute

$$\lambda_{\text{absmax}}(B) = \text{rq}_B(\underline{\mathbf{x}});$$

  set $\underline{\mathbf{x}^{\text{old}}} = \underline{\mathbf{x}}$;
end

scalar products. Both require $\mathcal{O}(n)$ floating point operations which yields a total complexity of $\mathcal{O}(k_{\max} \cdot n)$ floating point operations, where $k_{\max} \in \mathbb{N}$ denotes the number of iterations performed.

Hence, the power iteration method can be a very efficient approach to finding the dominant eigenvalue of matrices which we are interested in. Nevertheless, it should be noted that both sequences $\left(\underline{\mathbf{x}^{\mathbf{k}}}\right)_k$ and $\left(\lambda^k_{\text{absmax}}(B)\right)_k$ converge only linearly with convergence factors

$$\frac{\left|\lambda_{\text{second absmax}}(B)\right|}{\left|\lambda_{\text{absmax}}(B)\right|} \qquad \text{and} \qquad \left(\frac{\left|\lambda_{\text{second absmax}}(B)\right|}{\left|\lambda_{\text{absmax}}(B)\right|}\right)^2, \tag{B.7}$$

respectively, where $\lambda_{\text{second absmax}}(B)$ denotes the second dominant eigenvalue of B (see Trefethen and Bau, 1997, Theorem 27.1). The method thus suffers from slow convergence if there is an eigenvalue with a magnitude close to the dominant eigenvalue, resulting in a large $k_{\max}$.

The size of $k_{\max}$ is also influenced by the stopping criterion which is chosen for Algorithm 1. Stopping criteria for iterative eigenvalue algorithms usually depend on both the current vector iterate $\underline{\mathbf{x}^{\mathbf{k}}}$ and the associated approximate eigenvalue such that we compute $\lambda^k_{\text{absmax}}(B)$ in each iteration of Algorithm 1 rather than once at the end. One possibility is to stop iterating as soon as the norm of the absolute residual falls below a given threshold, i.e. once that $\left\|B\underline{\mathbf{x}^{\mathbf{k}}} - \lambda^k_{\text{absmax}}(B)\,\underline{\mathbf{x}^{\mathbf{k}}}\right\| \leq \varepsilon$, given a target $\varepsilon > 0$.

For more information on the power iteration method and its analysis, we refer to Trefethen and Bau (1997, Lecture 27).

Algorithm 2: Inverse iteration with shift

Input: matrix $B \in \mathbb{C}^{n\times n}$, shift $\mu \in \mathbb{C}$ such that $(B - \mu\mathcal{I})$ invertible, initial vector $\underline{\mathbf{x}^{\mathbf{0}}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$, some linear solver object

Output: approximation $\lambda(B) \in \mathbb{C}$ of the eigenvalue closest to μ, approximation $\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$ of an associated eigenvector

initialize $\underline{\mathbf{x}^{\mathrm{old}}} = \underline{\mathbf{x}^{\mathbf{0}}}$;

while *stopping criterion not satisfied* **do**

solve the following system for $\underline{\mathbf{x}}$ using the linear solver object:

$$(B - \mu\mathcal{I})\underline{\mathbf{x}} = \underline{\mathbf{x}^{\mathrm{old}}}\,;$$

normalize $\underline{\mathbf{x}}$ by setting

$$\underline{\mathbf{x}} = \underline{\mathbf{x}} \,/\, \|\underline{\mathbf{x}}\|\,;$$

compute

$$\lambda(B) = \mathrm{rq}_B(\underline{\mathbf{x}})\,;$$

set $\underline{\mathbf{x}^{\mathrm{old}}} = \underline{\mathbf{x}}$;

end

B.4.2. Inverse iteration with shift

Appying the power iteration method to the matrix $(B - \mu\mathcal{I})^{-1}$ yields a variant known as the *inverse iteration method with shift*, see Algorithm 2. This method can be used to find arbitrary eigenvalues of the matrix B with an adaptable convergence rate, provided a sufficiently accurate approximation of these eigenvalues is known.

Its convergence properties are directly inherited from the power iteration method. In particular, given a shift $\mu \in \mathbb{C}$ and an almost arbitrary initial vector (see assumptions for convergence of the power iteration method), the inverse iteration with shift converges linearly to the eigenvalue of B which is closest to μ and an associated eigenvector of B. Note that the eigenvalues of $(B - \mu\mathcal{I})^{-1}$ are of the form $\big(\lambda_1(B) - \mu\big)^{-1}, \ldots, \big(\lambda_n(B) - \mu\big)^{-1}$, where $\lambda_i(B)$, $i = 1, \ldots, n$, are the eigenvalues of B. The dominant eigenvalue of $(B-\mu\mathcal{I})^{-1}$ hence corresponds to the eigenvalue of B closest to μ. Analogous to (B.7), the factors specifying the linear convergence rate of the sequence of approximate eigenvectors and of the sequence of approximate eigenvalues are

$$\frac{\big|\lambda_{\text{closest to }\mu}(B) - \mu\big|}{\big|\lambda_{\text{second closest to }\mu}(B) - \mu\big|} \quad \text{and} \quad \left(\frac{\big|\lambda_{\text{closest to }\mu}(B) - \mu\big|}{\big|\lambda_{\text{second closest to }\mu}(B) - \mu\big|}\right)^2, \tag{B.8}$$

respectively. Note that the shift μ provides control over these factors.

Taking $\mu = 0$ corresponds to appying the power iteration method to B^{-1} and the method converges to the least dominant eigenvalue $\lambda_{\mathrm{absmin}}(B) \in \mathbb{C}$,

i.e. the eigenvalue of B with the smallest absolute value, and an eigenvector of B which is associated with $\lambda_{\text{absmin}}(B)$.

In each iteration, the inverse iteration method requires solving a system of linear equations with matrix $(B - \mu\mathcal{I})$ which corresponds to calculating a matrix–vector product with the inverse matrix $(B - \mu\mathcal{I})^{-1}$. Since the shift μ is a constant, it seems that this step can be performed efficiently by inverting the matrix $(B - \mu\mathcal{I})$ once at the beginning of the algorithm and working with the precomputed inverse in each iteration. Given the sparse matrix B, the matrix $(B - \mu\mathcal{I})$ is sparse as well. However, the inverse of a sparse matrix is not sparse in general. Therefore, it has to be expected that its storage requires $\mathcal{O}(n^2)$ memory cells and matrix–vector products require $\mathcal{O}(n^2)$ floating point operations. Since n can be large, inverting the matrix $(B - \mu\mathcal{I})$ is not a good idea in our case.

Nevertheless, there are efficient approaches to the direct solution of large, sparse systems of linear equations based on precomputing the LU decomposition of the system matrix and using forward and backward substitution to solve the system of linear equations in each iteration. Approaches which use LU decomposition techniques for sparse matrices are e.g. `SuperLU` (Demmel et al., 1999) and `UMFPACK` (Davis, 2004). As a rule of thumb, they work efficiently for small, medium-sized and large matrices up to $n = 100000$.

Alternatively, iterative methods for solving large, sparse systems of linear equations from the class of *Krylov subspace methods* can be used, such as the conjugate gradient (CG) method (Hestenes and Stiefel, 1952), the biconjugate gradient stabilized (BiCGSTAB) method (van der Vorst, 1992) or the generalized minimal residual (GMRES) method (Saad and Schultz, 1986). Krylov subspace methods for solving linear systems work by successively generating a sequence of successive matrix powers times an initial residual vector and considering the sequence of subspaces which are spanned by this sequence of vectors. These subspaces are known as *Krylov subspaces*. In each step of a Krylov subspace method, the approximation to the solution of the linear system is then formed by minimizing the residual over the current Krylov subspace. All of these operations are solely based on computing matrix–vector products and scalar products. This is what makes Krylov subspace methods so attractive for problems with a sparse matrix.

It should be noted that, the closer the shift μ is to an eigenvalue of B, the worse the spectral condition number of the matrix $(B - \mu\mathcal{I})$. As μ approaches an eigenvalue of B, the smallest singular value $\sigma_{\min}(B - \mu\mathcal{I})$ tends to 0 such that the matrix turns singular, see Section B.3. Luckily, the error has a dominant component in the direction of the eigenvector which we are solving for (cf. Trefethen and Bau, 1997, Exercise 27.5). This renders the inverse iteration method with shift practically usable. Reducing the condition number by combining the method which is employed for solving the linear system with preconditioning techniques is nevertheless beneficial since this usually improves the rate of convergence of iterative linear solvers.

For supplementary information on the inverse iteration method with shift,

Algorithm 3: Rayleigh quotient iteration

Input: Hermitian matrix $B \in \mathbb{C}^{n\times n}$, initial vector $\underline{\mathbf{x}^{\mathbf{0}}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$ such that $(B - \mu\mathcal{I})$ invertible for $\mu := \mathrm{rq}_B\big(\underline{\mathbf{x}^{\mathbf{0}}}\big)$, some linear solver object

Output: approximation $\lambda(B) \in \mathbb{C}$ of an eigenvalue determined by $\underline{\mathbf{x}^{\mathbf{0}}}$, approximation $\underline{\mathbf{x}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$ of an associated eigenvector

initialize $\mu = \mathrm{rq}_B\big(\underline{\mathbf{x}^{\mathbf{0}}}\big)$ and $\underline{\mathbf{x}^{\mathrm{old}}} = \underline{\mathbf{x}^{\mathbf{0}}}$;
while *stopping criterion not satisfied* **do**
  solve the following system for $\underline{\mathbf{x}}$ using the linear solver object:

$$(B - \mu\mathcal{I})\underline{\mathbf{x}} = \underline{\mathbf{x}^{\mathrm{old}}}\,;$$

  normalize $\underline{\mathbf{x}}$ by setting

$$\underline{\mathbf{x}} = \underline{\mathbf{x}} \,/\, \|\underline{\mathbf{x}}\|\,;$$

  compute

$$\lambda(B) = \mathrm{rq}_B(\underline{\mathbf{x}})\,;$$

  set $\mu = \lambda(B)$ and $\underline{\mathbf{x}^{\mathrm{old}}} = \underline{\mathbf{x}}$;
end

we refer to Trefethen and Bau (1997, Lecture 27).

B.4.3. Rayleigh quotient iteration

Convergence factors (B.8) play a key role in the convergence behavior of the inverse iteration method with shift. The algorithm converges the faster, the closer the shift μ is to an eigenvalue of B. Taking into account that μ is nothing else but an eigenvalue estimate, it is quite an obvious idea to accelerate the method by enriching μ with increasingly accurate knowledge about the eigenvalue which is searched for, namely the iterate of the approximate eigenvalue which is obtained in each iteration.

Following this idea yields a method known as *Rayleigh quotient iteration*, see Algorithm 3. It starts with a given initial vector $\underline{\mathbf{x}^{\mathbf{0}}} \in \mathbb{C}^n \setminus \{\underline{\mathbf{0}}\}$ that represents an initial guess for an eigenvector associated with the eigenvalue which is searched for. Each iteration is performed similar to the inverse iteration method with shift, but the shift μ is chosen as the Rayleigh quotient of the approximate eigenvector which has been obtained in the previous iteration.

It turns out that, if B is a Hermitian matrix, the method converges cubically to an eigenvalue and an associated eigenvector of B for almost all initial vectors (see e.g. Parlett, 1974, and the references given therein). Its asymptotically cubic convergence renders the method irresistible since it triples the number of correct digits with each iteration once the error is small enough. Unfortunately, setting up which eigenvalue the method converges to is not as straightforward

as with the inverse iteration method with shift, where the latter provides control over this property. For the Rayleigh quotient iteration, the limit of the generated sequence of approximate eigenvalue–eigenvector pairs depends on the initial vector. But the method not necessarily converges to the eigenvalue which is closest to the Rayleigh quotient of the initial vector. Rather, an initial vector is required that is sufficiently close to some eigenvector associated with the desired eigenvalue of B. The question of how such an initial vector can be generated properly will be answered in Section B.4.4.

Regarding the large, sparse system of linear equations which needs to be solved in each iteration, the same considerations apply as with the inverse iteration method with shift, see Section B.4.2. But the Rayleigh quotient iteration has a minor disadvantage. When combining the method with approaches to the direct solution of large, sparse systems of linear equations which compute the LU decomposition of the system matrix, the advantage of a precomputed decomposition is lost. Since the system matrix $(B - \mu\mathcal{I})$ changes with an updated shift μ, a new decomposition has to be computed in each iteration. We note that the cubic convergence usually compensates for this extra work.

For more information on the Rayleigh quotient iteration method, especially its analysis, we refer to Trefethen and Bau (1997, Lecture 27) and the detailed overview provided in Parlett (1974).

B.4.4. The TLIME algorithm

In Section B.4.2 and Section B.4.3, we have encountered two approaches to finding arbitrarily choosable eigenvalue–eigenvector pairs, each with their own advantages and drawbacks. The inverse iteration method with shift guarantees convergence to the eigenvalue closest to the shift μ, but it only exhibits linear convergence. The Rayleigh quotient iteration converges cubically, but convergence to a specific eigenvalue is only guaranteed for initial vectors in a local neighborhood of an associated eigenvector. The idea of combining the advantages of both methods by using Algorithm 2 to generate an initial vector for Algorithm 3 and then exploiting its cubic convergence is quite natural. A question which needs to be answered is when exactly the switch to Rayleigh quotient iteration can be performed, without risking that the method can still converge to an eigenvalue which we are not interested in. An answer to this question is provided by Szyld (1988) with his *two-level iterative method for eigenvalue calculations (TLIME)*.

In its original formulation, the method can be applied to generalized eigenvalue problems with real, symmetric matrices (i.e. Hermitian matrices over $\mathbb{R}$) and uses an associated vector norm for normalization of the vector iterate. Restricting the method to the basic eigenvalue problem (B.6), with B being a symmetric matrix over $\mathbb{R}$, it can be described as depicted in Algorithm 4. In this special case, the associated vector norm corresponds to the Euclidean norm $|\cdot|$.

Algorithm 4: TLIME

Input:
- symmetric matrix $B \in \mathbb{R}^{n \times n}$
- interval $J = (\gamma - \eta, \gamma + \eta)$, given by an estimate $\gamma \in \mathbb{R}$ for the eigenvalue which shall be approximated and a radius $\eta \in \mathbb{R}^{>0}$ around gamma in which the eigenvalue is expected
- target relative change $\delta \in \mathbb{R}^{>0}$ of the Rayleigh quotient for switching to Rayleigh quotient iteration if J is free of eigenvalues
- minimum number $m \in \mathbb{N}$ of inverse iterations before switching to Rayleigh quotient iteration if J is free of eigenvalues
- initial vector $\underline{\mathbf{x}^{\mathbf{0}}} \in \mathbb{R}^n \setminus \{\underline{\mathbf{0}}\}$
- some linear solver object

Output:
- flag external $\in \{0, 1\}$ indicating if J is free of eigenvalues
- approximation $\lambda(B) \in \mathbb{R}$ of an eigenvalue in J or of the eigenvalue closest to J if external $= 1$
- approximation $\underline{\mathbf{x}} \in \mathbb{R}^n \setminus \{\underline{\mathbf{0}}\}$ of an associated eigenvector

initialize $\lambda(B) =$ "arbitrary", $\underline{\mathbf{x}^{\text{old}}} = \underline{\mathbf{x}^{\mathbf{0}}} \,/\, \left|\underline{\mathbf{x}^{\mathbf{0}}}\right|$, external $= 1$, doRQI $= 0$;

while *stopping criterion not satisfied* **do**

  if doRQI $= 1$ **then**

    set $\mu = \lambda(B)$; `// do Rayleigh quotient iteration (RQI)`

  else

    set $\mu = \gamma$; `// do inverse iteration with shift (II)`

  end

  solve the following system for $\underline{\mathbf{y}}$ using the linear solver object:

$$(B - \mu\,\mathcal{I})\underline{\mathbf{y}} = \underline{\mathbf{x}^{\text{old}}}\,;$$

  normalize $\underline{\mathbf{y}}$ by setting $\omega = 1 \,/\, \left|\underline{\mathbf{y}}\right|$ and $\underline{\mathbf{x}} = \omega\,\underline{\mathbf{y}}$;

  set $\lambda^{\text{old}}(B) = \lambda(B)$ and compute $\lambda(B) = \text{rq}_B(\underline{\mathbf{x}})$;

  if doRQI $= 0$ & $\omega < \eta$ **then**

    set external $= 0$ and doRQI $= 1$; `// activate RQI`

  end

  if external $= 0$ & doRQI $= 1$ & $\lambda(B) \notin J$ **then**

    set doRQI $= 0$; `// reactivate II`

  end

  if *at least* $m \geq 2$ *iterations have been performed* **then**

    if external $= 1$ & doRQI $= 0$ & $|\lambda(B) - \lambda^{old}(B)| \,/\, |\lambda(B)| < \delta$ **then**

      set doRQI $= 1$; `// activate RQI`

    end

  end

  set $\underline{\mathbf{x}^{\text{old}}} = \underline{\mathbf{x}}$;

end

The TLIME algorithm computes approximations of an eigenvalue in a given interval $J := (\gamma - \eta, \gamma + \eta)$ and an associated eigenvector. It guarantees that, if an eigenvalue exists in J, the method will converge to one of the eigenvalues in J, while exploiting the cubic convergence of the Rayleigh quotient iteration. When J is free of eigenvalues, the method will determine this fact and converge to the eigenvalue closest to J. In this case the rate of convergence is dominated by the linear convergence of the inverse iteration method, but the Rayleigh quotient iteration is still used to finally accelerate convergence.

In particular, the TLIME algorithm starts performing inverse iteration with a shift which is given by the parameter γ that specifies the center of the interval J. It then employs two criteria for switching to Rayleigh quotient iteration and one criterion for switching back to inverse iteration. To explain these criteria, let the generated sequence of approximate eigenvalue–eigenvector pairs of B be denoted by $\left(\lambda^k(B), \underline{\mathbf{y^k}}\right)_k$, where each $\underline{\mathbf{y^k}}$ is the approximate eigenvector obtained right before the normalization step in Algorithm 4.

The first criterion for switching to Rayleigh quotient iteration is based on an inclusion theorem. This theorem guarantees the existence of at least one eigenvalue in J as soon as the sequence $(\omega^k)_k$ of normalizers $\omega^k := 1 \,/\, \left|\underline{\mathbf{y^k}}\right|$, which is monotonically decreasing, is smaller than the parameter η that specifies the width of the interval J. Then, it is furthermore known that the current approximate eigenvalue $\lambda^k(B)$ lies in J. See Szyld (1988) for details. In this case, the algorithm switches and tries to use the Rayleigh quotient iteration to converge to one of the eigenvalues in J. A criterion to switch back to the inverse iteration method is required since it can nevertheless happen that the current approximate eigenvector $\underline{\mathbf{y^k}}$ and its normalized variant are not close enough to an eigenvector associated with an eigenvalue in J. Note that, in this case, the Rayleigh quotient iteration might still converge to an eigenvalue of B which does not lie in J. Accordingly, the TLIME algorithm performs a switch back to inverse iteration if the approximate eigenvalue $\lambda^k(B)$ falls outside J at some point.

If the criterion for switching to Rayleigh quotient iteration mentioned above is never satisfied, then there is no eigenvalue in J. In this case, there is no alternative to sticking with inverse iteration if convergence to the eigenvalue closest to γ shall be guaranteed. Nevertheless, the method switches to Rayleigh quotient iteration at some point, but solely to accelerate convergence to this eigenvalue in the final iterations. To assure convergence to the right eigenvalue, inverse iteration is performed until the process becomes stationary. The process is considered as being stationary as soon as the relative change $|\lambda^k(B) - \lambda^{k-1}(B)| \,/\, |\lambda^k(B)|$ falls below a given threshold δ. To make sure that this switch is not triggered during the first iterations of the algorithm, caused by an initial vector which is close to an eigenvector that is associated with some eigenvalue outside J, a minimum amount of inverse iterations is performed before the criterion is activated.

For more information on the general formulation of the TLIME algorithm and its theoretical fundament, we refer to Szyld (1988).

B.4.5. Application to computing the spectral condition number

We close this section by briefly emphasizing how the eigenvalue algorithms which have been considered can be employed particularly for calculating the spectral condition number. As shown in Section B.3, it is possible to obtain the spectral condition number of a given matrix by computing the dominant eigenvalue and the least dominant eigenvalue of some associated Hermitian matrix. Hence, let the considered matrix $B \in \mathbb{C}^{n \times n}$ be this Hermitian matrix in the remainder of this section.

For computing the dominant eigenvalue $\lambda_{\mathrm{absmax}}(B) \in \mathbb{R}$ of B, each method can be used:

- The power iteration method in its basic formulation from Section B.4.1 converges to $\lambda_{\mathrm{absmax}}(B)$.
- The inverse iteration method with shift $\mu = c \in \mathbb{R}$ converges to $\lambda_{\mathrm{absmax}}(B)$ if $|c|$ is an upper limit of $\left|\lambda_{\mathrm{absmax}}(B)\right|$ and $\mathrm{sgn}(c) = \mathrm{sgn}\big(\lambda_{\mathrm{absmax}}(B)\big)$.
- If an initial vector is available that is sufficiently close to some eigenvector associated with $\lambda_{\mathrm{absmax}}(B)$, the Rayleigh quotient iteration method converges to $\lambda_{\mathrm{absmax}}(B)$. Although such an initial vector can be generated using one of the above methods, it is preferable to employ the TLIME algorithm.
- Provided that B is a matrix over $\mathbb{R}$, the TLIME algorithm for the interval $J = (\gamma - \eta, \gamma + \eta)$ with $\gamma = c$ and $\eta = 0$ converges to $\lambda_{\mathrm{absmax}}(B)$, if $|c|$ is an upper limit of $\left|\lambda_{\mathrm{absmax}}(B)\right|$ and $\mathrm{sgn}(c) = \mathrm{sgn}\big(\lambda_{\mathrm{absmax}}(B)\big)$.

An approximate upper limit $|c|$ which is cheap to compute for sparse matrices is the infinity norm $\|B\|_\infty$, which corresponds to the maximum absolute row sum of the matrix. More generally, we have $\left|\lambda_{\mathrm{absmax}}(B)\right| \leq \|B\|$ for the matrix norm induced by any vector norm on $\mathbb{C}^n$. Note that $\underline{\mathbf{x}} \neq \underline{\mathbf{0}}$ and

$$\left|\lambda_{\mathrm{absmax}}(B)\right| \|\underline{\mathbf{x}}\| = \left\|\lambda_{\mathrm{absmax}}(B)\, \underline{\mathbf{x}}\right\| = \|B\underline{\mathbf{x}}\| \leq \|B\|\, \|\underline{\mathbf{x}}\|$$

for an arbitrary vector norm $\|\cdot\|$ on $\mathbb{C}^n$ and every eigenvector $\underline{\mathbf{x}}$ associated with $\lambda_{\mathrm{absmax}}(B)$. Knowledge about $\mathrm{sgn}\big(\lambda_{\mathrm{absmax}}(B)\big)$ is required to choose the correct sign of c.

The least dominant eigenvalue $\lambda_{\mathrm{absmin}}(B) \in \mathbb{R}$ of B is the eigenvalue closest to 0. Therefore, it can be obtained similarly by:

- the inverse iteration method with shift $\mu = 0$,
- the Rayleigh quotient iteration method, if an initial vector is available that is sufficiently close to some eigenvector associated with $\lambda_{\mathrm{absmin}}(B)$,
- the TLIME algorithm for the interval $J = (\gamma - \eta, \gamma + \eta)$ with $\gamma = 0$ and $\eta = 0$, provided that B is a matrix over $\mathbb{R}$.

Concrete combinations of those eigenvalue algorithms, which we have practically used in this thesis to calculate the spectral condition number of matrices, will be considered in the next section.

B.5. Implementation in the `dune-istl` module

As part of this thesis, the eigenvalue algorithms from Section B.4, an interface to a library of other eigenvalue algorithms and, built on top of both, an easy-to-use facility for computing the spectral condition number of a matrix have been implemented. We did this for being able to analyze those matrices which arise from the discretization methods introduced in Chapter 4. The code has been built into the modular toolbox for numerical software development named `DUNE`, which is introduced in Appendix A.1. Although this is currently not stated in the release notes, the code is part of the module `dune-istl` since the release of `DUNE` 2.5.

In particular, the power iteration based eigenvalue algorithms that are introduced in Section B.4 have been implemented as a class template named `Dune::PowerIteration_Algorithms`, whose structure is depicted in Table B.1. The implementation is restricted to square, symmetric matrices over $\mathbb{R}$ that are represented by objects of type `Dune::BCRSMatrix`, which is the standard type for sparse block matrices in `dune-istl`, and vectors over $\mathbb{R}$ that are represented by compatible objects of type `Dune::BlockVector` from `dune-istl`. As vector norm $\|\cdot\|$ for Algorithm 1, Algorithm 2 and Algorithm 3, the implementation uses the Euclidean norm $|\cdot|$. This norm is also employed by the stopping criterion that is implemented in `Dune::PowerIteration_Algorithms`. The algorithms stop iterating as soon as the norm of the absolute residual falls below a threshold $\varepsilon > 0$, i.e. once that $\left|B\underline{\mathbf{x}^{\mathbf{k}}} - \lambda^k(B)\,\underline{\mathbf{x}^{\mathbf{k}}}\right| \leq \varepsilon$, or if a maximum number of iterations has been performed. Here, B is the given matrix and $\lambda^k(B)$ as well as $\underline{\mathbf{x}^{\mathbf{k}}}$ are the current iterates of the approximate eigenvalue and the associated eigenvector, respectively. The threshold ε needs to be specified in each algorithm call, whereas a maximum number of iterations per algorithm call can be set globally via an optional parameter of the constructor.

To compute the Rayleigh quotient and the residual norm efficiently, the implementation uses the result of matrix–vector multiplications $B\underline{\mathbf{x}}$ or solutions of linear systems that have already been computed in the essential step of the algorithms. Recycling the solutions of linear systems for this purpose can be turned off using a compile-time switch, to allow for avoiding that inaccurate solutions have a negative impact on the computation of eigenvalues and their associated residuals. Setting this mode can help increasing the accuracy of each linear solver based algorithm at the cost of a bit of efficiency. This is beneficial, e.g., when using a very inexact linear solver.

`Dune::PowerIteration_Algorithms` is compatible with all iterative linear solvers and direct linear solvers for sparse matrices that are available in `dune-istl`. Linear solvers that operate matrix free (e.g. iterative solvers of Krylov type with appropriate preconditioners) need to be set up with the operator retrieved by the class template's method `getIterationOperator`. Linear solvers which require the matrix (e.g. direct solvers) need to be set up using the method `getIterationMatrix`.

Entity / class member	Description
`BCRSMatrix`	Template parameter selecting the specific type of `Dune::BCRSMatrix`
`BlockVector`	Template parameter selecting the specific type of `Dune::BlockVector`
`Real`	Type used for representing $\mathbb{R}$
`IterationOperator`	Type used for representing the iteration operator $(B - \mu\mathcal{I})$
`PowerIteration_Algorithms (const BCRSMatrix&)`	Constructs the class for a given square, symmetric matrix $B \in \mathbb{R}^{n\times n}$
`void applyPowerIteration (const Real&, BlockVector&, Real&)`	Performs Algorithm 1
`template<typename ISTLLinearSolver> void applyInverseIteration (const Real&, ISTLLinearSolver&, BlockVector&, Real&)`	Performs Algorithm 2 for $\mu = 0$
`template<typename ISTLLinearSolver> void applyInverseIteration (const Real&, const Real&, ISTLLinearSolver&, BlockVector&, Real&)`	Performs Algorithm 2
`template<typename ISTLLinearSolver> void applyRayleighQuotientIteration (const Real&, ISTLLinearSolver&, BlockVector&, Real&)`	Performs Algorithm 3 (one iteration of Algorithm 2 is performed beforehand in this specific implementation)
`template<typename ISTLLinearSolver> void applyTLIMEIteration (const Real&, const Real&, const Real&, ISTLLinearSolver&, const Real&, const std::size_t&, bool&, BlockVector&, Real&)`	Performs Algorithm 4
`IterationOperator& getIterationOperator ()`	Retrieves the iteration operator $(B-\mu\mathcal{I})$
`const BCRSMatrix& getIterationMatrix ()`	Retrieves the iteration operator $(B-\mu\mathcal{I})$ as a matrix that is provided on demand, when needed (e.g. for direct solvers or preconditioning)
`unsigned int getIterationCount ()`	Number of iterations performed in the last application of an algorithm

Table B.1.: Overview of class template `Dune::PowerIteration_Algorithms`.

In addition to the eigenvalue algorithms from Section B.4, wrappers have been implemented and built into `dune-istl` that allow for computing the extremal eigenvalues of a real, symmetric matrix or the singular values of an arbitrary real matrix, by calling an eigenvalue algorithm that is provided by a C++ library called `ARPACK++`[1]. This library comprises a collection of C++ bindings to a FORTRAN 77 library called `ARPACK`[2], see also Lehoucq et al. (1998a). The latter provides an implementation of eigenvalue algorithms which are known as the implicitly restarted Arnoldi method (IRAM) (Lehoucq and Sorensen, 1996; Lehoucq et al., 1998b) and the implicitly restarted Lanczos method (IRLM) (Calvetti et al., 1994), where the IRLM is a variant of the IRAM for symmetric matrices. These methods can be seen as a synthesis of the Arnoldi/Lanczos process with the implicitly shifted QR algorithm. While the QR algorithm is only suitable for eigenvalue problems with a dense matrix, the IRAM/IRLM were designed to compute eigenvalue–eigenvector pairs of large, sparse, nonsymmetric/symmetric matrices.

The main wrapper that allows for employing `ARPACK++`/`ARPACK` to compute the extremal eigenvalues or singular values of a matrix has been implemented as a class template `Dune::ArPackPlusPlus_Algorithms`. Its structure is depicted in Table B.2. In each call to a method that performs computations, a stopping tolerance needs to be specified which configures the target accuracy of the result. Furthermore, since we use the IRLM which is an iterative scheme, a parameter which configures the maximum number of iterations per algorithm call can be set globally via an optional parameter of the constructor of the class template. Internally, `Dune::ArPackPlusPlus_Algorithms` uses a class template `Dune::ArPackPlusPlus_BCRSMatrixWrapper`. The latter is a wrapper for a matrix $A \in \mathbb{R}^{m \times n}$ that is represented by an object of type `Dune::BCRSMatrix`. It provides methods which implement the matrix–vector products $A\underline{\mathbf{x}}$ and $A^{\mathrm{tr}}A\underline{\mathbf{x}}$ with a signature that is usable by `ARPACK++`. For the particular variant of the eigenvalue algorithm which is employed, these are the only required operations involving A.

An easy-to-use facility which allows for straightforward computation of the spectral condition number of a matrix has been implemented as a class template `MatrixInfo`. Its structure is presented in Table B.3. The underlying algorithm can be described as depicted in Algorithm 5.

To perform step 2 in Algorithm 5, `MatrixInfo` uses the class template `Dune::PowerIteration_Algorithms` which is described above. In particular, the approximation $\lambda_{\mathrm{absmax}}(B)$ is computed using a combination of the power iteration method (in its basic formulation from Section B.4.1) and the TLIME algorithm. Here, the power iteration method serves as a predictor for the sign of $\lambda_{\mathrm{absmax}}(B)$ and for an eigenvector. The predicted eigenvector is employed as initial vector for the TLIME algorithm. To obtain $\lambda_{\mathrm{absmax}}(B)$, the latter is configured as described in Section B.4.5, using the predicted sign

[1] `http://www.ime.unicamp.br/~chico/arpack++/`
[2] `http://www.caam.rice.edu/software/ARPACK/`

Entity / class member	Description
`BCRSMatrix`	Template parameter selecting the specific type of `Dune::BCRSMatrix`
`BlockVector`	Template parameter selecting the specific type of `Dune::BlockVector`
`Real`	Type used for representing $\mathbb{R}$
`ArPackPlusPlus_Algorithms (const BCRSMatrix&)`	Constructs the class for a given matrix $A \in \mathbb{R}^{m \times n}$
`void computeSymMaxMagnitude (const Real&, BlockVector&, Real&)`	Assumes A to be square, symmetric and performs IRLM to compute its dominant eigenvalue $\lambda_{\mathrm{absmax}}(A) \in \mathbb{R}$ and an associated eigenvector
`void computeSymMinMagnitude (const Real&, BlockVector&, Real&)`	Assumes A to be square, symmetric and performs IRLM to compute its least dominant eigenvalue $\lambda_{\mathrm{absmin}}(A) \in \mathbb{R}$ and an associated eigenvector
`void computeNonSymMax (const Real&, BlockVector&, Real&)`	Assumes A to be nonsymmetric and performs IRLM on $A^{\mathrm{tr}}A$ to compute the largest singular value $\sigma_{\max}(A) \in \mathbb{R}$ of A and an associated singular vector
`void computeNonSymMin (const Real&, BlockVector&, Real&)`	Assumes A to be nonsymmetric and performs IRLM on $A^{\mathrm{tr}}A$ to compute the smallest singular value $\sigma_{\min}(A) \in \mathbb{R}$ of A and an associated singular vector
`unsigned int getIterationCount ()`	Number of iterations performed in the last application of an algorithm

Table B.2.: Overview of class template `Dune::ArPackPlusPlus_Algorithms`.

Entity / class member	Description
`BCRSMatrix`	Template parameter selecting the specific type of `Dune::BCRSMatrix`
`Real`	Type used for representing $\mathbb{R}$
`MatrixInfo (const BCRSMatrix& A)`	Constructs the class for a given square matrix $A \in \mathbb{R}^{n \times n}$
`Real getCond2 (const bool assume_symmetric)`	Computes $\kappa_2(A)$ (if it is not yet available) using Algorithm 5 and retrieves $\kappa_2(A)$

Table B.3.: Overview of class template `MatrixInfo`.

Algorithm 5: Computation of $\kappa_2(A)$

Input: square matrix $A \in \mathbb{R}^{n \times n}$, flag symmetric $\in \{0, 1\}$ indicating if symmetry of A can be assumed or not

Output: approximation $\kappa_2(A)$ of the spectral condition number of A

if symmetric $= 1$ **then** `// step 1`
  set $B = A$;
else
  compute $A^{\mathrm{tr}}A$ and set $B = A^{\mathrm{tr}}A$;
end

compute approximations $\lambda_{\mathrm{absmax}}(B)$ and $\lambda_{\mathrm{absmin}}(B)$; `// step 2`

if symmetric $= 1$ **then** `// step 3`
  set $\sigma_{\max}(A) = \left|\lambda_{\mathrm{absmax}}(B)\right|$ and $\sigma_{\min}(A) = \left|\lambda_{\mathrm{absmin}}(B)\right|$ according to formula (B.5);
else
  using that $\lambda_{\mathrm{absmax}}(B) \mathrel{\widehat{=}} \lambda_{\max}(B)$ and $\lambda_{\mathrm{absmin}}(B) \mathrel{\widehat{=}} \lambda_{\min}(B)$, set $\sigma_{\max}(A) = \sqrt{\lambda_{\mathrm{absmax}}(B)}$ and $\sigma_{\min}(A) = \sqrt{\lambda_{\mathrm{absmin}}(B)}$ according to formula (B.4);
end

set $\kappa_2(A) = \sigma_{\max}(A)$ / $\sigma_{\min}(A)$; `// step 4`

of $\lambda_{\mathrm{absmax}}(B)$ and the infinity norm $\|B\|_\infty$ as a cost-efficient upper limit of $\left|\lambda_{\mathrm{absmax}}(B)\right|$. The approximation $\lambda_{\mathrm{absmin}}(B)$ is computed directly using the TLIME algorithm. Again, see Section B.4.5. As a linear solver which drives the TLIME algorithm, the `dune-istl` interface to `SuperLU` is used.

On machines where the `ARPACK` library is available, `MatrixInfo` uses `Dune::ArPackPlusPlus_Algorithms` to directly compute $\sigma_{\max}(A)$ instead of $\lambda_{\mathrm{absmax}}(B)$ in steps 2 and 3 of Algorithm 5. This has shown to be a faster way of obtaining $\sigma_{\max}(A)$ for the set of matrices which we dealt with, while yielding similar results in terms of accuracy. However, in our experience, `Dune::ArPackPlusPlus_Algorithms` has problems finding eigenvalues in the lower end of the spectrum of matrices. At least for the set of matrices which we dealt with, it was not a reliable alternative in computing $\sigma_{\min}(A)$. The particular variant of the IRLM which is employed only converged for small matrices that result from very coarse meshes, while the TLIME algorithm even converged for most of the large matrices that result from fine meshes. For this reason, `MatrixInfo` does not employ `Dune::ArPackPlusPlus_Algorithms` to obtain $\sigma_{\min}(A)$. Even on machines where `ARPACK` is available, it sticks to `Dune::PowerIteration_Algorithms` to gain $\sigma_{\min}(A)$ by performing steps 2 and 3 of Algorithm 5, as described above.

It is worth noting that `MatrixInfo` is available via the header include `#include <dune/istl/eigenvalue/test/matrixinfo.hh>`, since it has been

built into `dune-istl` to facilitate testing. Its main purpose, however, is to be deployed in user code. It has been used to obtain all results regarding the spectral condition number of those matrices which arise from the discretization methods introduced in Chapter 4.

C. Basic terminology and facts from elementary differential geometry

In this appendix, we recall some basic terminology and facts from elementary differential geometry that we are using in the main part of this thesis. In Section C.1, we introduce the notion of C^k-hypersurfaces. In Section C.2, we briefly explain smoothness assumptions which we use throughout the thesis.

The content of Section C.1 is taken from Dziuk and Elliott (2013, Section 2). For general introductions to differentiable manifolds and hypersurfaces, please refer to Lee (2009, 2012) and to Burstall (1999, Section 1).

C.1. Hypersurfaces

Hypersurfaces can be defined via parametrizations in the following way.

Definition C.1.1 (Parametrized C^k-hypersurface (cf. Dziuk and Elliott, 2013, Section 2.1))**.** Let $k \in \mathbb{N} \cup \{\infty\}$. We call $\mathcal{M} \subset \mathbb{R}^d$ a $(d-1)$-dimensional *parametrized C^k-hypersurface* if, for every point $\underline{\mathbf{x_0}} \in \mathcal{M}$, there exists an open set $U \subset \mathbb{R}^d$ containing $\underline{\mathbf{x_0}}$, an open connected set $V \subset \mathbb{R}^{d-1}$, and a map $\mathbf{X}\colon V \to \mathcal{M} \cap U$ with the following properties:

- $\mathbf{X} \in C^k\big(V; \mathbb{R}^d\big)$,
- $\mathbf{X}$ is bijective,
- $\operatorname{rank} D\mathbf{X} = d-1$ on V.

The map $\mathbf{X}$ is called a *local parametrization* of $\mathcal{M}$, while $\mathbf{X}^{-1}$ is called a *local chart*. A collection $(\mathbf{X_i})_{i\in I}$ of local parametrizations $\mathbf{X_i} \in C^k\big(V_i; \mathbb{R}^d\big)$ with $\bigcup_{i\in I} \mathbf{X_i}(V_i) = \mathcal{M}$ is called a *C^k-atlas*. If $\mathbf{X_i}(V_i) \cap \mathbf{X_j}(V_j) = \emptyset$, then the map ${\mathbf{X_i}}^{-1} \circ \mathbf{X_j}$, by assumption, is a C^k-diffeomorphism.

A second, alternative definition by means of level sets of scalar functions is given as follows.

Definition C.1.2 (C^k-hypersurface (Dziuk and Elliott, 2013, Definition 2.1))**.** Let $k \in \mathbb{N}\cup\{\infty\}$. A set $\mathcal{M} \subset \mathbb{R}^d$ is called a $(d-1)$-dimensional *C^k-hypersurface* if, for each point $\underline{\mathbf{x_0}} \in \mathcal{M}$, there exists an open set $U \subset \mathbb{R}^d$ containing $\underline{\mathbf{x_0}}$ and a function $\Phi \in C^k(U)$ with the property that

$$\mathcal{M} \cap U = \{\underline{\mathbf{x}} \in U \mid \Phi(\underline{\mathbf{x}}) = 0\} \qquad \text{and} \qquad \nabla\Phi \neq 0 \quad \text{on} \quad \mathcal{M} \cap U.$$

Definition C.1.3 (Orientable C^k-hypersurface (cf. Dziuk and Elliott, 2013, Section 2.2))**.** A C^1-hypersurface $\mathcal{M} \subset \mathbb{R}^d$ is called *orientable* if there exists a continuous vector field $\boldsymbol{\nu}_{\mathcal{M}} \colon \mathcal{M} \to \mathbb{R}^d$ such that $\boldsymbol{\nu}_{\mathcal{M}}(\underline{\mathbf{x}})$ is a unit normal vector to $\mathcal{M}$ for all $\underline{\mathbf{x}} \in \mathcal{M}$.

A connection between parametrized hypersurfaces and hypersurfaces in the sense of Definition C.1.2 is given by the following lemma.

Lemma C.1.4 (Consistency of Definition C.1.2 and Definition C.1.1 (Dziuk and Elliott, 2013, Lemma 2.2))**.** *Assume that $\mathcal{M}$ is a C^k-hypersurface in $\mathbb{R}^d$. Then for every $\underline{\mathbf{x}} \in \mathcal{M}$ there exists an open set $U \subset \mathbb{R}^d$ with $\underline{\mathbf{x}} \in U$ and a parametrized C^k-hypersurface $\mathbf{X} \colon V \to \mathcal{M} \cap U$ such that $\mathbf{X}$ is a bijective map from V onto $\mathcal{M} \cap U$. If $\mathbf{X} \colon V \to \mathcal{M} \cap U$ is a parametrized C^k-hypersurface and $\theta \in V$, then there is an open set $\tilde{V} \subset V$ with $\theta \in \tilde{V}$ such that $\mathbf{X}(\tilde{V})$ is a C^k-hypersurface.*

Lemma C.1.4 implies that we can locally always work with hypersurfaces in the sense of Definition C.1.2. Moreover, since both definitions can be used interchangeably, notions from both definitions can be used without reference to the definition that has actually been used to introduce them.

C.2. Smoothness assumptions

Following Dziuk and Elliott (2013, Section 2.1 and Section 2.2), the notion of a C^k-atlas from Definition C.1.1 can be employed to define the notion of an l-times differentiable function on a hypersurface $\mathcal{M}$, provided that $\mathcal{M}$ is a C^k-hypersurface with $k \geq l$.

In the main part of this thesis, we deal with l-times differentiable functions or $C^l\big(\mathcal{M}\big)$-functions on some hypersurface $\mathcal{M}$, where $l \leq 2$. It would therefore be sufficient to assume that the given hypersurfaces are C^1-hypersurfaces or C^2-hypersurfaces, depending on the features of the equations which we are dealing with. Keeping this in mind, for the sake of simplicity, we avoid precise assumptions on the regularity of the hypersurfaces which we are dealing with in the main part of this thesis. We could abstractly require that these hypersurfaces are *sufficiently smooth* for what we are doing, leaving actual smoothness considerations up to the reader. Since we do not want our assumptions to distract from the essentials, we simply assume that *smooth hypersurfaces*, i.e., C^∞-hypersurfaces are given.

List of Symbols

Symbol	Description
$\mathbb{C}$	set of complex numbers
$\mathbb{N}$	set of natural numbers
$\mathbb{R}$	set of real numbers
$\mathbb{R}^{>0}$	the positive numbers in $\mathbb{R}$
$\mathbb{R}^{\geq 0}$	the non-negative numbers in $\mathbb{R}$
$\mathbb{R}^d$	real coordinate space of dimension d which models the ambient Cartesian space
$\diamond$	symbol which is used as a wildcard
$\boldsymbol{\diamond}$	bold, upright symbols denote some vector field
$\underline{\boldsymbol{\diamond}}$	underlined, bold, upright symbols denote some constant vector
$\emptyset$	the empty set
$\equiv$	symbol for indicating that an entity (e.g. a scalar/vector field or a time-dependent set) is equivalent to a constant entity of the same kind
$\subset$	symbol for indicating that a set is a subset of another set, usually a proper subset: $\mathfrak{A} \subset \mathfrak{B} :\Leftrightarrow \mathfrak{A} \subseteq \mathfrak{B} \wedge \mathfrak{A} \neq \mathfrak{B}$; see also $\subseteq$
$\subseteq$	symbol for indicating that a set is a subset of another set: $\mathfrak{A} \subseteq \mathfrak{B} :\Leftrightarrow \forall x \in \mathfrak{A} \colon x \in \mathfrak{B}$; see also $\subset$
Δ	classical Laplace operator ("the Laplacian") in ambient Cartesian space
$\Delta_{\mathcal{M}}$	Laplace–Beltrami operator on some hypersurface $\mathcal{M}$, such as Γ
∇	transposed classical gradient operator in ambient Cartesian space, see also $\nabla \cdot$
$\nabla_{\mathcal{M}}$	transposed surface gradient operator on some hypersurface $\mathcal{M}$, such as Γ; see also $\nabla_{\mathcal{M}} \cdot$
$\nabla \cdot$	classical divergence operator in ambient Cartesian space, see also ∇
$\nabla_{\mathcal{M}} \cdot$	surface divergence operator on some hypersurface $\mathcal{M}$, such as Γ; see also $\nabla_{\mathcal{M}}$

Symbol	Description
∇_h	piecewise variant of the transposed classical gradient operator in ambient Cartesian space, as used in DG schemes; see also ∇_{fl}
∇_{fl}	piecewise variant of the transposed classical gradient operator in ambient Cartesian space, for piecewise (multi-)linear continuous functions over the geometry mesh $\mathcal{T}_{\mathit{fl}}(\Omega_\Phi)$; see also ∇_h
∂_i	differential operator taking the i-th partial derivative in ambient Cartesian space, see also $\frac{\partial}{\partial x_i}$
$\partial_i^{\mathcal{M}}$	differential operator taking the i-th surface partial derivative on some hypersurface $\mathcal{M}$
$\frac{\partial}{\partial x_i}$	differential operator taking the i-th partial derivative in ambient Cartesian space, see also ∂_i
∂_t	differential operator taking the partial derivative with respect to time
$\frac{\mathrm{d}}{\mathrm{d}t}$	differential operator taking the derivative with respect to time, can be applied to functions that solely depend on time
$\partial^\bullet$	differential operator taking the material derivative of scalar fields on evolving hypersurfaces
$\|\cdot\|_{H^1(\Omega_{\mathit{fl}})}$	bulk H^1-norm on the discretized geometry
$\|\cdot\|_{H^1(\Gamma_{\mathit{fl}})}$	surface H^1-norm on the discretized geometry
$\|\cdot\|_{L^2(\Omega_{\mathit{fl}})}$	bulk L^2-norm on the discretized geometry
$\|\cdot\|_{L^2(\Gamma_{\mathit{fl}})}$	surface L^2-norm on the discretized geometry
$\lvert\cdot\rvert$	Euclidean norm, also known as 2-norm
$\lvert\cdot\rvert$	absolute value of a real number
$\dot\cup$	$\mathfrak{A}\,\dot\cup\,\mathfrak{B}$ denotes the union of two disjoint sets $\mathfrak{A}$ and $\mathfrak{B}$
$\mathrm{cl}\,(\mathfrak{A})$	closure of a set $\mathfrak{A}$
$\mathfrak{A}^{\mathrm{c}}$	absolute complement of a set $\mathfrak{A}$ in some fixed universe
$\mathrm{diam}(\mathfrak{A})$	diameter of a set $\mathfrak{A}$
$\mathrm{int}\,(\mathfrak{A})$	interior of a set $\mathfrak{A}$
$\mathrm{meas}_{\mathbb{R}^d}(\mathfrak{A})$	d-dimensional measure of a set $\mathfrak{A}$
$\mathcal{N}(\mathfrak{A})$	d-dimensional neighborhood of a subset $\mathfrak{A}$ of the ambient Cartesian space
$\{\cdot\}$	average operator, as used in DG methods and defined in Section 3.2.1, Section 4.2.2, and Section 5.3.2

Symbol	Description
$\{ \cdot \}_\omega$	weighted average operator, as used in the SWIPG formulation and defined in Section 3.2.3
$[\![\cdot]\!]$	jump operator, as used in DG methods and defined in Section 3.2.1, Section 4.2.2, and Section 5.3.2
$\diamond^\Phi$	extension of a datum $\diamond$ to the level set domain Ω_Φ
$\diamond^{\text{ext}}$	extension of a datum $\diamond$ to the surface extension domain Ω_{ext}
$\tilde{\diamond}^{\text{ext}}$	modified extended data functions, as defined in equation (4.9)
$\mathbb{1}_{\mathfrak{A}}$	the characteristic function (indicator function) of a subset $\mathfrak{A}$ of the ambient Cartesian space
A	usually denotes some matrix in $\mathbb{R}^{m\times n}$, with $m, n \in \mathbb{N}$, which represents a system of linear equations $A\underline{\mathbf{x}} = \underline{\mathbf{b}}$
A^*	conjugate transpose of a vector/matrix A
A^{tr}	transpose of a vector/matrix A
α	usually denotes the uniform scaling parameter which is used if the scaling parameters α_{in} and α_{out} of the narrow band Ω_δ are chosen to be equal
α_{in}	scaling parameter of the narrow band Ω_δ, see also α and δ_{in}
α_{out}	scaling parameter of the narrow band Ω_δ, see also α and δ_{out}
B	usually denotes some matrix in $\mathbb{C}^{n\times n}$, $n \in \mathbb{N}$
$\underline{\mathbf{b}}$	usually denotes some vector in $\mathbb{R}^m$, $m \in \mathbb{N}$, which is the right-hand side in a system of linear equations $A\underline{\mathbf{x}} = \underline{\mathbf{b}}$
$C^0\big(\mathfrak{A};\mathfrak{B}\big)$	space of continuous functions on a domain $\mathfrak{A}$, mapping into a target set $\mathfrak{B}$
$C^0\big(\mathfrak{A}\big)$	the space $C^0\big(\mathfrak{A};\mathbb{R}\big)$
$C^k\big(\mathfrak{A};\mathfrak{B}\big)$	space of k-times continuously differentiable functions on a domain $\mathfrak{A}$, mapping into a target set $\mathfrak{B}$
$C^k\big(\mathfrak{A}\big)$	the space $C^k\big(\mathfrak{A};\mathbb{R}\big)$

Symbol	Description
$\mathcal{D}_b$	diffusivity tensor of a bulk PDE, see Section 1.2
$\mathcal{D}_s$	diffusivity tensor of a surface PDE, see Section 1.2
$\mathcal{D}$	usually denotes some bulk domain or open set
D	static bulk domain that is used in our reformulation approach from Section 5.3.1; results from projecting the space–time representation of $\Gamma(t)$ to spatial-only coordinates
$D_\hbar$	discrete reconstruction of the bulk domain D
$\mathfrak{D}$	some bulk domain which contains the reconstructed bulk domain $D_\hbar$
$D\mathbf{f}$	Jacobian matrix of a vector-valued function $\mathbf{f}$
δ	width $\delta := \delta_{\text{in}} + \delta_{\text{out}}$ of the narrow band Ω_δ, measured in the codomain of the level set function Φ, i.e., in "level set levels"
δ_{in}	scaling parameter $\delta_{\text{in}} := \alpha_{\text{in}} \cdot h$ of the narrow band Ω_δ, as defined in Section 4.2.2
δ_{out}	scaling parameter $\delta_{\text{out}} := \alpha_{\text{out}} \cdot h$ of the narrow band Ω_δ, as defined in Section 4.2.2
d	dimension of the ambient Cartesian space
$\mathrm{d}x$	differential in spatial integrals of codimension 0, also known as d-dimensional volume measure
$\mathrm{d}\sigma$	differential in spatial integrals of codimension 1, also known as $(d-1)$-dimensional surface measure
$\mathrm{d}\varsigma$	differential in spatial integrals of codimension 2, also known as $(d-2)$-dimensional surface measure
$\mathrm{d}t$	differential in time integrals
E	some face of a cut cell or classical mesh element
$\mathcal{E}_h^{\text{int}}(\mathcal{D})$	internal skeleton of the (cut cell) mesh $\mathcal{T}_h(\mathcal{D})$
$f_b(u_b)$	bulk reaction term of a bulk–surface model, see Section 1.2
$f_{b,s}(u_b, u_s)$	bulk coupling term of a bulk–surface model, see Section 1.2
$f_s(u_s)$	surface reaction term of a bulk–surface model, see Section 1.2
$f_{s,b}(u_b, u_s)$	surface coupling term of a bulk–surface model, see Section 1.2

Symbol	Description
g_b	scalar field which represents some source/sink density defined in Ω
g_s	scalar field which represents a source/sink density defined on some hypersurface $\mathcal{M}$, such as Γ
γ_b	interior penalty parameter for the bulk part of the problem
γ_s	interior penalty parameter for the surface part of the problem
$\gamma_{\mathcal{D}}$	diffusivity-dependent scaling function, as used in the SWIPG formulation and defined in Section 3.2.3
γ_{np}	parameter of the normal penalty stabilization mechanism introduced in Section 4.2.4
Γ	$(d-1)$-dimensional hypersurface, denoted by $\Gamma(t)$ if it is time-dependent, usually the boundary of a bulk domain Ω
$\Gamma_\hbar$	discrete reconstruction of the hypersurface Γ
Γ_l	level set of Φ associated with the level $l \in \mathbb{R}$, note that $\Gamma_0 = \Gamma$
$\Gamma_{l,\hbar}$	level set of $\Phi_\hbar$ associated with the level $l \in \mathbb{R}$, note that $\Gamma_{0,\hbar} = \Gamma_\hbar$
h	width (i.e. maximum element size) of the fundamental mesh $\mathcal{T}_h(\Omega_\Phi)$ or of the mesh that is employed by some fitted method
h_E	local mesh size at a face E of a cut cell or classical mesh element
$\hbar$	width (i.e. maximum element size) of the geometry mesh $\mathcal{T}_\hbar(\Omega_\Phi)$
$H_{\mathcal{M}}$	total curvature of some hypersurface $\mathcal{M}$ (see Definition 2.2.6), where $H := H_\Gamma$
$H^k(\mathcal{D})$	Sobolev spaces over some bulk domain or hypersurface $\mathcal{D}$
$\mathcal{I}$	identity operator on some space
k	the polynomial degree that is used for constructing some discrete finite element space
$\kappa_2(A)$	spectral condition number of a matrix A, i.e., the condition number with respect to the Euclidean norm $\lvert\cdot\rvert$
K	some cut cell or classical mesh element

Symbol	Description
K_E^-	adjacent element of an internal face E of a cut cell or classical mesh element
K_E^+	adjacent element of an internal face E of a cut cell or classical mesh element
$\lambda(B)$	eigenvalues of a square matrix B
$L^p(\mathcal{D})$	Lebesgue spaces over some bulk domain or hypersurface $\mathcal{D}$
m	usually denotes the number $m \in \mathbb{N}$ of equations in a system of linear equations
$m(t)$	amount of some quantity/quantities at a given time t, usually the total amount of quantities that are associated with a system of bulk–surface equations
$\mathcal{M}$	some hypersurface, such as Γ
$\mathcal{M}_l$	restricted level sets $\mathcal{M}_l := \Gamma_l \cap \Omega_{\text{ext}}$ of Φ in the surface extension domain Ω_{ext}, note that $\mathcal{M}_0 = \Gamma$
$\mathcal{M}_t$	space–time representation of an evolving hypersurface $\mathcal{M}(t)$
M	an arbitrary subset of Γ or of some related hypersurface $\mathcal{M}$
$\boldsymbol{\mu}_{\partial\mathcal{M}}$	field of intrinsic unit normal vectors to the boundary of some hypersurface $\mathcal{M}$
n	usually denotes the number $n \in \mathbb{N}$ of unknowns in a system of linear equations
$\mathbf{n}_{\partial\mathcal{D}}$	field of unit normal vectors to the boundary of some bulk domain $\mathcal{D}$
$\boldsymbol{\nu}_{\mathcal{M}}$	field of unit normal vectors to some hypersurface $\mathcal{M}$, where $\boldsymbol{\nu} := \boldsymbol{\nu}_{\Gamma}$
$\mathcal{O}$	Bachmann–Landau notation for describing the asymptotic behavior of functions: $f \in \mathcal{O}(g) :\Leftrightarrow$ f grows asymptotically no faster than g
Ω	d-dimensional bulk domain, denoted by $\Omega(t)$ if it is time-dependent
Ω_h	discrete reconstruction of the bulk domain Ω
Ω_δ	narrow band which is used as a surface extension domain Ω_{ext}
$\Omega_{\delta,h}$	discrete reconstruction of the narrow band Ω_δ which is used as a surface extension domain

Symbol	Description
Ω_{ext}	some general surface extension domain
Ω_Φ	level set domain, i.e., bulk domain which is used for our level set function Φ
$P_k(\mathcal{D})$	space of polynomial functions of total degree less than or equal to k over some domain $\mathcal{D}$
ϕ	azimuthal angle of polar coordinates (r, ϕ), for $d = 2$, or of spherical coordinates (r, θ, ϕ), for $d = 3$
$\mathcal{P}_\mathcal{M}$	tangential projection operator for some hypersurface $\mathcal{M}$, where $\mathcal{P} := \mathcal{P}_\Gamma$
Φ	level set function which is employed for describing the geometry
Φ_h	discrete analogue to the level set function Φ
$Q_k(\mathcal{D})$	space of polynomial functions over some domain $\mathcal{D}$, with a degree less than or equal to k in each coordinate direction
$\mathbf{q_b}$	vector field which represents some flux defined in Ω
$\mathbf{q_s}$	vector field which represents a flux defined on some hypersurface $\mathcal{M}$, such as Γ
r	radial distance of polar coordinates (r, ϕ), for $d = 2$, or of spherical coordinates (r, θ, ϕ), for $d = 3$
R	an arbitrary subset of Ω or of some related bulk domain
$\sigma(A)$	singular values of a matrix A
sgn	the signum function which yields the sign -1, 0 or 1 of a number
t	time variable
t^n	discrete time variable
T	maximum time in an observation period $[0, T]$
$\hat{\mathcal{T}}_h(\mathcal{D})$	active mesh for some bulk domain $\mathcal{D}$
$\mathcal{T}_h(\mathcal{D})$	cut cell mesh or classical mesh of some bulk domain $\mathcal{D}$
$\mathcal{T}_h(\Omega_\Phi)$	fundamental mesh that is employed in our UDG schemes, i.e., a mesh of the level set domain Ω_Φ

Symbol	Description
$\mathcal{T}_h(\Omega_\Phi)$	geometry mesh that is employed in our UDG schemes, i.e., a (usually fine) mesh of the level set domain Ω_Φ
τ^n	size of the n-th time step
θ	zenith angle (also known as inclination angle) of spherical coordinates (r, θ, ϕ) for $d = 3$
$\mathrm{tr}(A)$	the trace of a square matrix A
U	usually denotes some unitary matrix in $\mathbb{C}^{n\times n}$, with $n \in \mathbb{N}$
u_b	concentration of a scalar bulk quantity on a bulk domain Ω
$u_{b,h}$	discrete analogue to u_b
u_h	discrete solution in a discrete approximation space that is not further specified
u_s	concentration of a surface-bound scalar quantity on a hypersurface Γ
$u_{s,h}$	discrete analogue to u_s
$V_{b,h}(\mathcal{D})$	discrete approximation space associated with the bulk part of a problem, comprising functions over some bulk domain $\mathcal{D}$
$V_{s,h}(\mathcal{D})$	discrete approximation space associated with the surface part of a problem, comprising functions over a bulk domain or hypersurface $\mathcal{D}$
$\mathbf{v_b}$	field of material velocity vectors of $\Omega(t)$
$\mathbf{v_s}$	field of material velocity vectors of $\Gamma(t)$
$\mathbf{v}$	field of material velocity vectors of $\Omega(t) \cup \Gamma(t)$
φ_b	test function in some space associated with the bulk part of a problem
$\varphi_{b,h}$	discrete analogue to φ_b, i.e., test function in a discrete approximation space associated with the bulk part of a problem
φ_h	test function in a discrete approximation space that is not further specified
φ_s	test function in some space associated with the surface part of a problem
$\varphi_{s,h}$	discrete analogue to φ_s, i.e., test function in a discrete approximation space associated with the surface part of a problem
$\mathbf{w_b}$	field of velocity vectors that are intrinsic to Ω
$\mathbf{w_s}$	field of velocity vectors that are intrinsic to Γ

List of Acronyms

Acronym	Expansion	Page
`DUNE`	the Distributed and Unified Numerics Environment (`https://www.dune-project.org`)	215
ALE	arbitrary Lagrangian–Eulerian	
ALE ESFEM	ESFEM based on an ALE point of view	16
ALE FEM	FEM based on an ALE point of view	15
BiCGSTAB	biconjugate gradient stabilized (van der Vorst, 1992)	236
CG	conjugate gradient (Hestenes and Stiefel, 1952)	236
DG	discontinuous Galerkin	15
DOF	degree of freedom	130
ESFEM	evolving SFEM (Dziuk and Elliott, 2007a)	16
Eulerian ESFEM	ESFEM based on an Eulerian point of view (Dziuk and Elliott, 2010)	22
Eulerian SDG	SDG based on an Eulerian point of view	113
Eulerian SFEM	SFEM based on an Eulerian point of view (Dziuk and Elliott, 2008)	22
FD	finite difference	14
FEM	finite element method	14
FV	finite volume	15
GMRES	generalized minimal residual (Saad and Schultz, 1986)	236
GOR model	Goryachev model (Goryachev and Pokhilko, 2008)	26
IIPG	incomplete IPG (see e.g. Dawson et al., 2004)	73
IPG	interior penalty Galerkin	
IRAM	implicitly restarted Arnoldi method (Lehoucq and Sorensen, 1996; Lehoucq et al., 1998b)	244

Acronym	**Expansion**	**Page**
IRLM	implicitly restarted Lanczos method (Calvetti et al., 1994)	244
NIPG	non-symmetric IPG (Rivière et al., 1999, 2001)	73
OBB-DG	Oden–Babuška–Baumann DG (Oden et al., 1998)	73
PDE	partial differential equation	5
SDG	surface DG	16
SFEM	surface FEM (Dziuk, 1988)	16
SIPG	symmetric IPG (Wheeler, 1978) (see also Douglas and Dupont, 1976)	73
surfactant	surface active agent	1
SWIPG	symmetric weighted IPG (Ern et al., 2009)	75
TLIME	two-level iterative method for eigenvalue calculations (Szyld, 1988)	238
UDG	unfitted DG (Bastian and Engwer, 2009; Engwer, 2009)	19
WP model	wave-pinning model (Mori et al., 2008), (Giese, Eigel, Westerheide, Engwer and Klipp, 2015a)	26
XFEM	extended FEM	20

Bibliography

H. Abels, K. F. Lam, B. Stinner. *Analysis of the Diffuse Domain Approach for a Bulk-Surface Coupled PDE System.* SIAM Journal on Mathematical Analysis, 47 (2015) (5), pp. 3687–3725.

D. Adalsteinsson, J. Sethian. *Transport and diffusion of material quantities on propagating interfaces via level set methods.* Journal of Computational Physics, 185 (2003) (1), pp. 271–288.

S. Aland, J. Lowengrub, A. Voigt. *Two-phase flow in complex geometries: A diffuse domain approach.* Comput. Model. Eng. Sci., 57 (2010) (1), pp. 77–108.

P. F. Antonietti, A. Dedner, P. Madhavan, S. Stangalino, B. Stinner, M. Verani. *High Order Discontinuous Galerkin Methods for Elliptic Problems on Surfaces.* SIAM Journal on Numerical Analysis, 53 (2015) (2), pp. 1145–1171.

D. N. Arnold, F. Brezzi, B. Cockburn, L. D. Marini. *Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems.* SIAM Journal on Numerical Analysis, 39 (2002) (5), pp. 1749–1779.

R. Barreira, C. M. Elliott, A. Madzvamuse. *The surface finite element method for pattern formation on evolving biological surfaces.* Journal of Mathematical Biology, 63 (2011) (6), pp. 1095–1119.

J. W. Barrett, C. M. Elliott. *Fitted and unfitted finite-element methods for elliptic equations with smooth interfaces.* IMA J. Numer. Anal., 7 (1987) (3), pp. 283–300.

P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, O. Sander. *A generic grid interface for parallel and adaptive scientific computing. Part II: implementation and tests in DUNE.* Computing, 82 (2008a) (2), pp. 121–138.

P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, O. Sander. *A generic grid interface for parallel and adaptive scientific computing. Part I: abstract framework.* Computing, 82 (2008b) (2), pp. 103–119.

P. Bastian, C. Engwer. *An unfitted finite element method using discontinuous Galerkin.* International Journal for Numerical Methods in Engineering, 79 (2009) (12), pp. 1557–1576.

P. Bastian, C. Engwer, J. Fahlke, O. Ippisch. *An Unfitted Discontinuous Galerkin method for pore-scale simulations of solute transport.* Mathematics and Computers in Simulation, 81 (2011) (10), pp. 2051–2061. MAMERN 2009: 3rd International Conference on Approximation Methods and Numerical Modeling in Environment and Natural Resources.

P. Bastian, F. Heimann, S. Marnach. *Generic implementation of finite element methods in the Distributed and Unified Numerics Environment (DUNE).* Kybernetika, 46 (2010) (2), pp. 294–315.

T. Belytschko, W. K. Liu, B. Moran. *Nonlinear Finite Elements for Continua and Structures.* John Wiley and Sons, Ltd., New York, 2000.

T. Belytschko, N. Moës, S. Usui, C. Parimi. *Arbitrary discontinuities in finite elements.* International Journal for Numerical Methods in Engineering, 50 (2001) (4), pp. 993–1013.

M. Bertalmío, L.-T. Cheng, S. Osher, G. Sapiro. *Variational Problems and Partial Differential Equations on Implicit Surfaces.* Journal of Computational Physics, 174 (2001) (2), pp. 759–780.

M. Bertalmío, F. Mémoli, L.-T. Cheng, G. Sapiro, S. Osher. *Variational Problems and Partial Differential Equations on Implicit Surfaces: Bye Bye Triangulated Surfaces?* In S. Osher, N. Paragios, eds., *Geometric Level Set Methods in Imaging, Vision, and Graphics*, chap. 20, pp. 381–397. Springer, New York, NY, 2003.

M. Blatt, P. Bastian. *The Iterative Solver Template Library.* In B. Kågström, E. Elmroth, J. Dongarra, J. Waśniewski, eds., *Applied Parallel Computing. State of the Art in Scientific Computing.* Springer Berlin Heidelberg. ISBN 978-3-540-75755-9, 2007 pp. 666–675.

M. Blatt, A. Burchardt, A. Dedner, C. Engwer, J. Fahlke, B. Flemisch, C. Gersbacher, C. Gräser, F. Gruber, C. Grüninger, D. Kempf, R. Klöfkorn, T. Malkmus, S. Müthing, M. Nolte, M. Piatkowski, O. Sander. *The Distributed and Unified Numerics Environment, Version 2.4.* Archive of Numerical Software, 4 (2016) (100), pp. 13–29.

M. Booty, M. Siegel. *A hybrid numerical method for interfacial fluid flow with soluble surfactant.* Journal of Computational Physics, 229 (2010) (10), pp. 3864–3883.

F. Brezzi, L. D. Marini, E. Süli. *Discontinuous Galerkin methods for first-order hyperbolic problems.* Mathematical Models and Methods in Applied Sciences, 14 (2004) (12), pp. 1893–1903.

M. Burger. *Finite element approximation of elliptic partial differential equations on implicit surfaces.* Computing and Visualization in Science, 12 (2009) (3), pp. 87–100.

M. Burger, O. L. Elvetun, M. Schlottbom. *Analysis of the Diffuse Domain Method for Second Order Elliptic Boundary Value Problems.* Foundations of Computational Mathematics, 17 (2017) (3), pp. 627–674.

E. Burman, S. Claus, P. Hansbo, M. G. Larson, A. Massing. *CutFEM: Discretizing geometry and partial differential equations.* International Journal for Numerical Methods in Engineering, 104 (2015) (7), pp. 472–501.

E. Burman, P. Hansbo, M. G. Larson, A. Massing. *Cut Finite Element Methods for Partial Differential Equations on Embedded Manifolds of Arbitrary Codimensions.* ArXiv e-prints, (2016a). `arXiv:1610.01660`.

E. Burman, P. Hansbo, M. G. Larson, S. Zahedi. *Cut finite element methods for coupled bulk–surface problems.* Numerische Mathematik, 133 (2016b) (2), pp. 203–231.

F. Burstall. *Basic Riemannian Geometry.* In B. Davies, Y. Safarov, eds., *Spectral Theory and Geometry*, vol. 273 of *London Mathematical Society Lecture Note Series*, chap. 1, pp. 1–29. Cambridge University Press, 1999.

B. L. Buzbee, F. W. Dorr, J. A. George, G. H. Golub. *The Direct Solution of the Discrete Poisson Equation on Irregular Regions.* SIAM Journal on Numerical Analysis, 8 (1971) (4), pp. 722–736.

D. Calvetti, L. Reichel, D. C. Sorensen. *An implicitly restarted Lanczos method for large symmetric eigenvalue problems.* Electronic Transactions on Numerical Analysis, 2 (1994), pp. 1–21.

M. C. Calzada, G. Camacho, E. Fernández-Cara, M. Marín. *Fictitious domains and level sets for moving boundary problems. Applications to the numerical simulation of tumor growth.* Journal of Computational Physics, 230 (2011) (4), pp. 1335–1358.

P. Cermelli, E. Fried, M. E. Gurtin. *Transport relations for surface integrals arising in the formulation of balance laws for evolving fluid interfaces.* Journal of Fluid Mechanics, 544 (2005), pp. 339–351.

T. F. Chan, L. A. Vese. *An Active Contour Model without Edges.* In M. Nielsen, P. Johansen, O. F. Olsen, J. Weickert, eds., *Scale-Space Theories in Computer Vision.* Springer Berlin Heidelberg. ISBN 978-3-540-48236-9, 1999 pp. 141–151.

T. F. Chan, L. A. Vese. *Active contours without edges.* IEEE Transactions on Image Processing, 10 (2001) (2), pp. 266–277.

M. Chaplain, M. Ganesh, I. Graham. *Spatio-temporal pattern formation on spherical surfaces: numerical simulation and application to solid tumour growth.* Journal of Mathematical Biology, 42 (2001) (5), pp. 387–423.

A. V. Chechkin, I. M. Zaid, M. A. Lomholt, I. M. Sokolov, R. Metzler. *Bulk-mediated diffusion on a planar surface: Full solution.* Phys. Rev. E, 86 (2012), p. 041101.

L.-T. Cheng, P. Burchard, B. Merriman, S. Osher. *Motion of Curves Constrained on Surfaces Using a Level-Set Approach.* Journal of Computational Physics, 175 (2002) (2), pp. 604–644.

A. Chorin, J. Marsden. *A Mathematical Introduction to Fluid Mechanics*, vol. 4 of *Texts in Applied Mathematics*. Springer New York, third edn., 2000.

T. A. Davis. *Algorithm 832: UMFPACK V4.3—an Unsymmetric-pattern Multifrontal Method.* ACM Trans. Math. Softw., 30 (2004) (2), pp. 196–199.

C. Dawson, S. Sun, M. F. Wheeler. *Compatible algorithms for coupled flow and transport.* Computer Methods in Applied Mechanics and Engineering, 193 (2004) (23–26), pp. 2565–2580.

K. Deckelnick, G. Dziuk, C. M. Elliott, C.-J. Heine. *An h-narrow band finite-element method for elliptic equations on implicit surfaces.* IMA Journal of Numerical Analysis, 30 (2010) (2), pp. 351–376.

K. Deckelnick, C. M. Elliott, T. Ranner. *Unfitted Finite Element Methods Using Bulk Meshes for Surface Partial Differential Equations.* SIAM Journal on Numerical Analysis, 52 (2014) (4), pp. 2137–2162.

A. Dedner, P. Madhavan. *Discontinuous Galerkin methods for hyperbolic and advection-dominated problems on surfaces.* ArXiv e-prints, (2015). `arXiv:1505.06752`.

A. Dedner, P. Madhavan. *Adaptive discontinuous Galerkin methods on surfaces.* Numerische Mathematik, 132 (2016) (2), pp. 369–398.

A. Dedner, P. Madhavan, B. Stinner. *Analysis of the discontinuous Galerkin method for elliptic problems on surfaces.* IMA Journal of Numerical Analysis, 33 (2013) (3), pp. 952–973.

A. Demlow. *Higher-Order Finite Element Methods and Pointwise Error Estimates for Elliptic Problems on Surfaces.* SIAM Journal on Numerical Analysis, 47 (2009) (2), pp. 805–827.

J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, J. W. H. Liu. *A supernodal approach to sparse partial pivoting.* SIAM J. Matrix Analysis and Applications, 20 (1999) (3), pp. 720–755.

E. W. Dent, A. V. Kwiatkowski, L. M. Mebane, U. Philippar, M. Barzik, D. A. Rubinson, S. Gupton, J. E. V. Veen, C. Furman, J. Zhang, A. S. Alberts, S. Mori, F. B. Gertler. *Filopodia are required for cortical neurite initiation.* Nature Cell Biology, 9 (2007) (12), pp. 1347–1359.

D. Di Pietro, A. Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*, vol. 69 of *Mathématiques et Applications*. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-22979-4.

U. Diewald, T. Preusser, M. Rumpf. *Anisotropic diffusion in vector field visualization on Euclidean domains and surfaces.* IEEE Transactions on Visualization and Computer Graphics, 6 (2000) (2), pp. 139–149.

J. E. Dolbow. *An Extended Finite Element Method with Discontinuous Enrichment for Applied Mechanics.* Ph.D. thesis, Northwestern University, 1999.

J. Donea, A. Huerta, J.-P. Ponthot, A. Rodríguez-Ferran. *Arbitrary Lagrangian–Eulerian Methods.* In *Encyclopedia of Computational Mechanics*, vol. 1, chap. 14, pp. 413–437. John Wiley & Sons, Ltd. ISBN 9780470091357, 2004.

J. Dorsey, P. Hanrahan. *Digital Materials and Virtual Weathering.* Scientific American, 282 (2000), pp. 64–71.

J. Douglas, T. Dupont. *Interior Penalty Procedures for Elliptic and Parabolic Galerkin Methods.* In R. Glowinski, J. L. Lions, eds., *Computing Methods in Applied Sciences: Second International Symposium December 15–19, 1975*, vol. 58 of *Lecture Notes in Physics*, pp. 207–216. Springer Berlin Heidelberg. ISBN 978-3-540-37550-0, 1976.

G. Dziuk. *Finite Elements for the Beltrami operator on arbitrary surfaces.* In *Partial Differential Equations and Calculus of Variations*, vol. 1357 of *Lecture Notes in Mathematics*, pp. 142–155. Springer. ISBN 978-3-540-50508-2, 1988.

G. Dziuk, C. M. Elliott. *Finite elements on evolving surfaces.* IMA Journal of Numerical Analysis, 27 (2007a) (2), pp. 262–292.

G. Dziuk, C. M. Elliott. *Surface Finite Elements for Parabolic Equations.* Journal of Computational Mathematics, 25 (2007b) (4), pp. 385–407.

G. Dziuk, C. M. Elliott. *Eulerian finite element method for parabolic PDEs on implicit surfaces.* Interfaces and Free Boundaries, 10 (2008), pp. 119–138.

G. Dziuk, C. M. Elliott. *An Eulerian approach to transport and diffusion on evolving implicit surfaces.* Computing and Visualization in Science, 13 (2010), pp. 17–28.

G. Dziuk, C. M. Elliott. *Finite element methods for surface PDEs.* Acta Numerica, 22 (2013), pp. 289–396.

K. Ecker. *Regularity Theory for Mean Curvature Flow*, vol. 57 of *Progress in Nonlinear Differential Equations and Their Applications.* Birkhäuser Boston, 2004.

C. M. Elliott, T. Ranner. *Finite element analysis for a coupled bulk–surface partial differential equation.* IMA Journal of Numerical Analysis, 33 (2013) (2), pp. 377–402.

C. M. Elliott, B. Stinner. *Analysis of a diffuse interface approach to an advection diffusion equation on a moving surface.* Mathematical Models and Methods in Applied Sciences, 19 (2009) (05), pp. 787–802.

C. M. Elliott, B. Stinner, V. Styles, R. Welford. *Numerical computation of advection and diffusion on evolving diffuse interfaces.* IMA Journal of Numerical Analysis, 31 (2011) (3), pp. 786–812.

C. M. Elliott, B. Stinner, C. Venkataraman. *Modelling cell motility and chemotaxis with evolving surface finite elements.* Journal of The Royal Society Interface, 9 (2012) (76), pp. 3027–3044.

C. M. Elliott, V. Styles. *An ALE ESFEM for Solving PDEs on Evolving Surfaces.* Milan Journal of Mathematics, 80 (2012) (2), pp. 469–501.

C. M. Elliott, C. Venkataraman. *Error analysis for an ALE evolving surface finite element method.* Numerical Methods for Partial Differential Equations, 31 (2015) (2), pp. 459–499.

N. Emken. *A coupled bulk–surface reaction–diffusion–advection model for cell polarization.* Ph.D. thesis, University of Münster, 2016.

C. Engwer. *An Unfitted Discontinuous Galerkin Scheme for Micro-scale Simulations and Numerical Upscaling.* Ph.D. thesis, University of Heidelberg, 2009.

C. Engwer, F. Heimann. *Dune-UDG: A Cut-Cell Framework for Unfitted Discontinuous Galerkin Methods.* In *Advances in DUNE.* Springer, 2012 pp. 89–100.

C. Engwer, A. Nüssing. *Geometric Reconstruction of Implicitly Defined Surfaces and Domains with Topological Guarantees.* ACM Trans. Math. Softw., 44 (2017) (2), pp. 14:1–14:20.

C. Engwer, T. Ranner, S. Westerheide. *An unfitted discontinuous Galerkin scheme for conservation laws on evolving surfaces.* In A. Handlovičová, D. Ševčovič, eds., *Proceedings of ALGORITMY 2016, 20th Conference on Scientific Computing, Vysoké Tatry, Podbanské, Slovakia, March 13–18, 2016.* Publishing House of Slovak University of Technology in Bratislava. ISBN 978-80-227-4454-4, 2016 pp. 44–54. Also: arXiv e-prints (arXiv:1602.01080) (02/2016).

C. Engwer, S. Westerheide. *Heterogeneous Coupling for Implicitly Described Domains.* In J. Erhel, M. J. Gander, L. Halpern, G. Pichot, T. Sassi, O. Widlund, eds., *Domain Decomposition Methods in Science and Engineering XXI*, vol. 98 of *Lecture Notes in Computational Science and Engineering*. Springer International Publishing, Cham. ISBN 978-3-319-05789-7, 2014 pp. 809–817.

C. Engwer, S. Westerheide. *Unfitted DG schemes for coupled bulk–surface PDEs on complex geometries*, 2018. In preparation.

A. Ern, A. F. Stephansen, P. Zunino. *A discontinuous Galerkin method with weighted averages for advection–diffusion equations with locally small and anisotropic diffusivity.* IMA Journal of Numerical Analysis, 29 (2009) (2), pp. 235–256.

H. Federer. *Curvature Measures.* Transactions of the American Mathematical Society, 93 (1959) (3), pp. 418–491.

F. H. Fenton, E. M. Cherry, A. Karma, W.-J. Rappel. *Modeling wave propagation in realistic heart geometries using the phase-field method.* Chaos: An Interdisciplinary Journal of Nonlinear Science, 15 (2005) (1), p. 013502.

A. Friedman, W. Littman. *Industrial Mathematics: A Course in Solving Real-World Problems.* Society for Industrial and Applied Mathematics, Philadelphia, 1994.

S. Ganesan, A. Hahn, K. Held, L. Tobiska. *An accurate numerical method for computation of two-phase flows with surfactants.* In *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2012).* TU Vienna, Austria, September 10–14, 2012 .

S. Ganesan, L. Tobiska. *Arbitrary Lagrangian–Eulerian finite-element method for computation of two-phase flows with soluble surfactants.* Journal of Computational Physics, 231 (2012) (9), pp. 3685–3702.

H. Garcke, J. Kampmann, A. Rätz, M. Röger. *A coupled surface-Cahn–Hilliard bulk-diffusion system modeling lipid raft formation in cell membranes.* ArXiv e-prints, (2015). `arXiv:1509.03655`.

H. Garcke, K. F. Lam, B. Stinner. *Diffuse interface modelling of soluble surfactants in two-phase flow.* Communications in Mathematical Sciences, 12 (2014) (8), pp. 1475–1522.

E. H. Georgoulis, E. Hall, P. Houston. *Discontinuous Galerkin Methods for Advection-Diffusion-Reaction Problems on Anisotropically Refined Meshes.* SIAM Journal on Scientific Computing, 30 (2007) (1), pp. 246–271.

W. Giese, M. Eigel, S. Westerheide, C. Engwer, E. Klipp. *Influence of cell shape, inhomogeneities and diffusion barriers in cell polarization models.* Physical Biology, 12 (2015a) (6), p. 066014.

W. Giese, M. Eigel, S. Westerheide, C. Engwer, E. Klipp. *Supporting Text S1.* In Giese et al. (2015a), pp. 1–18 of supplementary data.

J. Giesselmann, T. Müller. *Geometric error of finite volume schemes for conservation laws on evolving surfaces.* Numerische Mathematik, 128 (2014) (3), pp. 489–516.

R. Glowinski, T.-W. Pan, J. Periaux. *A fictitious domain method for Dirichlet problem and applications.* Computer Methods in Applied Mechanics and Engineering, 111 (1994) (3), pp. 283–303.

A. Goryachev, A. Pokhilko. *Dynamics of Cdc42 network embodies a Turing-type mechanism of yeast cell polarity.* FEBS Letters, 582 (2008) (10), pp. 1437–1443.

J. Grande, C. Lehrenfeld, A. Reusken. *Analysis of a high order Trace Finite Element Method for PDEs on level set surfaces.* ArXiv e-prints, (2016). Accepted for publication in SIAM Num. Ana. (2017), `arXiv:1611.01100`.

J. B. Greer. *An Improvement of a Recent Eulerian Method for Solving PDEs on General Geometries.* Journal of Scientific Computing, 29 (2006) (3), pp. 321–352.

J. B. Greer, A. L. Bertozzi, G. Sapiro. *Fourth order partial differential equations on general geometries.* Journal of Computational Physics, 216 (2006) (1), pp. 216–246.

W. Hackbusch, S. A. Sauter. *Composite Finite Elements for problems containing small geometric details. Part II: Implementation and numerical results.* Computing and Visualization in Science, 1 (1997a) (1), pp. 15–25.

W. Hackbusch, S. A. Sauter. *Composite Finite Elements for the approximation of PDEs on domains with complicated micro-structures.* Numerische Mathematik, 75 (1997b) (4), pp. 447–472.

A. Hahn, K. Held, L. Tobiska. *ALE-FEM for Two-Phase Flows.* Proc. Appl. Math. Mech., 13 (2013) (1), pp. 319–320.

A. Hahn, K. Held, L. Tobiska. *Modelling of Surfactant Concentration in a Coupled Bulk Surface Problem.* Proc. Appl. Math. Mech., 14 (2014) (1), pp. 525–526.

E. Hairer, C. Lubich, G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, vol. 31 of *Springer Series in Computational Mathematics.* Springer-Verlag Berlin Heidelberg, second edn., 2006. ISBN 978-3-540-30663-4.

D. Halpern, O. E. Jensen, J. B. Grotberg. *A theoretical study of surfactant and liquid delivery into the lung.* Journal of Applied Physiology, 85 (1998) (1), pp. 333–352.

A. Hansbo, P. Hansbo. *An unfitted finite element method, based on Nitsche's method, for elliptic interface problems.* Computer Methods in Applied Mechanics and Engineering, 191 (2002) (47-48), pp. 5537–5552.

P. Hansbo, M. G. Larson, S. Zahedi. *Characteristic cut finite element methods for convection–diffusion problems on time dependent surfaces.* Computer Methods in Applied Mechanics and Engineering, 293 (2015), pp. 431–461.

F. Heimann. *An Unfitted Higher-Order Discontinuous Galerkin Method for Incompressible Two-Phase Flow with Moving Contact Lines.* Ph.D. thesis, University of Heidelberg, 2013.

F. Heimann, C. Engwer, O. Ippisch, P. Bastian. *An unfitted interior penalty discontinuous Galerkin method for incompressible Navier–Stokes two-phase flow.* International Journal for Numerical Methods in Fluids, 71 (2013) (3), pp. 269–293.

M. R. Hestenes, E. Stiefel. *Methods of Conjugate Gradients for Solving Linear Systems.* Journal of Research of the National Bureau of Standards, 49 (1952) (6), pp. 409–436.

W. Hundsdorfer, J. G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, vol. 33. Springer Berlin Heidelberg, 2003.

T. Jahnke, C. Lubich. *Error Bounds for Exponential Operator Splittings.* BIT Numerical Mathematics, 40 (2000) (4), pp. 735–744.

A. J. James, J. Lowengrub. *A surfactant-conserving volume-of-fluid method for interfacial flows with insoluble surfactant.* Journal of Computational Physics, 201 (2004) (2), pp. 685–722.

A. Johansson, M. G. Larson. *A high order discontinuous Galerkin Nitsche method for elliptic problems with fictitious boundary.* Numerische Mathematik, 123 (2013) (4), pp. 607–628.

L. Ju, Q. Du. *A finite volume method on general surfaces and its error estimates.* Journal of Mathematical Analysis and Applications, 352 (2009) (2), pp. 645–668.

J. Kockelkoren, H. Levine, W.-J. Rappel. *Computational approach for modeling intra- and extracellular dynamics.* Phys. Rev. E, 68 (2003), p. 037702.

F. Kummer. *Extended discontinuous Galerkin methods for two-phase flows: the spatial discretization.* International Journal for Numerical Methods in Engineering, 109 (2017) (2), pp. 259–289.

F. Kummer, B. Müller, T. Utz. *Time integration for extended discontinuous Galerkin methods with moving domains.* International Journal for Numerical Methods in Engineering, 113 (2018) (5), pp. 767–788.

D. Kuzmin, J. Hamalainen. *Finite Element Methods for Computational Fluid Dynamics: A Practical Guide.* SIAM, 2014.

Y.-I. Kwon, J. J. Derby. *Modeling the coupled effects of interfacial and bulk phenomena during solution crystal growth.* Journal of Crystal Growth, 230 (2001) (1–2), pp. 328–335. Proceedings of the Third International Workshop om Modeling in Crystal Growth.

J. M. Lee. *Manifolds and Differential Geometry*, vol. 107 of *Graduate studies in mathematics.* American Mathematical Society, 2009. ISBN 9780821848159.

J. M. Lee. *Introduction to Smooth Manifolds*, vol. 218 of *Graduate Texts in Mathematics.* Springer New York, 2012. ISBN 9781441999825.

L. Lee, R. J. LeVeque. *An Immersed Interface Method for Incompressible Navier–Stokes Equations.* SIAM Journal on Scientific Computing, 25 (2003) (3), pp. 832–856.

R. Lehoucq, D. Sorensen, C. Yang. *ARPACK Users' Guide.* Society for Industrial and Applied Mathematics, 1998a.

R. Lehoucq, D. Sorensen, C. Yang. *The Implicitly Restarted Arnoldi Method.* In Lehoucq et al. (1998a), chap. 4, pp. 43–66, 1998b.

R. B. Lehoucq, D. C. Sorensen. *Deflation Techniques for an Implicitly Restarted Arnoldi Iteration.* SIAM Journal on Matrix Analysis and Applications, 17 (1996) (4), pp. 789–821.

C. Lehrenfeld. *High order unfitted finite element methods on level set domains using isoparametric mappings.* Computer Methods in Applied Mechanics and Engineering, 300 (2016), pp. 716–733.

M. Lenz, S. F. Nemadjieu, M. Rumpf. *A Convergent Finite Volume Scheme for Diffusion on Evolving Surfaces.* SIAM Journal on Numerical Analysis, 49 (2011) (1), pp. 15–37.

K. Y. Lervåg, J. Lowengrub. *Analysis of the diffuse-domain method for solving PDEs in complex geometries.* Communications in Mathematical Sciences, 13 (2015) (6), pp. 1473–1500.

R. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge Texts in Applied Mathematics, Cambridge University Press, 2002. ISBN 9780521009249.

R. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems.* Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, 2007. ISBN 9780898716290.

R. J. LeVeque, Z. Li. *The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources.* SIAM Journal on Numerical Analysis, 31 (1994) (4), pp. 1019–1044.

H. Levine, W.-J. Rappel. *Membrane-bound Turing patterns.* Phys. Rev. E, 72 (2005), p. 061912.

T. Lewiner, H. Lopes, A. W. Vieira, G. Tavares. *Efficient implementation of Marching Cubes' cases with topological guarantees.* Journal of Graphics Tools, 8 (2003) (2), pp. 1–15.

X. Li, J. Lowengrub, A. Rätz, A. Voigt. *Solving PDEs in complex geometries: a diffuse domain approach.* Communications in Mathematical Sciences, 7 (2009) (1), pp. 81–107.

F. Liehr, T. Preusser, M. Rumpf, S. Sauter, L. O. Schwen. *Composite Finite Elements for 3D Image Based Computing.* Computing and Visualization in Science, 12 (2009) (4), pp. 171–188.

W. E. Lorensen, H. E. Cline. *Marching cubes: A high resolution 3D surface construction algorithm.* ACM SIGGRAPH Computer Graphics, 21 (1987) (4), pp. 163–169.

J. S. Lowengrub, A. Rätz, A. Voigt. *Phase-field modeling of the dynamics of multicomponent vesicles: Spinodal decomposition, coarsening, budding, and fission.* Phys. Rev. E. Stat. Nonlin. Soft Matter Phys., 79 (2009), p. 031926.

C. B. Macdonald, S. J. Ruuth. *The Implicit Closest Point Method for the Numerical Solution of Partial Differential Equations on Surfaces*. SIAM Journal on Scientific Computing, 31 (2009) (6), pp. 4330–4350.

G. MacDonald, J. Mackenzie, M. Nolan, R. Insall. *A computational method for the coupled solution of reaction–diffusion equations on evolving domains and manifolds: Application to a model of cell migration and chemotaxis*. Journal of Computational Physics, 309 (2016), pp. 207–226.

Y. Maday, A. T. Patera, E. M. Rønquist. *An Operator-Integration-Factor Splitting Method for Time-Dependent Problems: Application to Incompressible Fluid Flow*. Journal of Scientific Computing, 5 (1990) (4), pp. 263–292.

P. Madhavan. *Analysis of discontinuous Galerkin methods on surfaces*. Ph.D. thesis, University of Warwick, 2014.

A. Madzvamuse, A. H. Chung. *Analysis and Simulations of Coupled Bulk–surface Reaction–Diffusion Systems on Exponentially Evolving Volumes*. Math. Model. Nat. Phenom., 11 (2016) (5), pp. 4–32.

A. Madzvamuse, A. H. W. Chung, C. Venkataraman. *Stability analysis and simulations of coupled bulk-surface reaction–diffusion systems*. Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 471 (2015) (2175).

W. Marth, A. Voigt. *Signaling networks and cell motility: a computational approach using a phase field description*. Journal of Mathematical Biology, 69 (2014) (1), pp. 91–112.

A. Massing, C. Gürkan. *A stabilized cut discontinuous Galerkin framework: I. Elliptic boundary value and interface problems*. ArXiv e-prints, (2018). `arXiv:1803.06635`.

R. Massjung. *An Unfitted Discontinuous Galerkin Method Applied to Elliptic Interface Problems*. SIAM Journal on Numerical Analysis, 50 (2012) (6), pp. 3134–3162.

E. S. Medvedev, A. A. Stuchebrukhov. *Proton diffusion along biological membranes*. Journal of Physics: Condensed Matter, 23 (2011) (23), p. 234103.

E. S. Medvedev, A. A. Stuchebrukhov. *Mechanism of long-range proton translocation along biological membranes*. FEBS Letters, 587 (2013) (4), pp. 345–349.

F. Mémoli, G. Sapiro, P. Thompson. *Implicit brain imaging*. NeuroImage, 23, Supplement 1 (2004), pp. S179–S188. Mathematics in Brain Imaging.

N. Moës, J. Dolbow, T. Belytschko. *A finite element method for crack growth without remeshing.* International Journal for Numerical Methods in Engineering, 46 (1999) (1), pp. 131–150.

Y. Mori, A. Jilkine, L. Edelstein-Keshet. *Wave-Pinning and Cell Polarity from a Bistable Reaction-Diffusion System.* Biophysical Journal, 94 (2008) (9), pp. 3684–3697.

B. Müller, S. Krämer-Eis, F. Kummer, M. Oberlack. *A high-order discontinuous Galerkin method for compressible flows with immersed boundaries.* International Journal for Numerical Methods in Engineering, 110 (2017) (1), pp. 3–30.

T. Myers, J. Charpin. *A mathematical model for atmospheric ice accretion and water flow on a cold surface.* International Journal of Heat and Mass Transfer, 47 (2004) (25), pp. 5483–5500.

T. G. Myers, J. P. F. Charpin, S. J. Chapman. *The flow and solidification of a thin fluid film on an arbitrary three-dimensional surface.* Physics of Fluids, 14 (2002) (8), pp. 2788–2803.

W. Nagata, H. R. Z. Zangeneh, D. M. Holloway. *Reaction–Diffusion Patterns in Plant Tip Morphogenesis: Bifurcations on Spherical Caps.* Bulletin of Mathematical Biology, 75 (2013) (12), pp. 2346–2371.

M. P. Neilson, J. A. Mackenzie, S. D. Webb, R. H. Insall. *Modeling Cell Movement and Chemotaxis Using Pseudopod-Based Feedback.* SIAM Journal on Scientific Computing, 33 (2011a) (3), pp. 1035–1057.

M. P. Neilson, D. M. Veltman, P. J. M. van Haastert, S. D. Webb, J. A. Mackenzie, R. H. Insall. *Chemotaxis: A Feedback-Based Computational Model Robustly Predicts Multiple Aspects of Real Cell Behaviour.* PLOS Biology, 9 (2011b) (5), pp. 1–11.

S. F. Nemadjieu. *Finite Volume Methods for Advection Diffusion on Moving Interfaces and Application on Surfactant Driven Thin Film Flow.* Ph.D. thesis, University of Bonn, 2012.

O. Nemitz, M. B. Nielsen, M. Rumpf, R. Whitaker. *Finite Element Methods on Very Large, Dynamic Tubular Grid Encoded Implicit Surfaces.* SIAM Journal on Scientific Computing, 31 (2009) (3), pp. 2258–2281.

J. A. Nitsche. *Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind.* Abh. Math. Sem. Univ. Hamburg, 36 (1971), pp. 9–15.

I. L. Novak, F. Gao, Y.-S. Choi, D. Resasco, J. C. Schaff, B. M. Slepchenko. *Diffusion on a curved surface coupled to diffusion in the volume: Application to cell biology.* Journal of Computational Physics, 226 (2007) (2), pp. 1271–1290.

J. T. Oden, I. Babuška, C. E. Baumann. *A Discontinuous hp Finite Element Method for Diffusion Problems.* Journal of Computational Physics, 146 (1998) (2), pp. 491–519.

M. A. Olshanskii, A. Reusken, J. Grande. *A Finite Element Method for Elliptic Equations on Surfaces.* SIAM Journal on Numerical Analysis, 47 (2009) (5), pp. 3339–3358.

K. Orlando, W. Guo. *Membrane Organization and Dynamics in Cell Polarity.* Cold Spring Harbor Perspectives in Biology, 1 (2009) (5).

S. Osher, R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, vol. 153 of *Applied Mathematical Sciences.* Springer, 2003.

S. Osher, J. Sethian. *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations.* J. Comput. Phys., 79 (1988) (1), pp. 12–49.

M. Otsuji, S. Ishihara, C. Co, K. Kaibuchi, A. Mochizuki, S. Kuroda. *A Mass Conserved Reaction–Diffusion System Captures Properties of Cell Polarity.* PLOS Computational Biology, 3 (2007) (6), p. e108.

B. N. Parlett. *The Rayleigh Quotient Iteration and Some Generalizations for Nonnormal Matrices.* Mathematics of Computation, 28 (1974) (127), pp. 679–693.

C. S. Peskin. *Numerical analysis of blood flow in the heart.* Journal of Computational Physics, 25 (1977) (3), pp. 220–252.

C. S. Peskin. *The immersed boundary method.* Acta Numerica, 11 (2002), pp. 479–517.

A. Petras, S. Ruuth. *PDEs on moving surfaces via the closest point method and a modified grid based particle method.* Journal of Computational Physics, 312 (2016), pp. 139–156.

O. Pironneau, J. Liou, T. Tezduyar. *Characteristic-galerkin and galerkin/least-squares space-time formulations for the advection-diffusion equation with time-dependent domains.* Computer Methods in Applied Mechanics and Engineering, 100 (1992) (1), pp. 117–141.

R. Plaza, F. Sánchez-Garduño, P. Padilla, R. Barrio, P. Maini. *The Effect of Growth and Curvature on Pattern Formation.* Journal of Dynamics and Differential Equations, 16 (2004) (4), pp. 1093–1121.

M. M. Rainey, D. Korostyshevsky, S. Lee, E. O. Perlstein. *The Antidepressant Sertraline Targets Intracellular Vesiculogenic Membranes in Yeast.* Genetics, 185 (2010) (4), pp. 1221–1233.

A. Rätz, A. Ribalta, A. Voigt. *Surface evolution of elastically stressed films under deposition by a diffuse interface model.* Journal of Computational Physics, 214 (2006) (1), pp. 187–208.

A. Rätz, M. Röger. *Turing instabilities in a mathematical model for signaling networks.* Journal of Mathematical Biology, 65 (2012) (6), pp. 1215–1244.

A. Rätz, M. Röger. *Symmetry breaking in a bulk–surface reaction–diffusion model for signalling networks.* Nonlinearity, 27 (2014) (8), p. 1805.

A. Rätz, A. Voigt. *PDE's on surfaces—a diffuse interface approach.* Commun. Math. Sci., 4 (2006) (3), pp. 575–590.

M. Rech, S. Sauter, A. Smolianski. *Two-scale composite finite element method for Dirichlet problems on complicated domains.* Numerische Mathematik, 102 (2006) (4), pp. 681–708.

M. G. Reuter, J. C. Hill, R. J. Harrison. *Solving PDEs in irregular geometries with multiresolution methods I: Embedded Dirichlet boundary conditions.* Computer Physics Communications, 183 (2012) (1), pp. 1–7.

B. Rivière, P. Bastian. *Discontinuous Galerkin Methods for Two-phase Flow in Porous Media.* Tech. Rep. 2004–28, IWR (SFB 359), University of Heidelberg, 2004.

B. Rivière, M. F. Wheeler, V. Girault. *Improved energy estimates for interior penalty, constrained and discontinuous Galerkin methods for elliptic problems. Part I.* Computational Geosciences, 3 (1999) (3), pp. 337–360.

B. Rivière, M. F. Wheeler, V. Girault. *A Priori Error Estimates for Finite Element Methods Based on Discontinuous Approximation Spaces for Elliptic Problems.* SIAM Journal on Numerical Analysis, 39 (2001) (3), pp. 902–931.

B. Rivière. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation.* Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. ISBN 978-0-89871-656-6.

R. V. Roy, A. J. Roberts, M. E. Simpson. *A lubrication model of coating flows over a curved substrate in space.* Journal of Fluid Mechanics, 454 (2002), pp. 235–261.

I. Rozada, S. J. Ruuth, M. J. Ward. *The Stability of Localized Spot Patterns for the Brusselator on the Sphere.* SIAM Journal on Applied Dynamical Systems, 13 (2014) (1), pp. 564–627.

B. Rubinstein, B. D. Slaughter, R. Li. *Weakly nonlinear analysis of symmetry breaking in cell polarity models.* Physical Biology, 9 (2012) (4), p. 045006.

S. J. Ruuth, B. Merriman. *A simple embedding method for solving partial differential equations on surfaces.* Journal of Computational Physics, 227 (2008) (3), pp. 1943–1961.

Y. Saad, M. H. Schultz. *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems.* SIAM Journal on Scientific and Statistical Computing, 7 (1986) (3), pp. 856–869.

K. Salari, P. Knupp. *Code Verification by the Method of Manufactured Solutions.* Tech. Rep. SAND2000-1444, Sandia National Laboratories, Albuquerque, NM (US) and Livermore, CA (US), 2000.

S. A. Sauter. *Composite Finite Elements for problems with complicated boundary. Part III: Essential Boundary Conditions.* Tech. Rep. 97-16, Mathematisches Seminar Kiel, University of Kiel, 1997.

W. Schiesser. *The numerical method of lines.* Academic Press Inc., 1991. ISBN 0-12-624130-9. Integration of partial differential equations.

J. Sethian. *Level Set Methods and Fast Marching Methods*, vol. 3 of *Cambridge Monographs on Applied and Computational Mathematics.* Cambridge University Press, second edn., 1999.

J. Stoer, R. Bulirsch. *Introduction to Numerical Analysis*, vol. 12 of *Texts in Applied Mathematics.* Springer New York, third edn., 2002. ISBN 978-0-387-95452-3.

G. Strang. *On the Construction and Comparison of Difference Schemes.* SIAM Journal on Numerical Analysis, 5 (1968) (3), pp. 506–517.

M. Sussman, P. Smereka, S. Osher. *A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow.* Journal of Computational Physics, 114 (1994) (1), pp. 146–159.

D. B. Szyld. *Criteria for Combining Inverse and Rayleigh Quotient Iteration.* SIAM Journal on Numerical Analysis, 25 (1988) (6), pp. 1369–1375.

P. Tang, F. Qiu, H. Zhang, Y. Yang. *Phase separation patterns for diblock copolymers on spherical surfaces: A finite volume method.* Phys. Rev. E, 72 (2005), p. 016710.

A. B. Tayler. *Mathematical Models in Applied Mechanics*, vol. 1 of *Oxford applied mathematics and computing science series.* Oxford University Press, New York, NY, USA, 1986.

K. Teigen, X. Li, J. Lowengrub, F. Wang, A. Voigt. *A diffuse-interface approach for modelling transport, diffusion and adsorption/desorption of material quantities on a deformable interface.* Communications in Mathematical Sciences, 7 (2009) (4), pp. 1009–1037.

K. E. Teigen, P. Song, J. Lowengrub, A. Voigt. *A diffuse-interface method for two-phase flows with soluble surfactants.* Journal of Computational Physics, 230 (2011) (2), pp. 375–393.

A. Toga. *Brain Warping.* Academic Press, New York, 1998.

S. Torabi, J. Lowengrub, A. Voigt, S. Wise. *A New Phase-Field Model for Strongly Anisotropic Systems.* Proceedings: Mathematical, Physical and Engineering Sciences, 465 (2009) (2105), pp. 1337–1359.

L. N. Trefethen, D. Bau. *Numerical Linear Algebra.* SIAM, 1997. ISBN 0-89871-361-7.

H. F. Trotter. *On the product of semi-groups of operators.* Proceedings of the American Mathematical Society, 10 (1959) (4), pp. 545–551.

G. Turk. *Generating Textures on Arbitrary Surfaces Using Reaction-diffusion.* SIGGRAPH Comput. Graph., 25 (1991) (4), pp. 289–298.

C. Varea, J. L. Aragón, R. A. Barrio. *Turing patterns on a sphere.* Phys. Rev. E, 60 (1999), pp. 4588–4592.

C. Venkataraman, T. Sekimura, E. A. Gaffney, P. K. Maini, A. Madzvamuse. *Modeling parr-mark pattern formation during the early development of Amago trout.* Phys. Rev. E, 84 (2011), p. 041923.

H. A. van der Vorst. *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems.* SIAM Journal on Scientific and Statistical Computing, 13 (1992) (2), pp. 631–644.

S. Westerheide. *Bildbasierte Lösung von Partiellen Differentialgleichungen mit Composite Finite Elements.* Diploma thesis, Institute for Computational and Applied Mathematics, University of Münster, 2011.

M. Wheeler. *An elliptic collocation-finite element method with interior penalties.* SIAM J. Numer. Anal., 15 (1978) (1), pp. 152–161.

A. Witkin, M. Kass. *Reaction-diffusion Textures.* SIGGRAPH Comput. Graph., 25 (1991) (4), pp. 299–308.

J.-J. Xu, H.-K. Zhao. *An Eulerian Formulation for Solving Partial Differential Equations Along a Moving Interface.* Journal of Scientific Computing, 19 (2003) (1), pp. 573–594.

O. Zienkiewicz, R. Taylor, J. Zhu. *The Finite Element Method: Its Basis and Fundamentals.* The Finite Element Method, Elsevier Science, 2013. ISBN 9780080951355.