An automated hierarchical eXtended finite element approach for multiphysics problems involving discontinuities

DISSERTATION

zur Erlangung des Grades
Doktor der Naturwissenschaften
- Dr. rer. nat. -

vorgelegt im
Fachbereich 3 (Mathematik und Informatik)
der Universität Bremen

von
Mischa JAHN

Supervisor:
Prof. Dr. Alfred SCHMIDT (University of Bremen, Germany)
Dr.-Ing. Jonathan MONTALVO-URQUIZO (MOCTech, Monterrey, Mexico)

Tag der mündlichen Prüfung: 18.09.2018

Abstract

An automated hierarchical eXtended finite element approach for multiphysics problems involving discontinuities

by Mischa JAHN

In this thesis, a hierarchical eXtended finite element method for the modeling and numerical simulation of multiphysics problems and its implementation into a framework that uses automated code generation is presented. The approach consists of introducing hierarchically ordered level set functions, motivated by the structure of the considered problem, to decompose a given hold-all domain into several subdomains. The decomposition is guaranteed to be geometrically consistent which means that no overlapping regions or voids can arise. Mathematically, the approach decouples the computational mesh from the physical domains and, thereby, allows for large deformations and topological changes, such as the rise of (new) subdomains. At domain boundaries, quantities, or their gradient, may be modeled discontinuously and eXtended approximation spaces are introduced for the (sharp) representation of such features on the discrete level. The enrichment is realized by Heaviside functions which are defined subject to the hierarchical level set functions and, hence, introduce additional basis functions and coefficients locally at the respective (sub)domain boundary. For imposing interface and boundary conditions, the Nitsche method is used.

By design, the developed approach is well suited to be implemented using automated code generation. As a result, the hierarchical eXtended finite element method is implemented as toolbox `miXFEM` for the `FEniCS` framework. Therefore, the core components of `FEniCS` are significantly extended and new methods (e.g. the subdivision of elements and the assembling of tensors) are added. As the evolution of interfaces is often part of the problem, the framework `miXFEM` is supplemented by a level set toolbox providing maintaining methods such as reinitialization and volume correction as well as methods for computing a non-material velocity field.

The method and its implementation is validated against several examples and then used for the modeling and simulation of different real-world applications in 2d and 3d. Since this thesis is motivated by several research projects where melting and solidification processes are of interest, we focus on these kind of problems and present results for a thermal upsetting process and different welding processes. However, due to the generality and flexibility of the developed framework, it can be used to rapidly implement and simulate problems from different areas such as multiphase flow or other problems with evolving geometries.

Zusammenfassung

An automated hierarchical eXtended finite element approach for multiphysics problems involving discontinuities

by Mischa JAHN

In der vorliegenden Arbeit wird eine hierarchische erweiterte Finite-Elemente-Methode für die Modellierung und Simulation von multiphysikalischen Problemen und deren Implementierung in ein Framework, welches das Konzept der automatisierten Codegenerierung nutzt, vorgestellt. Die Methode basiert auf der physikalisch motivierten Definition von hierarchisch geordneten Levelset-Funktionen, die ein gegebenes Gebiet in Teilgebiete zerlegen. Diese Zerlegung ist geometrisch konsistent, so dass keine Gebietsüberschneidungen oder Löcher entstehen können, und wird zur Entkopplung des Rechengitters von der eigentlichen physikalischen Geometrie genutzt. Dies erlaubt die Beschreibung von großen Gebiets- und auch Topologieänderungen. Um physikalische Größen dennoch exakt darzustellen und eventuelle Unstetigkeiten über Gebietsgrenzen zu berücksichtigen, werden erweiterte Finite-Elemente-Approximationsräume verwendet. Die Erweiterung um zusätzliche Basisfunktionen und Koeffizienten wird durch hierarchisch definierte Heaviside-Funktionen erreicht. Um Rand- und Interface-Bedingungen zu setzen, wird die Nitsche Methode verwendet.

Nach Konstruktion ist der entwickelte Ansatz gut für die Verwendung von Ansätzen aus dem Bereich der automatisierten Codegenerierung geeignet. Daher wird die hierarchische erweiterte Finite-Elemente-Methode als Toolbox `miXFEM` für das `FEniCS` framework implementiert. Dazu werden sämtliche Kernkomponenten von `FEniCS` erweitert und viele zusätzliche Methoden eingeführt, z. B. um geschnittene Elemente zu zerlegen oder die Matrizen und Vektoren aufzustellen. Die Evolution von Teilgebieten und deren Ränder ist häufig Teil der Lösung eines multiphysikalischen Problems, daher werden zusätzliche Methoden zur Pflege der Levelset-Funktionen, wie Reinitialisierungs- und Volumenerhaltungsmethoden, bereitgestellt. Die Implementierung der genannten Methoden erfolgt jedoch als eigenständige Levelset-Toolbox. Zudem werden Routinen zur Berechnung nicht-materieller Geschwindigkeitsfelder mitgeliefert.

Die Validierung der entwickelten Methode und deren Implementierung geschieht durch die Betrachtung zahlreicher Beispiele. Anschließend werden verschiedene Anwendungsbeispiele in 2D und 3D betrachtet. Da diese Arbeit durch verschiedene Projekte mit dem Fokus auf Schmelz- und Erstarrungsvorgänge motiviert wird, beschränken wir uns hauptsächlich auf die Modellierung und Simulation von Problem aus diesem Bereich, wie das Laser-Stoffanhäufen und Schweiß-Prozesse. Dennoch sei hervorgehoben, dass der vorgestellte Ansatz und das entwickelte Framework für die Modellierung und Simulation beliebiger Probleme, wie zum Beispiel im Bereich der Mehrphasenströmung, eingesetzt werden kann.

Acknowledgements

The present thesis originates from my work at the Center for industrial mathematics (ZeTeM) at the University of Bremen and was financially supported by the DFG (German Research Foundation) through the Collaborative Research Center 747 “Micro cold forming”.

First of all, I would like to express my gratitude to my advisor Prof. Dr. Alfred Schmidt for his support throughout my doctoral studies and project work. His advice and feedback has always been very helpful and is greatly appreciated. I also want to thank my co-advisor, Dr.-Ing. Jonathan Montalvo-Urquizo, for his efforts related to this thesis and this support during my research visits at the Centro de Investigación en Matemáticas A.C. (CIMAT) in Monterrey, Mexico.

My special thanks go to my colleague Andreas Luttmann who is probably one of the best office mates one can hope for (and also makes very delicious pizza). Our many discussions on and off-topic have been an essential element for doing this research and writing this thesis.

I am very grateful for the productive research environment at the ZeTeM and want to, in particular, acknowledge the discussions of mathematical and modeling problems with Simon Grützer and Michael Eden as well as their willingness to proof-read this thesis.

Last but actually most importantly, I thank my family and my family in-law for all the support during my studies. Especially, I thank my wife Anki for her love and patience who, together with Laura, can make me relax and forget about mathematical problems or implementation issues in seconds.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

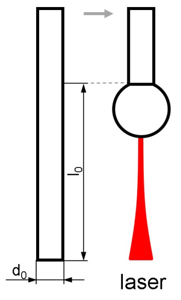
In modern engineering, various complex processes can be modeled by systems of partial differential equations (PDEs) which may be nonlinear and coupled. Problems involving multiple and coupled physical phenomena, such as solid-liquid phase transitions with fluid flow in the liquid, on various (sub)domains, are of particular interest. Such processes usually involve moving boundaries and interfaces and are referred to as *multiphysics problems* in this thesis. From a macroscopic point of view, material characteristics are often non-smooth across such interfaces and boundaries so that gradients of affected field quantities or the primary field quantities exhibit jumps. Therefore, interfaces and boundaries are modeled sharply and the involved quantities are modeled discontinuously. While examples for multiphysics problems can be found in various fields such as multiphase flow and situations with fluid-structure-interaction, this work is mainly motivated by some engineering applications that involve melting and solidification processes.

1.1 Research projects motivating this thesis

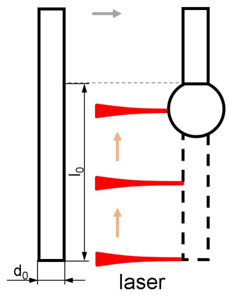
This thesis is motivated by different projects that have been worked on as multidisciplinary work between engineers and mathematicians. In all of these projects, the melting and solidification of different materials is considered which may involve a capillary boundary.

1.1.1 Laser-based thermal upsetting processes

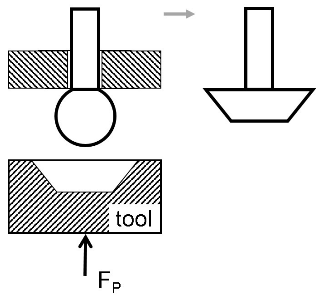
Within the Collaborative Research Center 747 "micro cold forming", industrial manufacturing methods are developed that make possible the production of hundreds or even thousands of metallic micro components per minute. Usually, these parts are smaller than 1 mm in two dimensions so that so-called size effects [6] have to be considered which are defined as "deviations from intensive or proportional extrapolated extensive values of a process which occur, when



(a) Coaxial process design.

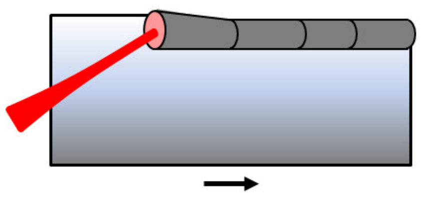


(b) Lateral process design.

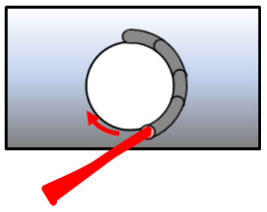


(c) Forming step after solidification.

Figure 1.1 Laser rod end melting.



(a) Sketch of process design to generate continuous and discontinuous preforms.



(b) Sketch of process design to generate continuous and closed preforms at voids.

Figure 1.2 Laser blank rim melting.

scaling the geometrical dimensions". While the occurrence of size effects prevents the application of conventional processes used in macro scale, they can be utilized for the development of new processes. One example of such a process is the laser-based thermal upsetting process, which is the first stage of an alternative cold forming process for the generation of functional parts in micro scale.

The general idea of the developed two-stage process is to generate a so-called preform in a thermal upsetting step and then apply a subsequent forming step in a open die. The thermal upsetting process, which is also called master forming process, is based on applying a laser heat source to the workpiece, melting the material and taking advantage of the shape-balance effect, which means that surface tension dominates gravity force. This two-stage cold forming process, which can generate upset ratios $s \gg 500$ in contrast to conventional multi-stage cold forming that is limited to $s \approx 2$, is described and analyzed in various publications, e.g. [7–13], and has been successfully applied to rods, blank rims, and pieced blanks of steel 1.4301.

Laser rod end melting: When considering rods, the idea behind the process is to upset a certain length l_0 of a rod with diameter d_0, which usually is between 0.1 mm and 0.5 mm, by applying a laser beam to the material. Depending on the process configuration, this can be done coaxially, see Figure 1.1(a), or laterally, see Figure 1.1(b), while in the latter case, the beam is deflected along the material. The laser beam melts parts of the rod and the molten material forms spherically due to the shape-balance effect. This arising geometry is preserved even after solidification so that the generated preform can be formed in, for example, an open die in a second step, see Figure 1.1(c).

Laser blank rim melting: The thermal upsetting process can also be applied to blanks whose thickness may vary from 25 μm to 0.5 mm. The process is based on deflecting a laser beam along the blank rim, thereby melting the upper part of the work piece. Depending on the choice of process parameters such as the laser power and the deflection velocity, continuous cylindrical preforms and discontinuous preforms can be generated. The process can also be used to generate collars at blank voids, see Figure 1.2(b).

1.1.2 Production of laser welded hybrid joints

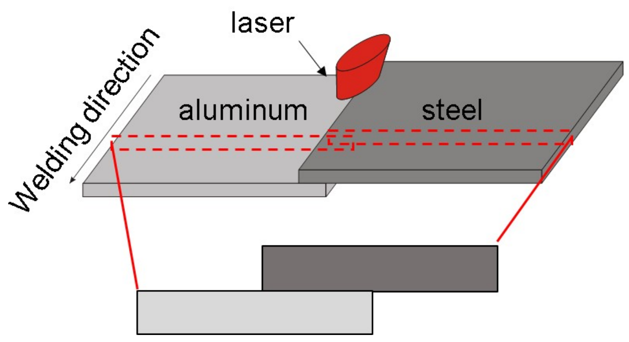
Another industrial application that can be modeled using an approach similar to the laser-based thermal upsetting process is the welding of hybrid joints. These hybrid joints offer certain economic advantages and their characteristics depend on, among others, the material behavior, inter metallic compounds, and the used process parameters and have to be carefully analyzed [14]. Within an AiF funded project (IGF 360 ZN), the production and quality of laser welded hybrid joints consisting of aluminum 3.2315 and steel 1.0338 as well as aluminum 3.2315 and titanium 3.7165 have been considered. Within the project, the process, the material, and the structure simulations have been coupled in order to predict the weld seam geometry which gives the limits of the specimen's stability and resilience. Produced specimens, which can be overlap joints and butt joints, are widely used within the automotive and aerospace industries.

Overlap joints: For the production of overlap weld joints, steel and aluminum sheets with thickness of around 1 mm (steel) and 1.5 mm (aluminum) are placed with a small overlapping region whose length is around 3 mm to 5 mm in most configurations. A defocused laser beam is then applied to the steel sheet. While the steel does not melt, the energy is conducted to the aluminum which melts since it has a much lower melting temperature than steel. The process setup is visualized in Figure 1.3.

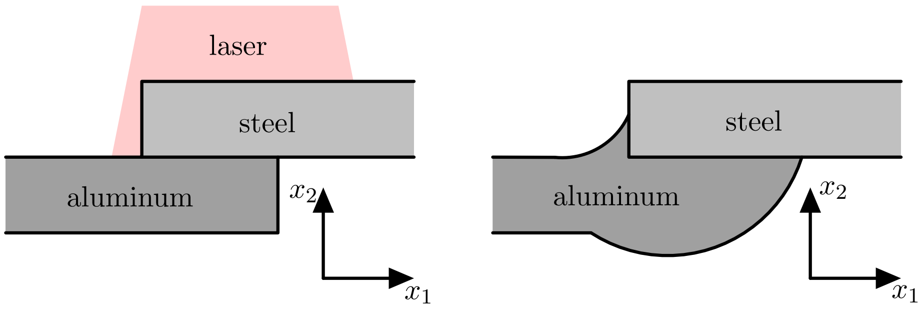
Butt joints: The process design when producing butt joints of aluminum and titanium is based on including a notch within the aluminum sheet and positioning the sheets vertically, where the titanium is inserted into the notch. Thereby, both materials overlap on each side, see Figure 1.4. Two laser sources are used to heat this region, melting the aluminum which then wets the titanium and, finally, create a hybrid joint. Typically, the thickness of the titanium sheets are between 0.9 mm and 2.0 mm and the thicknesses of the aluminum sheets varied between 1.2 mm and 3.0 mm.

1.1.3 Keyhole-based laser welding

In contrast to heat conduction welding, which is, for example, used for the production of laser welded hybrid joints, some applications need a narrow but deep weld seam geometry and a small heat affected zone. This can be achieved by applying a focused laser beam to the material, see Figure 1.5. If the intensity of the laser beam is high enough, a narrow and long hole, the so-called keyhole, is formed in the material, see Figure 1.6. In the keyhole, which follows the motion of the laser along the melting line, the laser beam is reflected multiple times so that the overall absorption rate is significantly increased while, at the same time, the heat-affected zone is locally limited. As a result, a small weld seam with a high depth can be created and a high process speed can be achieved. The keyhole-based laser welding process has been successfully applied to various materials and is already fit for mass production. However, there is still need

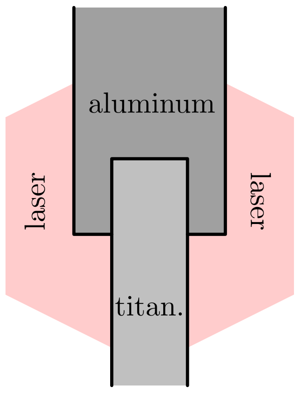


(a) Process design for overlap joint.

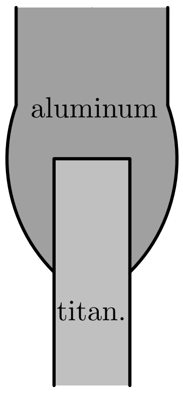


(b) 2D cross section at process start and process end.

Figure 1.3 Laser welded hybrid joint: Sketch of process design and realization for an overlap joint.



(a) Process start



(b) Process end

Figure 1.4 2D cross section of the butt joint at process start and process end.

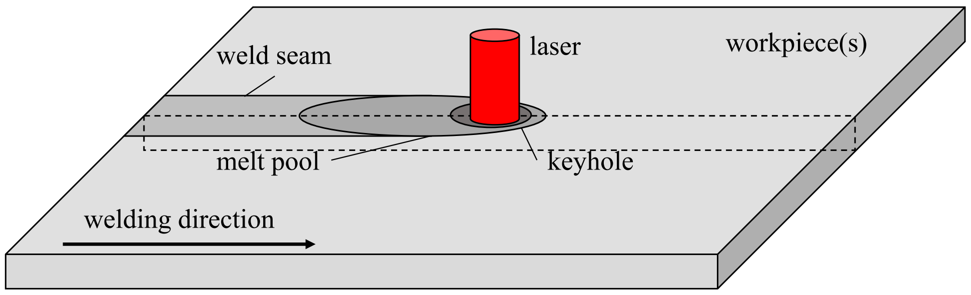


Figure 1.5 Process sketch of keyhole-based laser welding.

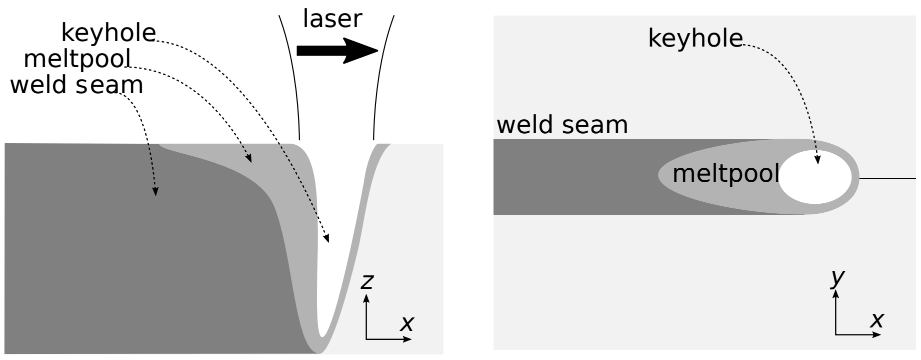


Figure 1.6 Side and top view of keyhole welding.

for simulating these processes, e.g., in order to optimize the weld seam and, hence, the resulting product.

1.2 Modeling and numerical solution of multiphysics problems

In general, engineering and industrial processes often include multiple phases whose (unknown) evolution is part of the problem and is therefore given implicitly by coupled equations describing different physical phenomena. For the modeling, these domain boundaries are often considered as discontinuities across which field quantities exhibit a jump or kink. In regards to the mentioned applications, all processes can be modeled with PDEs by coupling the Stefan problem [15, 16] with the Navier-Stokes equations including a free capillary surface [17]. The Stefan problem allows for the modeling of melting and solidification processes and the Navier-Stokes equations describe the fluid flow within the molten material and the evolution of the melt pool's surface. Each process model of the presented applications then consists of a system of coupled PDEs with suitable initial and boundary conditions.

PDE-based problems can be solved numerically using finite differences [18, Chap. 2], finite volumes [19], and finite elements [18, Chap. 3]. In this thesis, we only consider (variants of) the latter, which is the most common approach. Within the finite element method (FEM), a PDE is written in a variational form and discretized on a mesh so that the task of solving the problem comes down to computing the solution of an algebraic equation for some discrete coefficients. In order to get the full solution on the whole domain, the computed values are interpolated using the finite element basis functions.

A principle drawback of using the finite element method is that its implementation can be difficult, error-prone, and is usually very time consuming. This is because small changes of the problem formulation or discretization often result in many code changes. Also, the development of a more general framework that allows tackling different PDE-based problems instead of solving a particular one requires a lot of effort and testing, especially when specific methods are needed for a small subset of problems. Due to this, more and more frameworks make use of the idea to, at least partly, automate the generation of code. The basic concept of *automated code generation* is to write and use meta-programs which can interpret a higher level of abstraction and generate corresponding lower level code. Using automated code generation has the advantage of consistent, high quality code while also allowing for so-called abstract coding, whereby the user can write high-level code which is translated to low-level code. Thus, the approach allows for rapid prototyping and the solution of very different problems using the same framework. However, the consequence of the flexibility and usability of such approaches is that writing the programs which have to generate the code is a rather complex and time-consuming task.

In the past, all previously mentioned applications motivating this thesis have been tackled using numerical methods especially tailored to fit the models specific requirements despite the similarities of the respective application's model.

(Method 1) For the thermal upsetting process, a combined enthalpy-ALE method has been developed [9, 20, 21].

(Method 2) For the welding of hybrid joints, an enthalpy method has been used where wetting angles have been prescribed [14, 22].

(Method 3) For the keyhole-based laser welding, the approach is based on considering the heat equation where a precomputed keyhole geometry is used for imposing boundary conditions using on a adaptively refined mesh [14, 22].

The use of different methods particularly adapted to the application has been essential. This is because although the finite element method can be used for a wide range of physical and engineering problems, its application to multiphysics problems involving discontinuities is very limited. The reason for this is that conventional methods can only represent discontinuous quantities accurately when the interface and boundaries align with mesh boundaries. While this compatibility of mesh and (internal) domain boundaries can easily be guaranteed for steady-state situations by creating an appropriate mesh, much more complex approaches are needed to consider time dependent problems involving discontinuities. This is because in time-dependent situations, not only the mesh has to be moved and adapted according to the movement of the interfaces and boundaries, which is already very complex if more than two phases are involved, but also topological changes may arise and have to be considered.

Due to these shortcomings of the conventional finite element method, a lot of effort has been put in developing so-called *eXtended discretization methods* (XDMs). Instead of adapting the computational mesh during the simulation, the core idea of these methods is to decouple the computational mesh from the physical geometry. For this purpose, indicator functions are introduced which allow for the mesh independent representation of discontinuities and, thereby, separating the domain into subdomains. Based on the locations of the discontinuities, additional basis functions are added to the FEM-based approximation spaces, which are then called *extended* or *enriched*. These additional basis functions, which are added to accurately consider the discontinuous features at the domain boundaries and interfaces, are usually constructed by multiplying already included (standard) basis functions of an approximation space with a so-called *enrichment function* particularly chosen to describe the feature which has to be considered.

While the combination of eXtended discretization methods with indicator functions is a very interesting approach to model and solve problems with moving or evolving domains, handling problems which involve multiple domains separated by different boundaries and interfaces is still complex. Namely, issues arise concerning the domain decomposition and the enrichment scheme. As the sign of one indicator function can only decompose a domain into two subdomains, more indicator functions are needed to separate a given domain into several, meaning more than two, subdomains. Therefore, complex situations can arise, such as multi-junctions, where multiple interfaces or boundaries meet and intersect each other. Moreover, additional effort is required to ensure that the method, which has to allow the independent evolution of several discontinuities respectively indicator functions, is geometrically consistent, meaning that no voids or overlapping domains arise during the simulation. In regards to adding basis functions to the approximation space, the arbitrary evolution of the subdomains requires a robust approach so that the discretization allows for stable systems of equations. Therein, the process of enriching basis functions must be able to take into account multiple discontinuities.

A consequence of the arising difficulties when multiple subdomains are considered is that, to the best of the author's knowledge, most articles in literature derive a complete approach only for two-phase situations and, at most and only in a few cases, comment on how to extend the respective approach to tackle problems involving more domains. Instead, most approaches extend the discrete approximation space by introducing only one enrichment function containing the locations of all discontinuities to avoid typical problems caused by the enrichment scheme such as linear dependency. Hence, there are only few articles considering n-phase problems with $n > 2$.

With respect to the applications mentioned in Section 1.1, we want to exemplarily mention [23]. In this work, the micro structure evolution during the solidification process of multi-component alloys is modeled and simulated. For this, multiple level set functions are used and ideas of interface tracking methods are incorporated to allow for a consistent domain description.

Besides melting and solidification processes, where the additional challenge arises that we have non-material velocities moving the interfaces and material velocities that describe the fluid flow within the melt, an important research area where problems with multiple discontinuities arise is multiphase flow. An extensive overview of an approach called cut finite elements, which is very similar to the eXtended finite element method as pointed out in Remark 2.4, is given in [24]. However, in contrast to melting and solidification processes, multiphase flow problems are driven by material velocities which are usually continuous across interfaces. Therefore, the geometric consistency of a domain is implicitly conserved.

1.3 Research objectives

The overall objective of this thesis is to provide a flexible and robust framework, based on the combination of the level set method with the eXtended finite element method, for the modeling and simulation of multiphysics problems involving arbitrary discontinuities. There are three main challenges which have to be considered,

(Challenge 1) the geometrically consistent definition and representation of subdomains which may evolve or vanish in time,

(Challenge 2) a robust enrichment scheme that can consider an arbitrary number of discontinuous features of a quantity, involving jumps on its values or on the values of its derivative, and

(Challenge 3) the computational implementation of such a method into a flexible framework to tackle very different problems.

This thesis addresses these three issues for the numerical solution of problems involving multiple domains and evolving discontinuities by

(Task 1) introducing a domain decomposition method which is inspired by [25] and based on hierarchically ordered level set functions,

(Task 2) defining of corresponding hierarchically enriched approximation spaces using Heaviside enrichment [26] where interface conditions are imposed using Nitsche's method [27],

(Task 3) implementing the resulting so-called hierarchical eXtended finite element method into a framework using automated code generation [1], and

(Task 4) using this method and the framework for the solution of real-world multiphysics problems such as the applications motivating this thesis.

Therefore, the idea is to introduce a hierarchical level set method which allows the handling of any domain configuration, which may change in time in a consistent way, meaning that no special methods are required to prevent overlapping domains or the occurrence of voids. The domain boundaries are represented by (parts of) the zero level sets of the level set functions and discrete finite element approximation spaces are locally enriched using Heaviside enrichment. For imposing interface conditions, Nitsche's method is used. By introducing different enrichment levels that correspond to the level of hierarchy of the level set function, a robust scheme is developed which is especially suited to be implemented in a framework utilizing the idea of automated code generation. With this, the resulting extended framework can be used to rapidly model and solve all kind of PDE-based problems involving an arbitrary number of discontinuities. We demonstrate the method's capabilities by applying it to several examples with known a solution and to simulate complex industrial applications.

1.4 Outline of the thesis

The thesis is organized as follows:

(Chap. 2) In Chapter 2, we start with a brief introduction to solving PDE-based problems involving discontinuities. By limiting ourselves to a problem consisting of two subdomains separated by one interface at which a discontinuous feature occurs, the geometric situation can be described using the level set method. We present the general idea and introduce the notation used throughout this thesis. For an example, we consider an elliptic steady-state diffusion problem which is solved analytically and numerically. For the numerical solution, we successively describe the conventional finite element method, illustrate the importance of representing the discontinuity on mesh boundaries and address issues when considering time dependent problems. After that, we present the general idea of the finite element method that is based on decoupling mesh and geometry as powerful alternative and discuss challenges arising when it is used.

(Chap. 3) In Chapter 3, we develop a hierarchical eXtended finite element method, which is a robust approach to handle multiphysics problems involving arbitrary discontinuities. Therefore, we first present a geometrically consistent domain decomposition scheme that is based on hierarchically ordered level set functions. Based on these functions a conventional finite element approximation space is enriched using Heaviside enrichment. Boundary and interface conditions are imposed by Nitsche's method.

(Chap. 4) In Chapter 4, we address numerical challenges arising when solving multiphysics problem and present the implementation of the hierarchical eXtended finite element method. The developed approach is, by construction, well suited to be implemented within an automated code generation approach. For several reasons which are mentioned in the corresponding section, we choose to implement the method as a toolbox called `miXFEM` for the open source `FEniCS` framework. Therefore, we first give an overview of the `FEniCS` project and comment on a previous work to consider problems involving discontinuities. Afterwards, we present the design idea and the most important implementation aspects of the toolbox `miXFEM`. As the numerical solution of multiphysics problem often requires additional methods, we include approaches for handling evolving interfaces and methods for computing non-material velocity fields.

(Chap. 5) In Chapter 5, we present numerical results for various applications. We begin the chapter by briefly summarizing validation results obtained by considering several examples including academic problems with known analytical solution. Then, we focus on real world applications such as the generation of hybrid joint, keyhole-based laser welding, and the thermal upsetting process. By successively increasing the complexity of the considered problems, the enhanced modeling and simulation capabilities of the method and implemented framework as well as its generality and flexibility are illustrated.

(Chap. 6) Finally, the thesis is summarized in Chapter 6, and we discuss open questions and address future work.

Chapter 2

A brief introduction to solving PDE-based problems involving discontinuous features

To introduce some notation and motivate the method developed in Chapter 3, we consider the most basic setting for a multiphase problem which consists of a hold-all domain Ω that is separated into two subdomains by an interface. This setting can be described by

Definition 2.1 (Basic domain setting)**.** Let $\Omega \subset \mathbb{R}^d$ represent a physical domain consisting of two disjoint subdomains Ω_1 and Ω_2 that are separated by a sharp and sufficiently smooth internal boundary called *interface* $\Gamma_{1,2} := \text{interior}_{d-1}\left(\bar{\Omega}_1 \cap \bar{\Omega}_2\right)$ such that we have $\Omega = \Omega_1 \cup \Omega_2 \cup \Gamma_{1,2}$. The unit normal vector $\vec{n}$ is given by the outward-pointing normals $\vec{n}_i$ to $\partial\Omega_i$, $i = 1, 2$, and the unit normal at $\Gamma_{1,2}$ is defined as $\vec{n}_{1,2} = \vec{n}_1 = -\vec{n}_2$.

Using this setting, which is visualized in Figure 2.1, we briefly recall the weak solution theory and then illustrate the concepts of the conventional finite element method and the eXtended finite element method when a quantity exhibits a discontinuous feature across the interface $\Gamma_{1,2}$. Consider the following example which could result from applying Rothe's method to the (scaled) heat equation, see also Section 3.3.3.

Example 2.1 (Steady-state diffusion problem)**.** *Let $\Omega \subset \mathbb{R}^d$ be a fixed polygonally bounded domain, with $\partial\Omega = \Gamma_\mathrm{D} \cup \Gamma_\mathrm{N}$ and outwards-pointing normal $\vec{n}$, that consists of the disjoint subdomains Ω_1 and Ω_2 separated by a sharp and sufficiently smooth interface $\Gamma_{1,2}$ with unit*

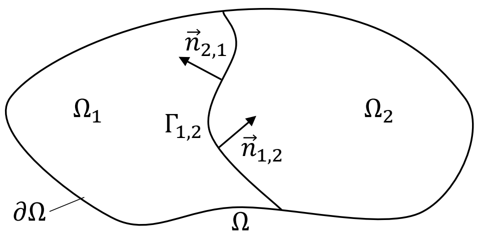


Figure 2.1 Exemplary setting as described in Definition 2.1.

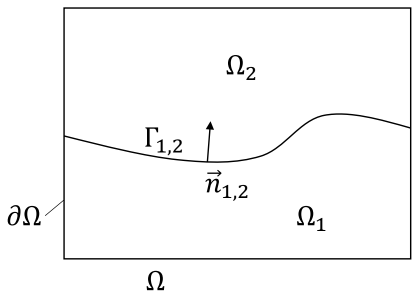


Figure 2.2 Exemplary setting for Example 2.1.

normal vector $\vec{n}_{1,2}$. For $\xi \geq 0$, $\kappa|_{\Omega_i} \in L^\infty(\Omega_i)$, and given data which is sufficiently smooth, find $u \colon \Omega_1 \cup \Omega_2 \to \mathbb{R}$ s.t. it solves the problem given by

$$\begin{aligned} \xi u - \nabla \cdot (\kappa \nabla u) &= f && \text{in } \Omega_1 \cup \Omega_2, && (2.1a)\\ u &= g_\mathrm{D} && \text{on } \Gamma_\mathrm{D}, && (2.1b)\\ -\kappa \nabla u \cdot \vec{n} &= g_\mathrm{N} && \text{on } \Gamma_\mathrm{N}, && (2.1c)\\ [\![\kappa \nabla u]\!] \cdot \vec{n}_{1,2} &= g_{1,2} && \text{on } \Gamma_{1,2}, && (2.1d)\\ [\![u]\!] &= q_{1,2} && \text{on } \Gamma_{1,2}, && (2.1e) \end{aligned}$$

for a situation similar to the setting depicted in Figure 2.2. Here, the operator $[\![\cdot]\!]$ denotes the jump of a quantity $[\![u]\!] = (u|_{\Omega_1})_{\Gamma_{1,2}} - (u|_{\Omega_2})_{\Gamma_{1,2}}$ that is given by comparing (in trace sense) the limiting value of u in Ω_1 and Ω_2 when approaching the interface $\Gamma_{1,2}$.

Variational formulation and weak solution: In order to define the weak formulation of the problem and prove that there exists a unique weak solution, we first introduce

$$U := \{v \in H^1(\Omega_1 \cup \Omega_2) \, : \, v = g_\mathrm{D} \text{ on } \Gamma_\mathrm{D},\, [\![v]\!] = q_{1,2} \text{ on } \Gamma_{1,2}\} \tag{2.2}$$

and

$$V := \{v \in H^1(\Omega_1 \cup \Omega_2) \, : \, v = 0 \text{ on } \Gamma_\mathrm{D},\, [\![v]\!] = 0 \text{ on } \Gamma_{1,2}\}. \tag{2.3}$$

The weak formulation of Example 2.1 is then given by: Find $u \in U$ such that for all $v \in V$ the integral identity

$$\int_{\Omega_1 \cup \Omega_2} \xi u v \,\mathrm{d}x + \kappa \nabla u \cdot \nabla v \,\mathrm{d}x = \int_{\Omega_1 \cup \Omega_2} f v \,\mathrm{d}x - \int_{\Gamma_\mathrm{N}} g_\mathrm{N} v \,\mathrm{d}x + \int_{\Gamma_{1,2}} g_{1,2} v \,\mathrm{d}x \tag{2.4}$$

holds, where we assume that $f \in L^2(\Omega_1 \cup \Omega_2)$, $g_\mathrm{N} \in L^2(\Gamma_N)$, $g_{1,2} \in L^2(\Gamma_{1,2})$, and $\xi, \kappa \in L^\infty(\Omega)$, with $0 < a < \kappa$, $a \in \mathbb{R}$ almost everywhere.

If $u \in U$ satisfies equation (2.4) for all $v \in V$, we call u a weak solution to Example 2.1. Unfortunately, U is not a Hilbert space and obviously $U \neq V$, therefore we can not use the integral identity to test with the solution itself to gain important estimates and apply the theorem of Lax-Milgram.

Hence, we first introduce the following (weak) auxiliary problem for $\tilde{u}$, $v \in V$

$$\begin{aligned}\int_{\Omega_1 \cup \Omega_2} \xi \tilde{u} v \,\mathrm{d}x + \kappa \nabla \tilde{u} \cdot \nabla v \,\mathrm{d}x = &- \int_{\Omega_1 \cup \Omega_2} (\xi w v + \kappa \nabla w \cdot \nabla v) \,\mathrm{d}x \\ &+ \int_{\Omega_1 \cup \Omega_2} f v \,\mathrm{d}x - \int_{\Gamma_\mathrm{N}} g_\mathrm{N} v \,\mathrm{d}x + \int_{\Gamma_{1,2}} g_{1,2} v \,\mathrm{d}x,\end{aligned} \tag{2.5}$$

summarized to

$$a(\tilde{u}, v) = L(v) \tag{2.6}$$

with bilinear form a and linear form L, where we assume that $w \in H^1(\Omega_1 \cup \Omega_2)$ and that all other functions have the same regularity as before. With this assumptions, the theorem of Lax-Milgram can be applied to equation (2.5) showing that there is a unique solution $\tilde{u} \in V$ of equation (2.5). As a result, we only have to find a function $w \in H^1(\Omega_1 \cup \Omega_2)$ so that the conditions given by equation (2.1b) and equation (2.1e) hold to solve Example 2.1. In brief, the function w is in essence an extension of the boundary value of g_D and the jump $q_{1,2}$, see Section 3.1 and the following analysis for details. With this, we define $u = \tilde{u} + w \in U$ which is a solution of Example 2.1.

2.1 Representation of interfaces and boundaries: The level set method

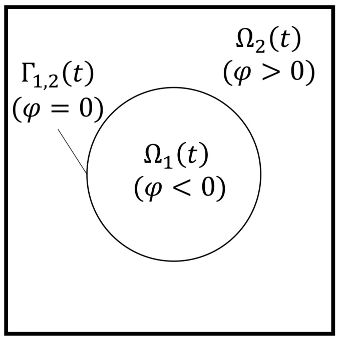
When considering analytical or numerical problems, an interface or boundary $\Gamma_{1,2}$ is usually represented by the zero level of an (explicitly given) indicator function or by an explicit parametrization. In this thesis, we exclusively use the level set method to describe interfaces and boundaries which, due to its simple idea and universalism, can be used for a very wide class of problems. Please note that while the level set method can of course also be used to describe steady-state interfaces and boundaries, we use the more general setting where all quantities may depend on time.

The basic idea of the level set method [28] is to introduce a continuous scalar function $\varphi\colon \Omega \times [t_0, t_f] \to \mathbb{R}$ on a given domain $\Omega \subset \mathbb{R}^d$, whose zero level set

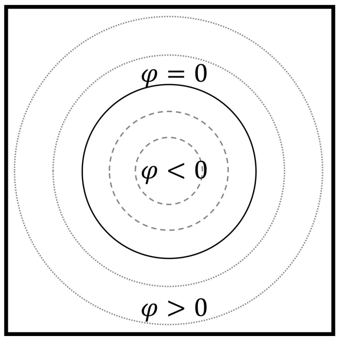
$$\Gamma_{1,2}(t) = \{\boldsymbol{x} \in \Omega : \varphi(\boldsymbol{x}, t) = 0\}, \quad t \in [t_0, t_f],$$

represents a time dependent discontinuity. By using the zero level set and the sign of $\varphi = \varphi(\cdot, t)$, the hold-all domain Ω can be divided into the *two* subdomains and the separating interface

$$\Omega(t) = \Omega_1(t) \cup \Omega_2(t) \cup \Gamma_{1,2}(t),$$



(a) Domains $\Omega_1(t)$ and $\Omega_2(t)$ are separated by the zero level set $\Gamma_{1,2}(t)$ of φ.



(b) Visualization of some level sets of φ.

Figure 2.3 Visualization of the idea of the level sets method using a scalar function φ.

with $\boldsymbol{x} \in \Omega_1(t) \Leftrightarrow \varphi(\boldsymbol{x}, t) < 0$ and $\boldsymbol{x} \in \Omega_2(t) \Leftrightarrow \varphi(\boldsymbol{x}, t) > 0$, cf. Figure 2.3.

For $\|\nabla\varphi\| > 0$ at $\Gamma_{1,2}$, the level set method allows for an easy computation of the unit normal $\vec{n}_{1,2}$ and the curvature $\mathcal{C}$ which are given by

$$\vec{n}_{1,2} = \frac{\nabla\varphi}{||\nabla\varphi||} \qquad \text{and} \qquad \mathcal{C} = \nabla \cdot \vec{n}_{1,2} = \nabla \cdot \left(\frac{\nabla\varphi}{\|\nabla\varphi\|} \right),$$

which is useful in many applications such as fluid dynamics.

There are various functions φ which can be defined and used within the level set method, however, from a numerical point of view it is important that the gradient $||\nabla\varphi||$ neither vanishes nor becomes too large. Due to this, the literature suggest to use a so called *signed distance function*, such as

$$\varphi(\boldsymbol{x}, t) = \begin{cases} -\min\limits_{\boldsymbol{y} \in \Gamma_{1,2}(t)} ||\boldsymbol{x} - \boldsymbol{y}||_2, & \text{if } \boldsymbol{x} \in \Omega_1(t) \\ \min\limits_{\boldsymbol{y} \in \Gamma_{1,2}(t)} ||\boldsymbol{x} - \boldsymbol{y}||_2, & \text{if } \boldsymbol{x} \in \Omega_2(t) \end{cases},$$

so that it is $||\nabla\varphi|| = 1$ if the interface is smooth.

Given the initial value $\varphi(\cdot, t_0)$ with zero level set $\Gamma_{1,2}(t_0)$, the evolution of the hyperbolic level set function φ and consequently of the interface $\Gamma_{1,2}$ in time can be described by the transport equation

$$\frac{\partial\varphi}{\partial t} + \vec{V} \cdot \nabla\varphi = 0, \tag{2.7}$$

where $\vec{V} = \vec{V}(\boldsymbol{x}, t)$ has to be a sufficiently smooth velocity field.

Since the transport equation itself cannot change the image of the function, which is, e.g., required to represent topology changes, an additional boundary condition on the inflow boundary $\partial\Omega_{\text{in}} := \{\boldsymbol{x} \in \partial\Omega \,:\, \vec{V}(\boldsymbol{x}, t) \cdot \vec{n}(\boldsymbol{x}) < 0,\, t \in [t_0, t_f]\}$ can be defined by a continuous function $\varphi_{\text{in}} \colon \partial\Omega_{\text{in}} \times [t_0, t_f] \to \mathbb{R}$. With this the level set problem in strong formulation is given by: Find

$\varphi \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$, s.t.

$$\begin{aligned} \frac{\partial \varphi}{\partial t} + \vec{V} \cdot \nabla \varphi = 0 \qquad & \text{in } \Omega \times (t_0, t_f), \\ \varphi(\cdot, t_0) = \varphi_0(\cdot) \qquad & \text{in } \Omega, \\ \varphi(\cdot, t) = \varphi_{\text{in}}(\cdot, t) \qquad & \text{on } \partial\Omega_{\text{in}} \times [t_0, t_f] \end{aligned} \tag{2.8}$$

holds.

2.2 A brief introduction to (fitted) finite element methods

A powerful framework for the numerical approximation of PDE problems is the finite element method (FEM). The general idea is to write a PDE in a variational form, discretize the physical domain using a simplicial mesh, and define a discrete approximation space whose basis usually consists of piecewise polynomials. Then, the task of solving a PDE-based problem comes down to computing the solution of a finite dimensional system of equations.

Given a domain Ω that is bounded polygonally, let $\mathcal{S}_h$ be a shape regular mesh consisting of d-simplices with d denoting the dimension and $h > 0$ is the maximum diameter $h = \max_{S \in \mathcal{S}_h} \operatorname{diam}(S)$ such that $\bigcup_{S \in \mathcal{S}_h} = \bar{\Omega}$. For any such triangulation $\mathcal{S}_h$, we define

$$V^m_{\text{cg},h} = \{v_h \in C^0(\Omega) : v_h|_S \in \mathcal{P}^m(S),\ \forall S \in \mathcal{S}_h\}, \tag{2.9}$$

to be the finite element approximation space consisting of continuously connected piecewise polynomials in $\mathcal{P}^m$, with $m \geq 1$ denoting the polynomial degree, where all degrees of freedom at common nodes of neighboring elements S are shared by all adjacent elements, i.e. these nodes are *single valued*. In addition, we define

$$V^m_{\text{dg},h} = \{v_h \in L^2(\Omega) : v_h|_S \in \mathcal{P}^m(S),\ \forall S \in \mathcal{S}_h\}, \tag{2.10}$$

to be the space containing discontinuous piecewise polynomials where all elements contain their own degrees of freedom so that nodes on shared edges are *multi valued*.

2.2.1 The finite element method for steady-state problems

Over the last decades, the finite element method has been very successfully used to simulate various applications that can be described by so-called *single-phase* problems. Using additional techniques, it can also be used to consider more complex problems involving two or more subdomains and coupled quantities. However, it is well known that the approximation quality and convergence behavior of the finite element method depend on the mesh and the polynomial degree of the discrete function space. Therefore, if the PDE-based problem involves a non-smooth feature, the mesh has to be conforming to the feature's location, e.g., by representing it via edges or facets, to achieve the maximal order of convergence. In the following, we consider a steady-state heat equation on a domain consisting of two subdomains separated by a sharp interface where solution exhibits a kink at the interface. This specific version of Example 2.1 is used for illustrating the convergence behavior.

Example 2.2 (Specific version of Example 2.1)**.** *Consider Example 2.1 on* $\Omega = (0,1)^2$ *with* $\partial\Omega = \Gamma_\mathrm{D} \cup \Gamma_\mathrm{N}$*, where* $\Gamma_\mathrm{D} = [0,1] \times \{0\} \cup [0,1] \times \{1\}$ *and* $\Gamma_\mathrm{N} = \partial\Omega \setminus \Gamma_\mathrm{D}$*. We define the subdomains by* $\Omega_1 = (0,1) \times (0,0.5)$ *and* $\Omega_2 = (0,1) \times (0.5,1)$ *separated by* $\Gamma_{1,2} = (0,1) \times \{0.5\}$*, cf. Figure 2.4. For* $\kappa|_{\Omega_1} = 2$*,* $\kappa|_{\Omega_2} = 1$*, and* $\xi = 0$*, we choose the right-hand-side* f *and the boundary conditions* g_D *and* g_N *as well as the flux* $g_{1,2}$ *and the jump* $q_{1,2}$ *across* $\Gamma_{1,2}$ *such that*

$$u(\boldsymbol{x}) = u(x,y) = \kappa\left(\exp(\kappa\varphi(\boldsymbol{x})) - 1\right) \tag{2.11}$$

with $\kappa|_{\Omega_1} = 2$ *and* $\kappa|_{\Omega_2} = 1$ *and*

$$\varphi(\boldsymbol{x}) = \varphi(x,y) = y - 0.5 \tag{2.12}$$

is a solution of Example 2.1.

We approximate the solution of this problem with the finite element method. Let $\mathcal{S}_h$ *be a triangulation covering* Ω_h *with* $\Omega_h = \Omega$ *and* $\Gamma_{\mathrm{D},h} = \Gamma_\mathrm{D}$ *and* $\Gamma_{\mathrm{N},h} = \Gamma_\mathrm{N}$ *since* Ω *is polygonally bounded. Since it is* $q_{1,2} = 0$ *in equation (2.1e) for the given setting, we expect the solution of the given problem to be continuous. Consequently, we choose* $V^m_{\mathrm{cg},h}$ *as discrete approximation space. After replacing all quantities with their discrete counterparts respectively their finite element approximation, the discretized variational formulation reads:*

Find $u_h \in V^m_{\mathrm{cg},h}$ *with* $u_h = g_{\mathrm{D},h}$ *on* Γ_D *s.t. for all* $v_h \in V^m_{\mathrm{cg},h}$

$$\underbrace{\int_{\Omega_h} \kappa_h \nabla u_h \cdot \nabla v_h \,\mathrm{d}x}_{a(u_h,v_h)} = \underbrace{\int_{\Omega_h} f_h v_h \,\mathrm{d}x - \int_{\Gamma_\mathrm{N}} g_{\mathrm{N},h} v_h \,\mathrm{d}x + \int_{\Gamma_{1,2,h}} g_{1,2,h} v_h \,\mathrm{d}x}_{L(v_h)} \tag{2.13}$$

holds.

Approximation quality and convergence order on a fitted mesh

It is well known that the approximation quality and, hence, the order of convergence of the finite element method with respect to the approximation error depends not only on the polynomial degree of the basis functions but also on the computational mesh which has to resolve boundaries and interfaces on mesh facets or edges. On a fitted mesh, the L^2 and H^1-semi approximation

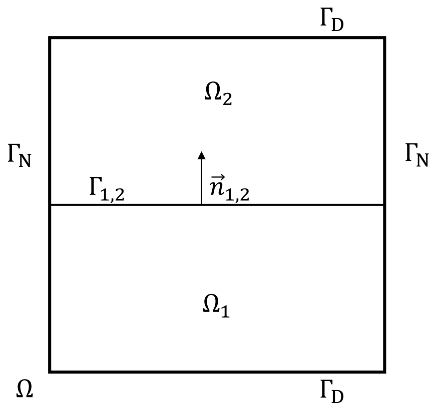


Figure 2.4 Setting for Example 2.2.

N_{el}	$\inf_{v_h\in V^1_{\text{cg}}} \|u-v_h\|_{L^2(\Omega)}$	eoc	$\inf_{v_h\in V^1_{\text{cg}}} \|u-v_h\|_{H^1(\Omega)}$	eoc
2×4^2	0.021575	-	0.273396	-
2×8^2	0.005449	1.9864	0.137909	0.9878
2×16^2	0.001366	2.0000	0.069109	0.9989
2×32^2	0.000342	2.0138	0.034574	1.0077
2×64^2	0.000085	2.0568	0.017289	1.0350
2×128^2	0.000021	2.2524	0.008645	1.1608

Table 2.1 Approximation errors and estimated order of convergence for V^1_{cg} using a fitted mesh.

errors of the problem at hand can be estimated by

$$\inf_{v_h\in V^m_{\text{cg}}} \|u-v_h\|_{L^2(\Omega)} \leq c_1 h^m \|u\|_{H^m(\Omega_1\cup\Omega_2)}, \tag{2.14}$$

$$\inf_{v_h\in V^m_{\text{cg}}} \|\nabla u-\nabla v_h\|_{L^2(\Omega)} \leq c_2 h^{m-1} \|u\|_{H^m(\Omega_1\cup\Omega_2)}, \tag{2.15}$$

for $m\geq 1$. For illustration, we consider Example 2.2 for $m=1$ on a uniform triangulation with $N_{\text{el}} = \{2\times 4^2, 2\times 8^2, 2\times 16^2, 2\times 32^2, 2\times 64^2, 2\times 128^2\}$ elements so that the interface $\Gamma_{1,2,h}$ align with facets of elements, i.e. $\Gamma_{1,2,h}=\Gamma_{1,2}$. The convergence order is then as expected, see Table 2.1.

Approximation quality and convergence order on an unfitted mesh

If the triangulation is unfitted to the problem, the expected orders of convergence for the approximation errors decreases. According to [29, Sec. 7.9.1], we expect the following estimates

$$\inf_{v_h\in V^m_{\text{cg}}} \|u-v_h\|_{L^2(\Omega)} \leq c_1 h^{\frac{3}{2}} \|u\|_{H^m(\Omega_1\cup\Omega_2)}, \tag{2.16}$$

$$\inf_{v_h\in V^m_{\text{cg}}} \|\nabla u-\nabla v_h\|_{L^2(\Omega)} \leq c_2 \sqrt{h}\, \|u\|_{H^m(\Omega_1\cup\Omega_2)}. \tag{2.17}$$

N_{el}	$\inf_{v_h \in V_{\text{cg}}^1} \lVert u - v_h \rVert_{L^2(\Omega)}$	eoc	$\inf_{v_h \in V_{\text{cg}}^1} \lVert \nabla u - \nabla v_h \rVert_{L^2(\Omega)}$	eoc
2×5^2	0.033778	-	0.603654	-
2×9^2	0.014919	1.3902	0.470692	0.4233
2×17^2	0.005953	1.4444	0.348421	0.4730
2×33^2	0.002243	1.4711	0.253769	0.4780
2×65^2	0.000820	1.4851	0.183075	0.4819
2×129^2	0.000296	1.4866	0.130949	0.4893

Table 2.2 Approximation errors and estimated order of convergence for V_{cg}^1 using an unfitted mesh.

to hold for $m \geq 1$ for the given example. Choosing $N_{\text{el}} = \{2 \times 5^2, 2 \times 9^2, 2 \times 17^2, 2 \times 33^2, 2 \times 65^2, 2 \times 129^2\}$ as number of elements, the resulting mesh does not resolve the interface $\Gamma_{1,2,h}$ and the numerical results confirm the expected error bounds, see Table 2.2.

2.2.2 The finite element method for time-dependent problems

Solving time-dependent problems with the finite element method is more complex since the problem has to be discretized in both space and time. In principle, there are two concepts available to tackle such problems, *coupled space-time* approaches [30] and the *method of lines* [31, 32]. Roughly speaking, space-time methods treat the time as additional spatial dimension and, thus, increase the dimension of the considered problem. For the numerical solution of time dependent problems using this approach, we then have to discretize the problem in the space-time domain which is usually done on so-called space-time slabs. Therefore, a conventional finite element discretization for the spatial discretization is usually combined with a discontinuous Galerkin time stepping scheme. In contrast to this, the methods of lines is based on successively discretizing both the space and the time dimension. In fact, the method of lines comes with two variants, the vertical method of lines and the horizontal method of lines, better known as Rothe's method. The vertical method of lines is based on leaving the time variable continuous and only discretizing the problem in space. Thereby, a system of ordinary differential equations (ODEs) is generated which can be solved using an explicit or implicit ODE-solver. The horizontal method of lines discretizes the time first and results in a sequence of quasi-steady-state problems, which can be solved using the finite element method.

While the use and implementation of a suitable discretization and solution technique as described in the previous paragraph already requires some effort, considering time-dependent problems with moving interfaces and boundaries is even more complex as we have to represent the boundaries during the solution process. In contrast to steady-state situations where all boundaries and interfaces are fixed and often explicitly given, these quantities are usually part of the solution when considering time-dependent problems. This is especially true for multiphysics problems involving several subdomains. Therefore, we need methods to represent boundaries and interfaces. In general, there are two types of methods, explicit *interface tracking* approaches and implicit *interface capturing* approaches [33]. Both are very important concepts used when tackling time-dependent problems.

Interface tracking: The explicit interface tracking method relies on representing an interface as a polygonal line on mesh boundaries. Any movement of the interface described by an equation is then resolved by moving the mesh accordingly. The most common approach to track the interface is the arbitrary Lagrangian Eulerian (ALE) method [34]. Therein, the time derivate is considered with respect to a deforming mesh configuration which results in a modified convection term that additionally takes the mesh velocity into account. The obvious advantage of interface tracking methods is that we always have a mesh fitted to the problem and subdomains due to the representation of the interface by mesh boundaries. However, for problems involving a large geometrical deformation, it might be necessary to include a remeshing technique whose implementation can be complex and which is numerically expensive. Moreover, tracking methods basically only move and, with limitations, evolve the interface that has to be given by an initial configuration but cannot take into account topological changes that arise in most multiphysics problems such as multiphase-flow or melting and solidification processes.

Interface capturing: In contrast to tracking methods, the interface capturing approach is based on considering an indicator function, such as a signed distance function, on a fixed mesh and define subdomains, e.g., by the sign of this function. As the interface and subdomains do not dependent on the mesh boundaries, this approach is *unfitted* and quantities on intersected elements have to somehow be approximated. While the simplest approximation is to assign each element completely to one of the subdomains, more elaborated approaches use an average value that may, for example, depend on the ratio of the subvolumes corresponding to each domain. In general, interface capturing methods are easy to implement and offer a high flexibility, as they can handle topological changes naturally. However, due to the fact that the mesh is not fitted to the problem, only moderate convergence rates can be expected as exemplarily shown for the steady-state heat equation in the previous section.

A concept that can be used within both approaches is adaptivity [35]. Usually, the concept of adaptive finite element methods is used to automatically refine and coarsen a mesh to compute a solution meeting a given error bound by using a posteriori error estimations. However, it can also be used to refine the mesh based on geometrical data such as the position of the interface. In regards to time-dependent problems, refining the mesh in the vicinity of the interface allows for a better approximation of the local discontinuous feature. Unfortunately, the mesh refinement often results in a lot of additional basis functions and degrees of freedom so that the size of the corresponding discrete system can blow up.

2.3 A brief introduction to eXtended discretization methods

Due to the mentioned issues and limitations of conventional finite element methods, especially for time-dependent multiphysics problems, the scientific community put a lot of effort in developing so-called *eXtended discretization methods* (XDMs). The core idea of eXtended discretization methods is to decouple the computational mesh from the physical geometry by extending or enriching the FEM-based approximation space instead of adapting the computational mesh. Consequently, these methods belong to the class of *unfitted methods*. The enrichment of the approximation space is based on a partition-of-unity concept and introduces additional basis functions to accurately represented discontinuous features on meshes which are independent from the respective (sub)domain geometry. For this to work, the approach takes advantage of a priori known properties of the discontinuity, for example the type indicating whether a function itself is discontinuous (strong discontinuity) or whether the gradient of a function is discontinuous (weak discontinuity) and incorporates this knowledge into the extended approximation space. In practice, boundaries and interfaces are usually represented using implicit interface capturing approaches such as the level set method, described in Section 2.1.

2.3.1 A short historical background of unfitted finite element methods

While the first unfitted finite element methods have been developed and analyzed for Dirichlet boundary conditions [36–38] a few decades ago, the development of numerical approaches to solve such problems was only initiated in the mid of the 1990's by [39, 40]. Therein, a partition-of-unity method (PUM) has been used to include additional knowledge of the problem into the function space. Based on this idea, the two (very similar) concepts, generalized finite element method (GFEM) [41, 42] and eXtended finite element method (XFEM) [43, 44], were simultaneously developed with the essential difference that in XFEM only a subset of the degrees of freedom and, hence, only parts of the domain is enriched. Since then various enrichment methods with sometimes only small differences have been developed and discussed in the literature, see, e.g., [26, 45] for reviews. However, all methods can be roughly classified by the criteria [26] whether

- the basis functions are mesh-free or mesh-based,
- the enrichment is extrinsic, meaning that new functions are added to the function space, or intrinsic, i.e. (some) basis functions are replaced by other functions, and whether
- the enrichment is performed globally or locally.

In this thesis, we focus on the eXtended finite element method which uses an mesh-based local extrinsic enrichment that is locally realized by the partition-of-unity concept. While the eXtended finite element method was originally introduced to numerically tackle structural mechanics problems where cracks are present, it has been successfully applied to various problems, e.g. solidification processes involving moving interfaces [46, 47] and two-phase flow [29]. Overviews of applications which have been numerically solved with this method can be found in i.a. [26, 45, 48]. A consequence of decoupling mesh and geometry when using XFEM and related methods is the rise of several challenges in regards to the numerics that have to be dealt with. A brief summary including references to the significant articles is given in Section 2.3.3.

2.3.2 The eXtended finite element method

We present the idea of the eXtended finite element method using similar notations as in [24, 29, 49, 50]. Consider a situation as depicted in Section 2.2.1, that means we have an hold-all domain Ω which is separated into two subdomains Ω_1 and Ω_2 by an interface $\Gamma_{1,2}$ which, for the ease of notation, we now denote with Γ in the remainder of this chapter. As before let $\mathcal{S}_h$ be a shape regular mesh consisting of d-simplices with maximum diameter $h = \max_{S\in\mathcal{S}_h} \operatorname{diam}(S) > 0$ so that $\bigcup_{S\in\mathcal{S}_h} = \bar{\Omega}$. Since in general we cannot expect that an interface Γ coincides with facets of elements, we request that the curvature of the interface is bounded and that the resolution of the mesh near the interface is sufficiently high to prevent multiple intersections of an element edge, see [49]. Furthermore, we introduce the following notation that will be used (and extended) in the next chapters:

For any simplex $S \in \mathcal{S}_h$, $\Gamma_S = \Gamma \cap S$ denote the intersection segment of Γ in S, $\mathcal{S}_h^\Gamma := \{S \in \mathcal{S}_h : S \cap \Gamma \neq \emptyset\}$ is the set of intersected elements, and $\Omega^\Gamma := \bigcup_{S\in\mathcal{S}_h^\Gamma} S$ is the corresponding domain. Moreover, $S_i = S \cap \Omega_i$ is the part of the element which belongs to Ω_i, $i = 1, 2$.

Remark 2.2 (A comment on the notation). Please note that the notation used within this section also remains valid if the discontinuity aligns with element boundaries. In fact, this is a simpler situation as we may locally use the approach presented in the previous section and do not necessarily have to enrich the respective elements.

Remark 2.3 (Approximation of the interface Γ). For the sake of clarity, we assume that the interface is known and explicitly given in this section. In addition to this, we suppose that all functions and integrals on the interface can be evaluated and computed exactly. Please note that in real-world applications, the interface is often not only implicitly given and unknown but also part of the solution, e.g., in two-phase flow or two-phase Stefan problems. Often, the level set method, cf. Section 2.1, where the zero level set of a continuous scalar function represents the location of the discontinuity is used for this purpose. For computing integrals and evaluating functions on intersected cells, an approximation Γ_h of Γ of this zero level set is constructed.

Let $V^{\text{FEM}} = \operatorname{span}\{v_j\}_{j\in\mathcal{N}}$ be a conventional finite element space on the triangulation $\mathcal{S}_h$, where $\{v_j\}_{j\in\mathcal{N}}$ with $\mathcal{N} = \{1, \ldots, N_{\mathcal{B}}\}$ is the corresponding nodal basis and $N_{\mathcal{B}}$ denotes the dimension of V^{FEM}. The fundamental idea of the eXtended finite element method is to define the index set

$$\tilde{\mathcal{N}} := \{j \in \mathcal{N} \,:\, \operatorname{meas}_{d-1}(\Gamma \cap \operatorname{supp}(v_j)) > 0\} \tag{2.18}$$

of all basis functions whose support is intersected by Γ and enrich V^{FEM} near the discontinuity Γ with additional basis functions

$$\tilde{v}_k := v_k \cdot \psi, \quad k \in \tilde{\mathcal{N}}, \tag{2.19}$$

using a so-called enrichment function ψ. With this, the extended discrete approximation space is given by[*]

$$V_h := \operatorname{span}\{v_j\}_{j\in\mathcal{N}} \oplus \operatorname{span}\{\tilde{v}_k\}_{k\in\tilde{\mathcal{N}}}. \tag{2.20}$$

Aside from the location of the discontinuity, the choice of ψ obviously depends on the discontinuous feature which has to be considered. In general, quantities can exhibit a strong discontinuity, meaning that the function itself is discontinuous, or a weak discontinuity, i.e. the gradient of a function is discontinuous. In literature, there are several enrichment functions introduced to tackle different kinds of problems, see [26, 45] for an overview. However, the most common approach to consider strong discontinuous features is based on the Heaviside function

$$\psi_{\mathrm{H}} \colon \Omega \to \mathbb{R} \quad \text{with} \quad \psi_{\mathrm{H}}(\boldsymbol{x}) := H(\boldsymbol{x}) = \begin{cases} 1, & \text{for } \boldsymbol{x} \in \Omega_2 \\ 0, & \text{else} \end{cases}, \tag{2.21}$$

while for weak discontinuous features, the *modified abs-enrichment* [51]

$$\psi_{\mathrm{abs}} \colon \Omega \to \mathbb{R} \quad \text{with} \quad \psi_{\mathrm{abs}}(\boldsymbol{x}) := \sum_{j\in\mathcal{N}} |d(\boldsymbol{x})| v_j(\boldsymbol{x}) - \Big|\sum_{j\in\mathcal{N}} d(\boldsymbol{x}) v_j(\boldsymbol{x})\Big| \tag{2.22}$$

is usually used. This enrichment is based on the distance of a point $\boldsymbol{x} \in \Omega$ to the discontinuity Γ given by a function $d \colon \Omega \to \mathbb{R}$.

Please note that although enrichment functions are usually globally defined on Ω, the introduced enrichment is still local by construction since we only enrich basis functions v_k with $k \in \tilde{\mathcal{N}}$, cf. equation (2.19).

Remark 2.4. Over the last years, the term Cut Finite Element Method (CUTFEM) [52] has become more and more popular. It summarizes methods that are based on duplicating elements that are intersected by (unfitted) boundaries or interfaces and cut-off the conventional finite element approximation space at these locations [49]. Such methods are essentially equivalent to using XFEM with a Heaviside enrichment function.

[*]Please note that for $V^{\mathrm{FEM}} = V^m_{\mathrm{cg},h}$, we will have $V_h \subset H^1(\Omega_1 \cup \Omega_2)$, but usually $V_h \not\subset H^1(\Omega)$.

N_{el}	$\inf_{v_h \in V_{\text{h}}} \|u - v_h\|_{L^2(\Omega_1 \cup \Omega_2)}$	eoc	$\inf_{v_h \in V_h} \|\nabla u - \nabla v_h\|_{L^2(\Omega_1 \cup \Omega_2)}$	eoc
2×5^2	0.004993	-	0.192204	-
2×9^2	0.001650	1.8838	0.113583	0.8949
2×17^2	0.000479	1.9447	0.062084	0.9498
2×33^2	0.000129	1.9778	0.032262	0.9869
2×65^2	0.000033	2.0111	0.016183	1.0178
2×129^2	0.000008	2.0674	0.007878	1.0503

Table 2.3 Approximation errors and estimated order of convergence for the enriched function space V_h^1 using an unfitted mesh.

Approximation quality and convergence order of the eXtended finite element method

In contrast to the finite element method, the rates of convergence of the approximation errors $\inf_{v_h \in V_{\text{h}}} \|u - v_h\|_{L^2(\Omega_1 \cup \Omega_2)}$ and $\inf_{v_h \in V_h} \|\nabla u - \nabla v_h\|_{L^2(\Omega_1 \cup \Omega_2)}$ do not depend on the location of the interface when using the eXtended finite element method[†]. Consider the example from Section 2.2.1 now using the eXtended finite element method, where the function space V_h is generated by enriching $V^1_{\text{cg},h}$. As the interesting cases are given when the mesh does not resolve the interface Γ, cf. Remark 2.2, we compute the numerical solution for $N_{\text{el}} = \{2 \times 5^2, 2 \times 9^2, 2 \times 17^2, 2 \times 33^2, 2 \times 65^2, 2 \times 129^2\}$ and specify the approximation errors and orders of convergence in Table 2.3. As expected, we get the same estimates as when using the finite element method with a fitted mesh.

2.3.3 A brief overview of numerical challenges arising in unfitted methods

The fundamental idea of unfitted methods is to use uniform meshes and consider boundaries and interfaces by adapting the discrete approximation space. This provides a very flexible and powerful method that can be applied to a wide range of problems. However, consequences that arise out of these approaches are that, firstly, the discrete formulation and analysis of problems are much more complex compared to the fitted setting, and, secondly, that the implementation of such unfitted methods is very extensive as there are a lot of additional numerical issues to take into account. At this point, we want to briefly mention the challenges when using an eXtended discretization method. Most of them are addressed and elaborated on in more detail for the hierarchical eXtended finite element method in Section 3.3 and Section 4.3.

2.3.3.1 Discrete interface representation and approximation

Leaving the difficulty of solving time-dependent problems aside, which will be addressed later and, instead, focusing on steady-state problems, a fundamental aspect to be considered concerns discrete representation and approximation of the interface. This is due to the fact that in

[†]However, the approximation of the interface does indeed play an important role, cf. Section 2.3.3.1

practice boundaries and interfaces are often only given implicitly and, moreover, run independently of the mesh boundaries. Hence, the construction of an explicit interface approximation is mandatory to define discrete subdomains.

Suppose that, for example, an interface Γ is implicitly given by the zero level of a scalar function on a triangulation that is sufficiently fine so that no element edge is intersected more than once, cf. the assumptions in [49]. The most common approach to construct an explicit interface is based on connecting the intersections points of Γ with the element edges by a linear segment. While this linear representation can be easily constructed, it obviously does not provide good approximation and convergence properties, especially in regards to problems that involve terms related to second derivatives of the interface such as curvature.

A simple but useful extension of this approach has been developed in [29, Chap. 7.3], where an additional mesh is introduced obtained by regularly refining the original triangulation. Assuming that the interface is given by a quadratic level set function, the idea is to linearly interpolate this function on the refined mesh and define the discrete interface Γ_h using planar intersection segments while all quantities involving derivatives such as computing the normal or the curvature are based on the quadratic function. While this leads to an improvement of the results, the convergence rates are still not optimal.

Another possibility to represent the interface and boundaries discretely is to use a high-order approximation of the interface such as quadratic or cubic. For this purpose, an approach subdividing intersected elements into subcells with a so-called parametric or curved facet has been developed in [53].

2.3.3.2 Subdivision and quadrature

The challenge of approximating the discrete interface is closely related to the task of performing quadrature on intersected elements. While in the early works, a high order quadrature rule with many quadrature points has often been used, which performs rather poorly, almost all recent approaches subdivide elements and perform on the resulting subelements.

The subdivision of intersected simplicial elements is especially straightforward when the intersection segment is approximated linearly. Then, one can either directly map the arising subelements, that are triangles and quadrilaterals in 2D and tetrahedrons and wedges in 3D, to the respective reference elements or use a scheme, see, e.g., [54], to further subdivide the subelements to restore simplicial elements. The latter approach is especially convenient as only the quadrature rules already required by the finite element method for non-intersected elements have to be modified.

When using high-order approximations instead of a linear representation, an interface may be arbitrarily curved inside an element so that corresponding subtriangulations and quadrature approaches have to be based on isoparametric methods using reference elements that are higher-order on only one facet. While these methods are numerically much more expensive and harder to implement, the achieved convergence rates are (close to) optimal, see [53].

2.3.3.3 Essential boundary and interface conditions

A further consequence of the fact that physical boundaries and interfaces are, in general, not resolved by the computational mesh is that a strong imposition of essential boundary and interface conditions usually results in numerical issues, see e.g. [55]. Instead of a strong imposition, it is advantageous to add these conditions to the discrete variational formulation, similar to the way Neumann conditions are included. In principle, there are three concepts available to weakly impose Dirichlet and interface conditions: the penalty method, the use of Lagrangian multipliers, and Nitsche's method.

- **Penalty method:** The idea of the penalty method, used in [46, 56], is to include boundary conditions by adding discrete approximations of the conditions to the variational formulation, which then are multiplied by a penalty parameter. While this does not introduce additional unknowns to the problem, a drawback is that the equation, and hence the linear system's condition number, scales with the penalty parameter enforcing the condition.
- **Lagrangian multipliers:** In contrast to the penalty method, using Lagrangian multipliers to impose Dirichlet boundary conditions as introduced in [57], does add additional unknowns to the problem. Moreover, the space of the multipliers has to be chosen carefully to satisfy the "inf-sup" condition [58].
- **Nitsche's method:** Nitsche's method [27] does not introduce additional unknowns and can be considered as a *consistent* penalty method. There are two versions available, a symmetric and a non-symmetric variant. Both versions of the method introduce a stabilization parameter which is necessary to ensure inf-sup stability, see Section 2.3.3.4, and which has an impact on the condition number of the discrete linear system. However, the non-symmetric variant allows setting this parameter to 1. Since its original introduction, Nitsche's method has been extended and adapted to impose all kind of boundary and interface conditions. It has been successfully used in a number of articles considering different applications, see for example [29, 49].

An overview of the mentioned approaches applied to two-phase flow with mass transportation is given in [50] and for a more general (but very brief) summary see [26] and the references therein. In this thesis, we exclusively use Nitsche's method, which will be presented in more detail in Section 3.3.2.

2.3.3.4 Stability and conditioning issues

The most difficult challenges in regards to the eXtended finite element method and related variants consist in guaranteeing inf-sup stability and avoiding ill-conditioning of the discrete problem. We briefly mention the causes of the issues and touch on how to solve them. A detailed overview can be found in [24, 50].

The inf-sup stability [59], a weaker condition as the so-called coercivity, is required in a certain norm for the discrete bilinear form to ensure existence and uniqueness of a solution of the problem. Moreover, this property can be used to derive and prove (optimal) a priori error estimates of the discrete solution. This is mandatory for numerical approaches. For eXtended discretization methods, the issue of defining discrete bilinear forms that are inf-sup stable is caused by the intersections of interface and mesh. As an intersecting part of an element may be very small, the terms for imposing interface or boundary conditions are usually not sufficiently controlled within the weak formulation. As a consequence, an inverse inequality, which is essential to prove inf-sup stability of the system, does not hold. In order to retain this property and guarantee stability uniformly, we may have to introduce additional stabilization techniques. While a stabilization technique is required for all methods that can be used to impose interface and boundary conditions mentioned in the previous section, the Nitsche method is often the method of choice. The main advantage of this approach is that it is not only consistent but also stable

- independent of the choice of the stabilization parameter (non-symmetric variant), or
- for a sufficiently large stabilization parameter (symmetric variant),

if the averaging operator introduced in [49] is used. Unfortunately, depending on the construction, the stability parameter, which also controls the error at the respective interface or boundary and in many approaches depends on the length of the intersection segment of interface and mesh, has a direct impact on the system's condition number which scales with its magnitude. Thus, preconditioning is crucial within the numerical implementation. More detailed information is given in [50].

An additional stabilization method that can be used is the so-called ghost penalty approach introduced in [60]. In this method, additional stabilization terms regarding the jump of the gradient are added to control the condition number independently of the intersection between mesh and discontinuity.

2.3.3.5 Discretization of time-dependent problems

A principal challenge of the eXtended finite element method is the discretization and solution of time-dependent problems that involve moving or evolving interfaces. A good overview of suitable methods is given in [50, Sec. 3.1.2]. In principle, there are two different techniques available, space-time formulations and the method of lines, which both have been already mentioned in Section 2.2.2.

To a certain extent, the natural approach to discretize problems involving discontinuities is to use the concept of space-time elements. By considering the time as (additional) spatial dimension, the problem is transformed into a $d+1$ dimensional stationary problem. Just as within the framework of conventional space-time finite elements, the resulting $d+1$ dimensional domain is divided into so-called time slabs [61]. Time slabs are prisms resulting from considering the domain over a time interval, and the problem is defined in variational formulation for these time slabs that are decoupled by applying a discontinuous Galerkin method. While the literature suggests that the numerical costs are more or less comparable to time stepping methods, using this concept within eXtended discretization methods is rather complicated and challenging as we need to implement a subdivision scheme for intersected elements. An extensive introduction and description of space-time methods for the eXtended finite element method is given in [50].

When using the method of lines, there is a fundamental difference between conventional and eXtended finite element methods, as pointed out in [62]. Within conventional methods, the function space relies on the triangulation and there is an identification of mesh entities and basis functions with the corresponding degrees of freedom. Therefore, the basis functions can be considered as time independent. While this is obvious for methods using a fixed mesh, it is also true for moving mesh approaches like the ALE method which considers evolutions in time by introducing a mesh velocity and uses a parametrization of the function spaces. Thereby, all quantities are evaluated on the same geometry. In contrast to this, enriched basis functions of eXtended approximation spaces are time-dependent by construction since the enrichment functions consider the movement of the discontinuity. Consequently, we can only apply Rothe's method to problems when using XFEM as we need to firstly discretize in time and then in space. Therefore, it is important to point out that since the fundamental idea of the method of lines is to use a finite difference approximation of the time derivative, this quantity has to be sufficiently smooth with respect to time. While this is not necessarily the case in problems involving discontinuities, most of the problems can be scaled to retain a continuous time derivative. However, in any case no high-order approximations can be used, if the interface moves.

2.3.3.6 Miscellaneous

Last but not least there are some practical issues when using the eXtended finite element method. Since an enriched function does not generally fulfill the *Kronecker-delta property* meaning that for $u_h \in V_h$, $u_h(x_j) \neq u_j$, $j \in \mathcal{N}$, the implementation of nodal interpolation onto the enriched approximation space V_h is not straight forward. An alternative is to use the

L^2-projection instead. Another aspect is that routines used for visualizing computed results have to be extended to correctly represent enriched elements containing jumps and kinks.

Chapter 3

A hierarchical eXtended finite element method for multiphysics problems

Any modeling and discretization approach for multiphysics problems has to be able to consider arbitrary geometries defined by multiple interfaces at which different discontinuous features are present. While an approach based on the eXtended finite element method in combination with multiple level set functions is very flexible and independent from the problem's geometry, we need to resolve two main issues that concern, the domain decomposition and the enrichment scheme. The level set method as described in Section 2.1, using one indicator function, only allows for the decomposition of a domain into two subdomains. Hence, more level set functions are needed to separate a given domain into several subdomains and represent the boundaries by zero level sets. When doing this, complex situations such as multi-junctions can arise and additional effort is necessary to ensure that the resulting method is geometrically consistent, that is, no voids or overlapping domains arise. In order to sharply represent discontinuous features on arbitrary evolving subdomains, any introduced enrichment scheme has to be robust so that the generated eXtended approximation spaces and the discretization yields stable systems of equations, even if basis functions and the corresponding degrees of freedom have to be enriched by multiple discontinuities.

Inspired by [25], we present a robust and flexible approach that is based on introducing a *hierarchical order* to the problem. Therefore, a hold-all domain is decomposed by multiple level set functions that are *hierarchically ordered*. Using this order, which can be based on modeling assumptions or can be artificially introduced, the problem is reformulated and the approximation spaces are extended by using Heaviside enrichment for each discontinuity. The hierarchical level set method is a flexible domain decomposition scheme that is geometrically consistent and can represent all possible domain changes. Based on this decomposition, the Heaviside enrichment scheme provides a robust extension of the approximation space and allows to accurately consider arbitrary evolving discontinuities. We begin this chapter by introducing the general problem setting by

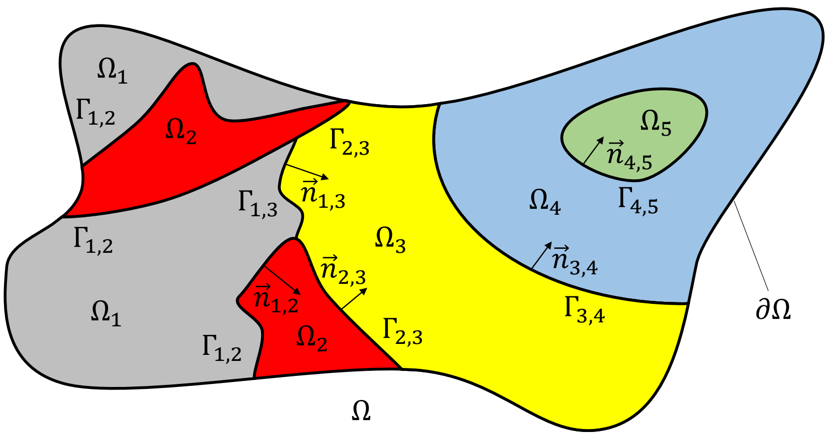


Figure 3.1 Exemplary setting as described in Definition 3.1.

Definition 3.1 (General domain setting)**.** Let $\Omega \subset \mathbb{R}^d$ be a physical domain with polygonal boundary $\partial\Omega$, consisting of N_{dom} pairwise disjoint subdomains $\Omega_i(t)$, for $t \in [t_0, t_f]$, $i = 1, \ldots, N_{\text{dom}}$, that are separated by sharp and sufficiently smooth internal boundaries called *interfaces* $\Gamma_{i,l}(t) := \text{interior}_{d-1}\left(\bar{\Omega}_i \cap \bar{\Omega}_l\right)$, with $i \neq l$. At the interfaces $\Gamma_{i,l}$, the outwards-pointing unit normal vectors are given by $\vec{n}_{i,l}(t) = \vec{n}_i(t) = -\vec{n}_l(t)$ and the outward-pointing normal vector to $\partial\Omega$ is denoted by $\vec{n}$, with $\vec{n}|_{\partial\Omega\cap\partial\Omega_i} = \vec{n}_i$.

A sketch of the setting described in Definition 3.1 for $N_{\text{dom}} = 5$ domains is given in Figure 3.1. We introduce the some notation in

Definition 3.2 (Operators for functions featuring a discontinuity)**.** For a quantity $u\colon \Omega \to \mathbb{R}$ which is sufficiently smooth on Ω_i, $i = 1, \ldots, N_{\text{dom}}$, the notation $u|_{\Omega_i}$ in general denotes the restriction of u onto the subdomain Ω_i. At an interface $\Gamma_{i,l}$, $i \neq l$, this notation is short for $(u|_{\Omega_i})_{\Gamma_{i,l}}$ and denotes (in trace sense) the limiting value of a quantity given in Ω_i when approaching an interface $\Gamma_{i,l}$, $i \neq l$.

In general, all quantities involved in a (multiphysics) problem may be discontinuous or have a discontinuous derivative at any interface $\Gamma_{i,l}$. Hence, we introduce some operators in

Definition 3.3 (Operators for functions featuring a discontinuity)**.** For a scalar quantity $u\colon \Omega \to \mathbb{R}$, which is sufficiently smooth on Ω_i, $i = 1, \ldots, N_{\text{dom}}$, but can feature a discontinuity across an interface $\Gamma_{i,l}$, we define for $\boldsymbol{x} \in \Gamma_{i,l}$, for all $i < l$,

- $[\![u(\boldsymbol{x})]\!] = (u|_{\Omega_i})_{\Gamma_{i,l}} - (u|_{\Omega_l})_{\Gamma_{i,l}}$ (jump operator),
- $\{u(\boldsymbol{x})\} = w_i\,(u|_{\Omega_i})_{\Gamma_{i,l}} + w_l\,(u|_{\Omega_l})_{\Gamma_{i,l}}$ (weighted average operator), and
- $\langle u(\boldsymbol{x})\rangle = w_l\,(u|_{\Omega_i})_{\Gamma_{i,l}} + w_i\,(u|_{\Omega_l})_{\Gamma_{i,l}}$ (cross-over weighted average operator),

with $0 \leq w_i$, $w_l \leq 1$ and $w_i + w_l = 1$. The *weights* w_i, w_l can be chosen arbitrarily, however, they impact the stability of a discrete system, see [50, Sec. 2]. In regard to the jump of a product, we will make use of the relation

$$[\![uv]\!] = [\![v]\!]\{u\} + \langle v\rangle [\![u]\!], \tag{3.1}$$

which can be easily verified. Please note that although we defined the operators and relation for scalar quantities, their definition can also be used for vector-valued quantities simply by component-wise consideration.

Before we present the hierarchical eXtended finite element method, we want to point out the complexity arising when multiple subdomains are involved in the problem which are separated by interfaces that feature different conditions. For this purpose, we consider an enhanced version of the Example 2.1.

3.1 Analytical example involving multiple subdomains

Following Definition 3.1 with $N_{\text{dom}} = 3$, let $\Omega \subset \mathbb{R}^d$ be a fixed domain that is polygonally bounded with $\partial\Omega = \Gamma_{\text{D}} \cup \Gamma_{\text{N}}$ and consists of the disjoint subdomains Ω_1, Ω_2, and Ω_3 separated by sharp and sufficiently smooth interfaces $\Gamma_{1,2}$, $\Gamma_{1,3}$, and $\Gamma_{2,3}$ with outward-pointing normal vectors $\vec{n}_{1,2}$, $\vec{n}_{1,3}$, and $\vec{n}_{2,3}$. Now consider the problem

Example 3.1 (Steady-state heat equation)**.** *For given data which is sufficiently smooth, find* $u\colon \Omega \to \mathbb{R}$ *s.t. it solves the problem given by*

$$\begin{aligned}
\xi u - \nabla\cdot(\kappa\nabla u) &= f && \text{in } \Omega_1 \cup \Omega_2 \cup \Omega_3, && (3.2)\\
u &= g_{\text{D}} && \text{on } \Gamma_{\text{D}}, && (3.3)\\
-\kappa\nabla u \cdot \vec{n} &= g_{\text{N}} && \text{on } \Gamma_{\text{N}}, && (3.4)\\
[\![\kappa\nabla u]\!]\cdot \vec{n}_{1,2} &= g_{1,2} && \text{on } \Gamma_{1,2}, && (3.5)\\
[\![u]\!] &= q_{1,2} && \text{on } \Gamma_{1,2}, && (3.6)\\
[\![\kappa\nabla u]\!]\cdot \vec{n}_{1,3} &= g_{1,3} && \text{on } \Gamma_{1,3}, && (3.7)\\
[\![u]\!] &= q_{1,3} && \text{on } \Gamma_{1,3}, && (3.8)\\
u|_{\Omega_2} &= g_2 && \text{on } \Gamma_{2,3}, && (3.9)\\
u|_{\Omega_3} &= g_3 && \text{on } \Gamma_{2,3} && (3.10)
\end{aligned}$$

for a situation similar to Figure 3.2.

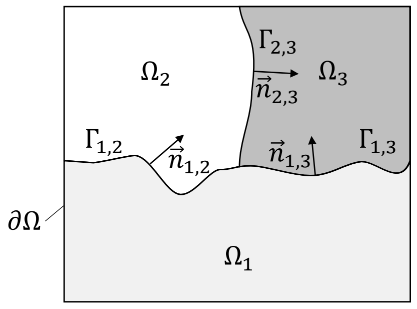


Figure 3.2 Exemplary setting for Example 3.1.

Variational formulation and weak solution

In the following, we want to define a weak formulation and show that there exists a unique solution to the problem at hand. We start by introducing

$$U := \{v \in H^1(\Omega_1 \cup \Omega_2 \cup \Omega_3)\ : v = g_\mathrm{D} \text{ on } \Gamma_\mathrm{D},\ v|_{\Omega_2} = g_2 \text{ on } \Gamma_{2,3},\ v|_{\Omega_3} = g_3 \text{ on } \Gamma_{2,3}, \\ [\![v]\!] = q_{1,2} \text{ on } \Gamma_{1,2},\ [\![v]\!] = q_{1,3} \text{ on } \Gamma_{1,3}\,\} \tag{3.11}$$

and

$$V := \{v \in H^1(\Omega_1 \cup \Omega_2 \cup \Omega_3)\ : v = 0 \text{ on } \Gamma_\mathrm{D},\ v = 0 \text{ on } \Gamma_{2,3}, \\ [\![v]\!] = 0 \text{ on } \Gamma_{1,2},\ [\![v]\!] = 0 \text{ on } \Gamma_{1,3}\,\}. \tag{3.12}$$

The weak formulation of Example 3.1 is then given by: Find $u \in U$ such that for all $v \in V$ the integral identity

$$\int_{\Omega_1\cup\Omega_2\cup\Omega_3} (\xi uv + \kappa\nabla u \cdot \nabla v)\,\mathrm{d}x = \int_{\Omega_1\cup\Omega_2\cup\Omega_3} fv\,\mathrm{d}x - \int_{\Gamma_\mathrm{N}} g_\mathrm{N} v\,\mathrm{d}x \\ + \int_{\Gamma_{1,2}} g_{1,2} v\,\mathrm{d}x + \int_{\Gamma_{1,3}} g_{1,3} v\,\mathrm{d}x \tag{3.13}$$

holds, where we assume that $0 \le \xi \in \mathbb{R}$, $f \in L^2(\Omega_1 \cup \Omega_2 \cup \Omega_3)$, $g_\mathrm{N} \in L^2(\Gamma_N)$, $g_{1,2} \in L^2(\Gamma_{1,2})$, $g_{1,3} \in L^2(\Gamma_{1,3})$, and $\kappa \in L^\infty(\Omega)$, with $0 < a < \kappa$, $a \in \mathbb{R}$.

Any function $u \in U$ satisfying equation (3.13) for all $v \in V$ is called weak solution of Example 3.1. Unfortunately, U is not a Hilbert space and $U \neq V$, so we can not use the integral identity to test with the solution itself to gain important estimates and apply the theorem of Lax-Milgram.

Hence, we first introduce the following (weak) auxiliary problem for $\tilde{u}$, $v \in V$

$$\int_{\Omega_1\cup\Omega_2\cup\Omega_3} \xi\tilde{u}v\,\mathrm{d}x + \kappa\nabla\tilde{u}\cdot\nabla v\,\mathrm{d}x = -\int_{\Omega_1\cup\Omega_2\cup\Omega_3} (\xi wv + \kappa\nabla w \cdot \nabla v)\,\mathrm{d}x \\ + \int_{\Omega_1\cup\Omega_2\cup\Omega_3} fv\,\mathrm{d}x - \int_{\Gamma_\mathrm{N}} g_\mathrm{N} v\,\mathrm{d}x \\ + \int_{\Gamma_{1,2}} g_{1,2} v\,\mathrm{d}x + \int_{\Gamma_{1,3}} g_{1,3} v\,\mathrm{d}x, \tag{3.14}$$

where we assume that it is $w \in H^1(\Omega_1 \cup \Omega_2 \cup \Omega_3)$ and that all other functions have the same regularity as before. With this assumptions, the theorem of Lax-Milgram can be applied to equation (3.14) saying that there is a unique solution $\tilde{u} \in V$ of equation (3.14). This motivates us to tackle Example 3.1 as we only have to find a function $w \in H^1(\Omega_1 \cup \Omega_2 \cup \Omega_3)$ so that the conditions given by equation (3.3), equation (3.6), equation (3.8), equation (3.9), and equation (3.10) hold.

Assuming $\overline{\Gamma_{i,l}} \cap \Gamma_{\mathrm{D}} = \emptyset$ for $1 \leq i < l \leq 3$, $g_{\mathrm{D}} \in H^{\frac{1}{2}}(\Gamma_{\mathrm{D}})$, $g_2, g_3 \in H^{\frac{1}{2}}(\Gamma_{2,3})$, $q_{1,2} \in H^{\frac{1}{2}}(\Gamma_{1,2})$, and $q_{1,3} \in H^{\frac{1}{2}}(\Gamma_{1,3})$, we introduce the extensions

$$\hat{g}_{\mathrm{D}} \colon \partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3 \to \mathbb{R} \quad \text{with} \quad \hat{g}_{\mathrm{D}} := \begin{cases} g_{\mathrm{D}}, & \text{on } \Gamma_{\mathrm{D}} \\ 0, & \text{else} \end{cases}, \tag{3.15}$$

$$\hat{g}_2 \colon \partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3 \to \mathbb{R} \quad \text{with} \quad \hat{g}_2 := \begin{cases} g_2, & \text{on } \Gamma_{2,3} \\ 0, & \text{else} \end{cases}, \tag{3.16}$$

$$\hat{g}_3 \colon \partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3 \to \mathbb{R} \quad \text{with} \quad \hat{g}_3 := \begin{cases} g_3, & \text{on } \Gamma_{2,3} \\ 0, & \text{else} \end{cases}, \tag{3.17}$$

$$\hat{q}_{1,2} \colon \partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3 \to \mathbb{R} \quad \text{with} \quad \hat{q}_{1,2} := \begin{cases} q_{1,2}, & \text{on } \Gamma_{1,2} \\ 0, & \text{else} \end{cases}, \tag{3.18}$$

$$\hat{q}_{1,3} \colon \partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3 \to \mathbb{R} \quad \text{with} \quad \hat{q}_{1,3} := \begin{cases} q_{1,3}, & \text{on } \Gamma_{1,3} \\ 0, & \text{else} \end{cases}, \tag{3.19}$$

and require

$$w_1 := (\hat{g}_{\mathrm{D}}) \in H^{\frac{1}{2}}\left(\mathrm{interior}_{d-1}\left(\bar{\Gamma}_{1,2} \cup \bar{\Gamma}_{1,3} \cup \left(\partial\Omega_1 \cap \bar{\Gamma}_D\right)\right)\right), \tag{3.20}$$

$$w_2 := \left(\hat{g}_{\mathrm{D}} - \hat{q}_{1,2} + \hat{g}_2\right) \in H^{\frac{1}{2}}\left(\mathrm{interior}_{d-1}\left(\bar{\Gamma}_{1,2} \cup \bar{\Gamma}_{2,3} \cup \left(\partial\Omega_2 \cap \bar{\Gamma}_D\right)\right)\right), \tag{3.21}$$

$$w_3 := \left(\hat{g}_{\mathrm{D}} - \hat{q}_{1,3} + \hat{g}_3\right) \in H^{\frac{1}{2}}\left(\mathrm{interior}_{d-1}\left(\bar{\Gamma}_{1,3} \cup \bar{\Gamma}_{2,3} \cup \left(\partial\Omega_3 \cap \bar{\Gamma}_D\right)\right)\right), \tag{3.22}$$

so that there exists extensions $\hat{w}_1 \in H^1(\Omega_1)$, $\hat{w}_2 \in H^1(\Omega_2)$, and $\hat{w}_3 \in H^1(\Omega_3)$, see [63]. Introducing their trivial extensions $\tilde{w}_1, \tilde{w}_2, \tilde{w}_3 \in H^1(\Omega_1 \cup \Omega_2 \cup \Omega_3)$, we now can define the function $w := \tilde{w}_1 + \tilde{w}_2 + \tilde{w}_3 \in H^1(\Omega_1 \cup \Omega_2 \cup \Omega_3)$.

Remark 3.4. The assumptions in equations (3.20) to (3.22) for the extensions defined in equations (3.15) to (3.19) concern the compatibility at the boundaries. In regards to applications, guaranteeing the compatibility of boundaries can be challenging, especially when multiple-junctions arise. We will address this aspect later.

Let $\tilde{u} \in V$ be the weak solution of equation (3.14), we define $u := \tilde{u} + w$ and see that (in the sense of traces)

$$u|_{\Gamma_\mathrm{D}} = \tilde{u}|_{\Gamma_\mathrm{D}} + w|_{\Gamma_\mathrm{D}} = 0 + \tilde{g}_\mathrm{D}|_{\Gamma_\mathrm{D}} = g_\mathrm{D}, \tag{3.23}$$

$$[\![u]\!]|_{\Gamma_{1,2}} = [\![\tilde{u} + w]\!]|_{\Gamma_{1,2}} = [\![\tilde{u}]\!]|_{\Gamma_{1,2}} + [\![w]\!]|_{\Gamma_{1,2}} = 0 - (-\tilde{q}_{1,2}|_{\Gamma_{1,2}}) = q_{1,2}, \tag{3.24}$$

$$[\![u]\!]|_{\Gamma_{1,3}} = [\![\tilde{u} + w]\!]|_{\Gamma_{1,3}} = [\![\tilde{u}]\!]|_{\Gamma_{1,3}} + [\![w]\!]|_{\Gamma_{1,3}} = 0 - (-\tilde{q}_{1,3}|_{\Gamma_{1,3}}) = q_{1,3}, \tag{3.25}$$

$$u_2|_{\Gamma_{2,3}} = \tilde{u}_2|_{\Gamma_{2,3}} + w_2|_{\Gamma_{2,3}} = \tilde{g}_2|_{\Gamma_{2,3}} = g_2, \tag{3.26}$$

$$u_3|_{\Gamma_{2,3}} = \tilde{u}_2|_{\Gamma_{2,3}} + w_3|_{\Gamma_{2,3}} = \tilde{g}_3|_{\Gamma_{2,3}} = g_3, \tag{3.27}$$

Therefore, $u \in U$ and u obviously solves equation (3.13) so that we can apply Lax-Milgram which gives us the existence of a unique solution of Example 3.1 under the above assumptions.

Remark 3.5. If we drop the assumption that $\Gamma_{i,l} \cap \Gamma_\mathrm{D} = \emptyset$ for $1 \leq i < l \leq 3$, some modifications concerning the jumps $q_{1,2}$ and $q_{1,3}$ and, hence, the functions w_1, w_2, and w_3 are necessary: First of all we introduce $\gamma_1, \gamma_2 \in H^{\frac{1}{2}}(\Gamma_{1,2})$ with $\gamma_1 + \gamma_2 = q_{1,2}$ and $\gamma_3, \gamma_4 \in H^{\frac{1}{2}}(\Gamma_{1,3})$ with $\gamma_3 + \gamma_4 = q_{1,3}$ and define the extensions $\hat{\gamma}_1, \hat{\gamma}_2, \hat{\gamma}_3, \hat{\gamma}_4 \colon \partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3 \to \mathbb{R}$ similar as in (3.15) to (3.19). Instead of conditions (3.20) to (3.22) we now request

$$w_1 := (\hat{g}_\mathrm{D} + \hat{\gamma}_1 + \hat{\gamma}_3) \in H^{\frac{1}{2}}\left(\mathrm{interior}_{d-1}\left(\bar{\Gamma}_{1,2} \cup \bar{\Gamma}_{1,3} \cup \left(\partial\Omega_1 \cap \bar{\Gamma}_D\right)\right)\right), \tag{3.28}$$

$$w_2 := (\hat{g}_\mathrm{D} - \hat{\gamma}_2 + \hat{g}_2) \in H^{\frac{1}{2}}\left(\mathrm{interior}_{d-1}\left(\bar{\Gamma}_{1,2} \cup \bar{\Gamma}_{2,3} \cup \left(\partial\Omega_2 \cap \bar{\Gamma}_D\right)\right)\right), \tag{3.29}$$

$$w_3 := (\hat{g}_\mathrm{D} - \hat{\gamma}_4 + \hat{g}_3) \in H^{\frac{1}{2}}\left(\mathrm{interior}_{d-1}\left(\bar{\Gamma}_{1,3} \cup \bar{\Gamma}_{2,3} \cup \left(\partial\Omega_3 \cap \bar{\Gamma}_D\right)\right)\right), \tag{3.30}$$

and choose the corresponding extensions that can be used to construct $u := \tilde{u} + w$ such that $u \in U$ satisfies equation (3.13).

3.2 Domain decomposition using multiple level set functions

When considering analytical or numerical problems, the interfaces and boundaries $\Gamma_{i,l}$ are usually represented by zero levels of (explicitly given) indicator functions or by an explicit parametrization. The level set method briefly presented in Section 2.1 is the most common approach to describe discontinuities and their evolution in time. However, as one level set function can only separate a given domain into two subdomains, multiple level set functions are required to describe settings involving more than two subdomains.

3.2.1 Brief overview of methods based on multiple level set functions

The most straightforward approach to describe multiphase situations using the level set method is to introduce one indicator function for each phase. One of the first works considering triple-junctions is [64], where all level set functions are considered separately and a reassignment step is performed afterwards to regain geometric consistency. This work is further extended in [65] in which the interface's motion depends on surface tension and bulk energies. In [66], a similar approach based on the same early work is presented. However, all methods still include model restrictions to prevent overlapping or voids between the domains.

While there have been various approaches proposed in the meantime, to the best of the author's knowledge, the issue of getting a geometrically consistent method is still usually considered by a post-processing step after moving the level set function. Instead of addressing this aspect, the approaches usually focus on the numerical efficiency. As the oldest approaches use one level set function to describe each domain individually, a lot of redundancies are introduced. In order to decrease the redundancies, some articles such as [67] introduces approaches where level set functions are used to identify the domains they separate. Thereby, the number of level set functions can be significantly reduced. While the proposed method is not immune to overlapping domains or the generation of voids, it does decreases the number of critical points.

In contrast to the mentioned articles, [25] proposes a simple but effective strategy to avoid the aforementioned problems by introducing a hierarchy among the indicator functions. Inspired by this work, we present a hierarchical level set method that allows to decompose any domain and automatically retains the geometric consistency when the subdomains evolve in time.

Remark 3.6 (Defining complex domains using multiple level set functions). The level set method is an often used approach to describe and construct domains that are (mostly) independent of the underlying computational mesh. In regards to single-phase problems, multiple level set functions in combination with boolean operations can also be used to define very complex domains, see [24, Appendix A] for examples. Therefore, it is widely used within the field of *constructive solid geometry* (CSG). In this context, a first level set function, usually referred to as *master function* defines the fundamental domain while *slave functions* remove parts of the domain defined by the master function. An exmaple of this technique is used in [68] to automatically generate high-order meshes for curved surfaces.

3.2.2 Domain decomposition by using a hierarchical level set method

We start this paragraph with a short motivation to emphasize the method's idea and its generality:

Motivation 3.1. *[Domain decomposition with hierarchical level set functions.] Consider a melting process of some workpiece in a gas atmosphere where we, for illustration purposes, assume that all quantities and boundaries are known for all $t \in [t_0, t_f]$. Given a hold-all domain Ω that consists of three subdomains, the surrounding gas atmosphere $\Omega_1(t)$, the solid material $\Omega_2(t)$, and the molten material $\Omega_3(t)$, the idea of the hierarchical level set method is to formulate the problem specifically for the given scenario to provide a characterization of the subdomains $\Omega_i(t)$ that is geometrically consistent also with respect to their evolution in time. Therefore,* *the domain setting and the physics are used to introduce a hierarchy and describe the problem as follows: Given a hold-all domain Ω and the workpiece geometry $\Gamma_1(t)$, we first define the surrounding gas atmosphere $\Omega_1(t)$ by introducing an indicator function $\varphi_1 \colon \Omega \to \mathbb{R}$ (a signed distance function to the work piece geometry) whose zero level set $\Gamma_1(t)$ (the work piece boundary) separates Ω into $\Omega_1(t)$ and $\Omega_1^{\mathsf{c}}(t)$, so that $\Omega = \Omega_1(t) \cup \Omega_1^{\mathsf{c}}(t) \cup \Gamma_1(t)$. Now, we introduce a second indicator function $\varphi_2 \colon \Omega \to \mathbb{R}$ which, restricted to $\Omega_1^{\mathsf{c}}(t)$, corresponds to a signed distance function to the melting temperature u_m whose zero level set $\Gamma_2(t)$ (the solid-liquid interface) separates $\Omega_1^{\mathsf{c}}(t)$ into a solid part $\Omega_2(t)$ and the remaining domain $\Omega_2^{\mathsf{c}}(t)$, which in this case is the molten material $\Omega_3(t)$. In total, we have $\Omega = \Omega_1(t) \cup \Gamma_1(t) \cup \Omega_2(t) \cup \Gamma_2(t) \cup \Omega_3(t)$ and the evolution of all subdomains is coupled. Since the zero level set $\Gamma_2(t)$ of the second indicator function $\varphi_2(\cdot, t)$ has an effect only on the domain $\Omega_1^{\mathsf{c}}(t)$*[*]*, where $\Omega_1(t)$ is defined by $\varphi_1(\boldsymbol{x}, t)$ respectively $\Gamma_1(t)$, we say that φ_1 is of higher hierarchy than φ_2.*

The presented idea can be generalized and used to decompose any domain into subdomains. Recall the situation as depicted in Definition 3.1, that is we have a hold-all domain $\Omega \subset \mathbb{R}^d$ with sufficiently smooth boundary consisting of up to N_{dom} pairwise disjoint subdomains $\Omega_i(t)$, $i = 1, \dots, N_{\text{dom}}$, for $t \in [t_0, t_f]$, which are separated by sharp and sufficiently smooth internal boundaries $\Gamma_{i,\tilde{l}}(t) = \text{interior}_{d-1}\left(\overline{\Omega_i(t)} \cap \overline{\Omega_{\tilde{l}}(t)}\right)$, with $i < \tilde{l}$.

We can describe this scenario by introducing multiple level set functions that are hierarchically ordered with the following constructive approach: For $i = 1, \dots, N_{\text{dom}} - 1$, let $\varphi_i \in C^1\left(\Omega \times (t_0, t_f)\right) \cap C^0(\bar{\Omega} \times [t_0, t_f])$ be such that the following conditions are fulfilled:

(Cond. 1) $\varphi_i|_{\Omega_i} < 0$,

(Cond. 2) $\varphi_i|_{\Omega_{\tilde{l}}} > 0$ for all $\tilde{l} > i$, and

(Cond. 3) $\varphi_i|_{\Gamma_i} = 0$ with $\Gamma_i(t) := \bigcup_{\tilde{l} > i} \Gamma_{i,\tilde{l}}(t)$.

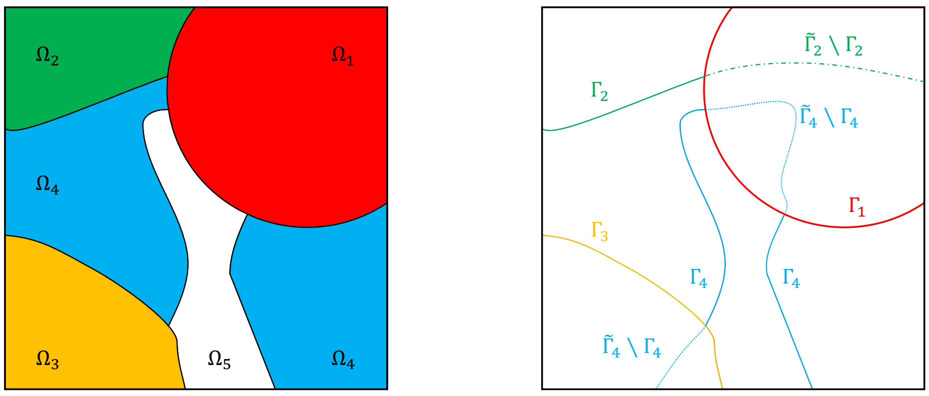
The domains $\Omega_i(t)$ then can be reconstructed by

$$\begin{aligned}
\Omega_i(t) &:= \left\{ \boldsymbol{x} \in \Omega : \varphi_i(\boldsymbol{x}, t) H(-\varphi_i(\boldsymbol{x}, t)) \prod_{l=1}^{i-1} H(\varphi_l(\boldsymbol{x}, t)) < 0 \right\}, \quad i = 1, \dots, N_{\text{dom}} - 1, \\
\Omega_{N_{\text{dom}}}(t) &:= \left\{ \boldsymbol{x} \in \Omega : \prod_{l=1}^{N_{\text{dom}}-1} H(\varphi_l(\boldsymbol{x}, t)) = 1 \right\},
\end{aligned} \tag{3.31}$$

with

$$H(\varphi_i(\boldsymbol{x}, t)) := \begin{cases} 1, & \text{for } \varphi_i(\boldsymbol{x}, t) > 0 \\ 0, & \text{else} \end{cases}, \quad i = 1, \dots, N_{\text{dom}} - 1, \tag{3.32}$$

[*]The level set function $\varphi_2(\cdot, t)$, which is related to the temperature $u(\cdot, t)$, is of course also defined on $\Omega_1(t)$. However, the melting temperature u_m corresponding to the melting temperature of the workpiece material does not cause phase transitions in the gas atmosphere. Hence, $u|_{\Omega_1}$ does not have a discontinuous feature at $\Gamma_2(t)$.


Ω_2
Ω_1
Ω_4
Ω_3
Ω_5
Ω_4
$\tilde{\Gamma}_2 \setminus \Gamma_2$
Γ_2
$\tilde{\Gamma}_4 \setminus \Gamma_4$
Γ_1
Γ_3
Γ_4
Γ_4
$\tilde{\Gamma}_4 \setminus \Gamma_4$

(a) Setting.

(b) Domain decomposition using hierarchical level set functions.

Figure 3.3 Domain decomposition using hierarchical level set functions.

denoting the Heaviside function with respect to φ_i and $\prod_{l\in\emptyset} H(\varphi_l(\boldsymbol{x},t)) = 1$ as the *empty product.*

We call the domains and level set functions *hierarchically ordered* and say that for $l < i$ the function φ_l is of *higher* or *upper* hierarchy than φ_i, since by this construction, φ_i has no influence on quantities in the domain $\bigcup_{l<i} \Omega_l$ but only on quantities in the remaining part. This is because for $\boldsymbol{x} \in \bigcup_{l<i} \Omega_l$, there is one index $k \in \{1,\dots,i-1\}$ with $\boldsymbol{x} \in \Omega_k$. Due to (Cond. 1), this implies $\varphi_k(\boldsymbol{x}) < 0$ as well as $H(\varphi_k(\boldsymbol{x})) = 0$ and, hence, $\prod_{l=1}^{i-1} H(\varphi_l(\boldsymbol{x},t)) = 0$. By contrast, we have $\prod_{l=1}^{i-1} H(\varphi_l(\boldsymbol{x},t)) = 1$ for $\boldsymbol{x} \in \left(\bigcup_{l<i} \Omega_l\right)^{\mathsf{c}}$.

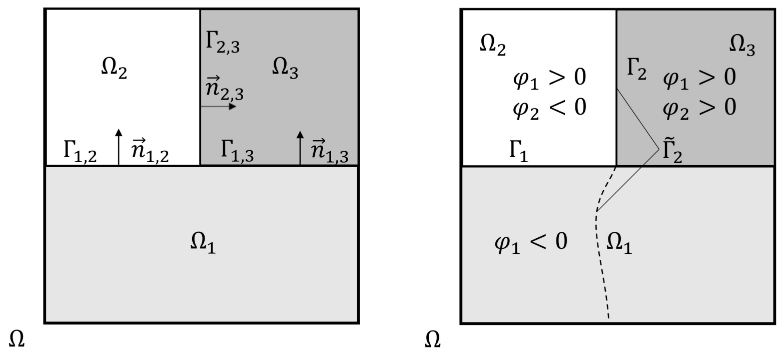
Since there are no requirements for φ_i on $\bigcup_{l<i} \Omega_l$ other than the regularity $\varphi_i \in C^1(\Omega \times (t_0,t_f)) \cap C^0(\bar{\Omega} \times [t_0,t_f])$, it is not necessarily $\Gamma_i = \tilde{\Gamma}_i$, where $\tilde{\Gamma}_i$ denotes the zero level of φ_i, but only $\Gamma_i \subseteq \tilde{\Gamma}_i$. Therefore, we use the same idea as in equation (3.31) and characterize Γ_i for $i = 1,\dots,N_{\text{dom}} - 1$[†] by

$$\begin{aligned}\Gamma_i(t) &:= \left\{ \boldsymbol{x} \in \Omega : \varphi_i(\boldsymbol{x},t) = 0 \wedge \prod_{l=1}^{i-1} H(\varphi_l(\boldsymbol{x},t)) > 0 \right\} \\ &= \left\{ \boldsymbol{x} \in \tilde{\Gamma}_i : \prod_{l=1}^{i-1} H(\varphi_l(\boldsymbol{x},t)) > 0 \right\}.\end{aligned} \tag{3.33}$$

Please note that this approach provides us with a straightforward strategy to create any required decomposition that is sufficiently smooth of a given hold-all domain Ω into subdomains Ω_i, cf. Figure 3.3. This is because we only have to introduce a (potentially artificial) hierarchy of domains and boundaries appropriate to the given problem and separate the domains using the presented scheme.

Example 3.2 (Creating domains using the hierarchical level set method)**.** *Consider the domain* $\Omega = [0,1]^2 \subset \mathbb{R}^2$ *with* $\Gamma_D = (0,1) \times \{0\}$ *and* $\Gamma_{\mathrm{N}} = \partial\Omega \setminus \Gamma_{\mathrm{D}}$ *that consists of* $\Omega_1 = (0,1) \times (0,0.5)$,

[†]As one interface separates two domains, we have one interfaces less than there are domains.



(a) Setting in Example 3.2.

(b) Domain decomposition using hierarchical level set functions.

Figure 3.4 Setting for Example 3.2 and domain decomposition using hierarchical level set functions.

$\Omega_2 = (0, 0.5) \times (0.5, 1)$ *and* $\Omega_3 = (0.5, 1) \times (0.5, 1)$ *separated by the interfaces* $\Gamma_{1,2} = (0, 0.5) \times \{0.5\}$*,* $\Gamma_{1,3} = (0.5, 1) \times \{0.5\}$*, and* $\Gamma_{2,3} = \{0.5\} \times (0.5, 1)$*.*

We can use the hierarchical level set method to describe this setting, which may describe the initial configuration of a problem involving evolving domains, by introducing $\varphi_1(\boldsymbol{x}) = y - 0.5$ *with* $\Gamma_1 = (0,1)\times\{0.5\} = \tilde{\Gamma}_1$ *and* $\varphi_2(\boldsymbol{x}) = x - 0.5$ *with* $\Gamma_2 = \{0.5\}\times(0.5, 1) \subset \tilde{\Gamma}_2 = \{0.5\}\times(0, 1)$*. Both level set functions clearly satisfy the conditions (Cond. 1) to (Cond. 3) and equations* (3.31) *and* (3.33)*. This setting is illustrated in Figure 3.4.*

Remark 3.7 (A priori knowledge for time dependent problems with evolving subdomains)*.* The presented, constructive approach is obviously based on an a priori study of the problem at hand in regards to how many subdomains can evolve, the causes of their evolution, and how the subdomains are related. As this information is already necessary to model the process, no additional knowledge is required and all problems can be formulated using this idea. Also note that although this a priori knowledge simplifies the matter significantly, we could additionally include a method to rearrange the hierarchies on the fly within the implementation of this approach. However, this is numerically expensive as all data has to be rearranged, too.

Remark 3.8 (Reformulation of problems to fit into the hierarchical setting)*.* (Almost) all PDE problems involving different domains and/or discontinuities can be reformulated to introduce a hierarchical order into the model. Usually, this hierarchical order is motivated straight-forward by physical phenomena but one can also introduce an artificial hierarchy. An example for problems where the order is physically motivated are applications involving different states of matter that are defined by temperature levels. In contrast to this, applications considering material's grain growth can be handled by introducing an arbitrary hierarchy among the different grains.

Remark 3.9 (A comment on the method's consistency in regards to the geometry and physics)*.* The presented hierarchical level set method is geometrically consistent by construction. However, this property comes at a certain price: By defining multiple interfaces by the same zero level set of an indicator function, see (Cond. 3), the approach automatically smoothens the conditions or velocity fields, which may result from different physical phenomena, at the merging points of interfaces. However, this is not a specific drawback of the proposed method but a more general issue since, when modeling multiple junctions, the problem usually tends to have more conditions than degrees of freedom at this point. Therefore, numerical approaches usually have to choose one condition while neglecting others, see e.g. [13].

3.3 A hierarchical eXtended finite element method

Using the introduced hierarchical representation of the subdomains, boundaries and interfaces, we now address the enrichment of discrete approximation spaces. Usually, multiphysics problems involve several domains and therefore discontinuities so that an appropriate discrete approximation space has to be multiple-enriched using different enrichment functions ψ_i, $i = 1, \dots, N_{\text{dom}} - 1$ that are, in particular, based on the discontinuities' location. Since the problem can in general involve strong and weak discontinuous features, we could either

(1) allow for completely different enrichments for weak and strong discontinuities, e.g. the *mod-abs* enrichment for the first and *Heaviside* enrichment for the latter, or

(2) use an enrichment scheme primarily developed for considering strong discontinuities and include additional conditions to enforce continuity at the corresponding interfaces.

As we are interested in developing a general approach to numerically compute solutions of multiphysics problems using the concept of automated code generation, we choose the second approach and introduce the hierarchical Heaviside enrichment which is combined with Nitsche's method to capture all arising discontinuous features. In the following, we use the setting and notation as in Section 2.3.2; that is, we have a shape regular mesh $\mathcal{S}_{h>0}$ with $\bigcup_{S \in \mathcal{S}_h} = \bar{\Omega}$ and a corresponding (standard) finite element space V^{FEM} with basis $\{v_j\}_{j \in \mathcal{N}}$ appropriate for the given problem. In addition, we assume that the triangulation is sufficiently fine so that any discontinuity does not intersect a mesh edge more than once.

For the sake of a simplified presentation, we start by considering the enrichment for steady-state situations. In Section 3.3.3, we comment on how to handle problems that are time-dependent and involve moving or evolving interfaces.

3.3.1 Hierarchical Heaviside enrichment

Let φ_i, $i = 1, \dots, N_{\text{dom}} - 1$, be hierarchically ordered level set functions with zero level sets $\tilde{\Gamma}_i$ as introduced in Section 3.2.2 separating a domain Ω into N_{dom} disjoint subdomains Ω_i. We assume that all $\tilde{\Gamma}_i$ are explicitly given, cf. Remark 2.3, to neglect the issue of constructing a discrete approximation. The discontinuities are represented by Γ_i which are each characterized as in equation (3.33).

For the construction of a suitable discrete approximation space, we define the hierarchical Heaviside-type enrichment functions $\psi_i\colon \Omega \to \{0,1\}$ by

$$\psi_i(\boldsymbol{x}) := \prod_{l=1}^{i} H(\varphi_l(\boldsymbol{x})), \quad i = 1, \dots, N_{\mathrm{dom}} - 1 \tag{3.34}$$

and introduce the sets

$$\mathcal{N}_i := \{j \in \mathcal{N} \,:\, \mathrm{meas}_{d-1}(\Gamma_i \cap \mathrm{supp}(v_j)) > 0\}, \quad i = 1, \dots, N_{\mathrm{dom}} - 1, \tag{3.35}$$

containing the indices of all basis functions $\{v_j\}_{j\in\mathcal{N}}$ of V^{FEM} whose support is intersected by Γ_i. Then, the basis functions that are added for considering the i-th discontinuity while respecting all upper hierarchy levels are constructed by

$$v_{i,k} := \psi_i \cdot v_k, \quad k \in \mathcal{N}_i. \tag{3.36}$$

and the resulting discrete approximation space is given by

$$V_h := \underbrace{\mathrm{span}\{v_j\}_{j\in\mathcal{N}}}_{=\,V^{\mathrm{FEM}}} \bigoplus_{i=1,\dots,N_{\mathrm{dom}}-1} \underbrace{\mathrm{span}\{v_{i,k}\}_{k\in\mathcal{N}_i}}_{=\,V^{\Gamma_i}}, \tag{3.37}$$

where all elements $u_h \in V_h$ have the structure

$$u_h = \sum_{j\in\mathcal{N}} u_j v_j + \sum_{i=1}^{N_{\mathrm{dom}}-1} \left(\sum_{k\in\mathcal{N}_i} u_{i,k} v_{i,k} \right) = \boldsymbol{u}_h \cdot \boldsymbol{v}_h, \tag{3.38}$$

with

$$\boldsymbol{u}_h = [\underbrace{u_1, \dots, u_{|\mathcal{N}|}}_{\text{std. coefficients}}, \underbrace{u_{1,1}, \dots, u_{1,|\mathcal{N}_1|}}_{\text{coefficients for } \Gamma_1}, \dots, \underbrace{u_{N_{\mathrm{dom}}-1,1}, \dots, u_{N_{\mathrm{dom}}-1,|\mathcal{N}_{N_{\mathrm{dom}}-1}|}}_{\text{coefficients for } \Gamma_{N_{\mathrm{dom}}-1}}]^T \tag{3.39}$$

and

$$\boldsymbol{v}_h = [\underbrace{v_1, \dots, v_{|\mathcal{N}|}}_{\text{std basis functions}}, \underbrace{v_{1,1}, \dots, v_{1,|\mathcal{N}_1|}}_{\text{basis functions for } \Gamma_1}, \dots, \underbrace{v_{N_{\mathrm{dom}}-1,1}, \dots, v_{N_{\mathrm{dom}}-1,|\mathcal{N}_{N_{\mathrm{dom}}-1}|}}_{\text{basis functions for } \Gamma_{N_{\mathrm{dom}}-1}}]^T, \tag{3.40}$$

where $\mathcal{N}_i$ denotes the respective set of enriched basis functions and $|\mathcal{N}_i|$ its cardinality. Hence, the enriched basis functions and the corresponding coefficients are just as hierarchically ordered as the level set functions. In regards to the matrix A that arises when a problem given by $a(u_h, v_h) = L(v_h)$ is written as linear system $A \cdot u = b$, the hierarchical order results in a block scheme

$$A = \begin{pmatrix} A_{\text{std}\times\text{std}} & A_{\Gamma_1\times\text{std}} & A_{\Gamma_2\times\text{std}} & \cdots & \cdots & A_{\Gamma_{N_{\text{dom}}-1}\times\text{std}} \\ A_{\text{std}\times\Gamma_1} & A_{\Gamma_1\times\Gamma_1} & A_{\Gamma_2\times\Gamma_1} & \cdots & \cdots & A_{\Gamma_{N_{\text{dom}}-1}\times\Gamma_1} \\ \vdots & \vdots & \vdots & \cdots & \cdots & \vdots \\ A_{\text{std}\times\Gamma_{N_{\text{dom}}-1}} & A_{\Gamma_1\times\Gamma_{N_{\text{dom}}-1}} & A_{\Gamma_2\times\Gamma_{N_{\text{dom}}-1}} & \cdots & \cdots & A_{\Gamma_{N_{\text{dom}}-1}\times\Gamma_{N_{\text{dom}}-1}} \end{pmatrix} \tag{3.41}$$

with

$$\begin{aligned} A_{\text{std}\times\text{std}} &= \Big(a(v_j, v_l)\Big)_{j,l\in\mathcal{N}}, \\ A_{\Gamma_i\times\text{std}} &= \Big(a(v_{i,k}, v_j)\Big)_{j\in\mathcal{N},\, k\in\mathcal{N}_i,\, i\in\{1,\ldots,N_{\text{dom}}-1\}}, \\ A_{\text{std}\times\Gamma_i} &= \Big(a(v_j, v_{i,k})\Big)_{j\in\mathcal{N},\, k\in\mathcal{N}_i,\, i\in\{1,\ldots,N_{\text{dom}}-1\}}, \\ A_{\Gamma_i\times\Gamma_l} &= \Big(a(v_{i,k}, v_{l,\tilde{k}})\Big)_{k\in\mathcal{N}_i,\, \tilde{k}\in\mathcal{N}_l,\, i,l\in\{1,\ldots,N_{\text{dom}}-1\}}. \end{aligned} \tag{3.42}$$

Due to the hierarchical construction, every level set function introduces its own enrichment via equation (3.34). This is also advantageous in regards to the linear independency of the discrete system with respect to enriched basis functions v_l with $\text{meas}_{d-1}(\Gamma_i \cap \text{supp}(v_l)) < \epsilon \ll |S|$, $S \in \mathcal{S}_h$, see Section 4.3.

Example 3.3 (Representation of a discontinuous function)**.** *On the domain* $\Omega = [0,1]^2 \subset \mathbb{R}^2$ *subdivided by the hierarchical level set functions* $\varphi_1(\boldsymbol{x}) = y - 0.5$ *and* $\varphi_2(\boldsymbol{x}) = x - 0.5$ *into* $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Gamma_{1,2} \cup \Gamma_{1,3} \cup \Gamma_{2,3}$*, we want to represent the piecewise constant function*

$$\kappa(\boldsymbol{x}) = \kappa_i \quad \text{on } \Omega_i, \quad i = 1,2,3, \tag{3.43}$$

see Figure 3.5(a), using the hierarchical eXtended finite element method. While the obvious choice of the approximation space which has to be enriched would be $V^0_{\text{dg},h}$*, which is defined in* (2.10)*, we choose* $V^{\text{FEM}} = V^1_{\text{cg},h}$*, cf* (2.9)*, instead for illustration purposes. Hence, it is*

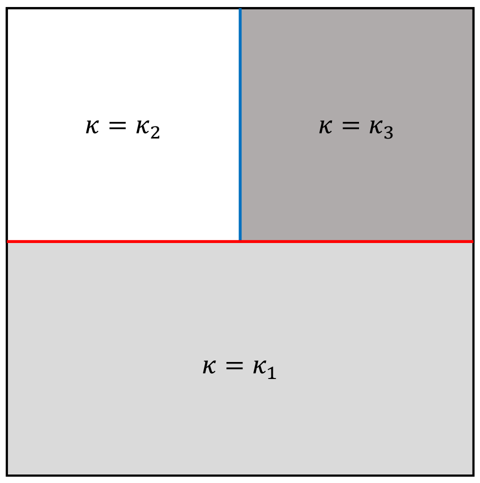
$$V_h = \{v_h \in C^0(\Omega_i) : v_h|_{S\cap\Omega_i} \in \mathcal{P}^1(S \cap \Omega_i),\ \forall S \in \mathcal{S}_h,\ i = 1,2,3\}. \tag{3.44}$$

Choosing a uniform triangulation $\mathcal{S}_h$ *consisting of* 2×5^2 *elements, the index set of the standard basis functions* $\{v_j\}_{j\in\mathcal{N}}$ *is given by*

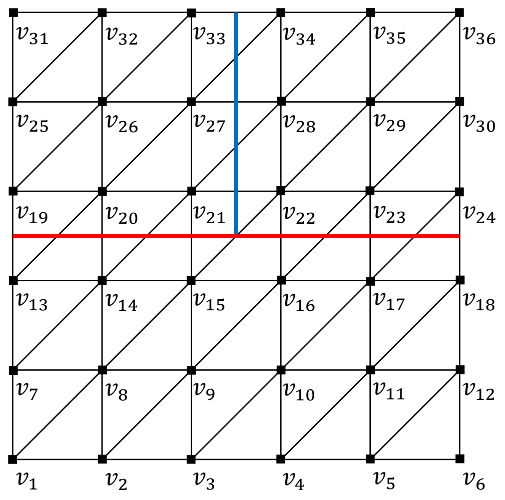
$$\mathcal{N} = \{1, \ldots, 36\}, \tag{3.45}$$

cf. Figure 3.5(b), the index set of basis functions enriched by Γ_1 *is given by*

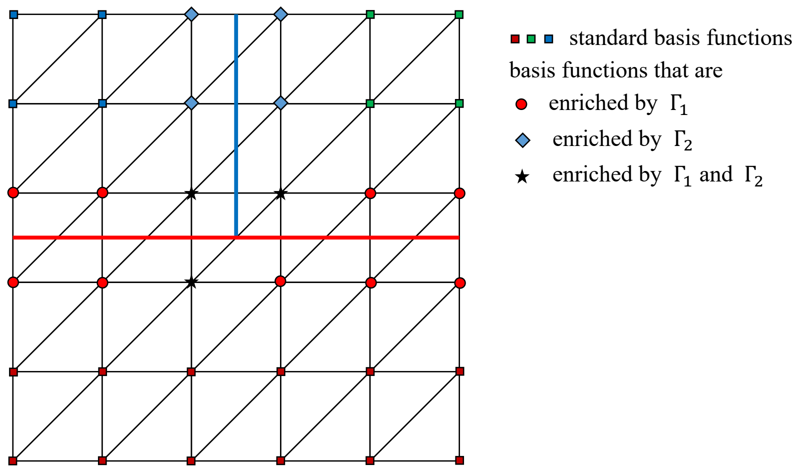
$$\mathcal{N}_1 = \{13, \ldots, 24\}, \tag{3.46}$$



(a) Setting in Example 3.3.



(b) Standard basis functions (numbered).



(c) Enrichment status of basis functions.

Figure 3.5 Setting in Example 3.3 including the numbering and the enrichment status of the basis functions.

and the index set of basis functions enriched by Γ_2 *is given by*

$$\mathcal{N}_2 = \{15, 21, 22, 27, 28, 33, 34\}, \tag{3.47}$$

see Figure 3.5(c) for a visualization. Using the representation of (3.39), the values of the coefficients are

$$u_j = \begin{cases} \kappa_1 & \text{for } j \in \{1, \dots, 24\} \\ \kappa_2 & \text{for } j \in \{25, \dots, 28, 31, \dots, 34\}, \\ \kappa_3 & \text{for } j \in \{29, 30, 35, 36\}, \end{cases} , \tag{3.48}$$

$$u_{1,k} = \begin{cases} \kappa_2 - \kappa_1 & \text{for } k \in \{13, \dots, 18, 19, \dots, 22\} \\ \kappa_3 - \kappa_1 & \text{for } k \in \{23, 24\}, \end{cases} \quad , \tag{3.49}$$

and

$$u_{2,k} = \kappa_3 - \kappa_2 \quad \text{for } k \in \mathcal{N}_2. \tag{3.50}$$

3.3.2 Imposing interface and boundary conditions using Nitsche's method

As discussed in Section 2.3.3.3, there are three methods available to impose boundary and interface conditions when using eXtended discretization methods: the Nitsche method, the use of Lagrangian multipliers, and the penalty approach. In this thesis, we exclusively use Nitsche's method [27], which will be derived following [49, 50, 69] for the hierarchical eXtended finite element method.

Let $\Omega \subset \mathbb{R}^d$ be a polygonally bounded domain with $\partial\Omega = \Gamma_\mathrm{D} \cup \Gamma_\mathrm{N}$, which is separated into N_dom disjoint subdomains Ω_i by $\Gamma_i \subseteq \tilde{\Gamma}_i$, which in turn are the zero level sets of hierarchically ordered level set functions φ_i, $i = 1, \dots, N_\mathrm{dom} - 1$. We assume that the mesh is fitted to $\partial\Omega$ and choose $\nabla \cdot (\kappa \nabla u)$ as a representative term for deriving Nitsche's method since terms and conditions related to interfaces and boundaries arise due to integration by parts.

For $u_h, v_h \in V_h$ and $\kappa \in \tilde{V}_h$, with $V_h, \tilde{V}_h$ denoting extended approximation spaces which are based on a hierarchical enrichment of given spaces $V^\mathrm{FEM}, \tilde{V}^\mathrm{FEM} \subset H^1(\Omega)$ that are not necessarily the same, we have, via integration by parts and reordering,

$$\begin{aligned}
&\sum_{i=1}^{N_\mathrm{dom}} \left(- \int_{\Omega_i} \nabla \cdot \left(\kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \right) v_h|_{\Omega_i} \,\mathrm{d}x \right) \\
&= \sum_{i=1}^{N_\mathrm{dom}} \int_{\Omega_i} \kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \nabla v_h|_{\Omega_i} \,\mathrm{d}x - \sum_{i=1}^{N_\mathrm{dom}} \int_{\partial\Omega_i} \kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_i v_h|_{\Omega_i} \,\mathrm{d}x \\
&= \sum_{i=1}^{N_\mathrm{dom}} \int_{\Omega_i} \kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \nabla v_h|_{\Omega_i} \,\mathrm{d}x \\
&\quad - \sum_{i=1}^{N_\mathrm{dom}} \int_{\partial\Omega_i \cap \Gamma_\mathrm{D}} \kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_i v_h|_{\Omega_i} \,\mathrm{d}x - \sum_{i=1}^{N_\mathrm{dom}} \int_{\partial\Omega_i \cap \Gamma_\mathrm{N}} \kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_i v_h|_{\Omega_i} \,\mathrm{d}x \\
&\quad - \sum_{i=1}^{N_\mathrm{dom}-1} \sum_{l=i+1}^{N_\mathrm{dom}} \int_{\partial\Omega_i \cap \partial\Omega_l} \kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_{i,l} v_h|_{\Omega_i} \,\mathrm{d}x \\
&\quad - \sum_{i=1}^{N_\mathrm{dom}-1} \sum_{l=i+1}^{N_\mathrm{dom}} \int_{\partial\Omega_i \cap \partial\Omega_l} \kappa|_{\Omega_l} \nabla u_h|_{\Omega_l} \cdot \vec{n}_{l,i} v_h|_{\Omega_l} \,\mathrm{d}x.
\end{aligned} \tag{3.51}$$

For the presentation of Nitsche's method, we focus on interface conditions, which are more general than boundary conditions, see Remark 3.10. According to Definition 3.1, we have

$\Gamma_{i,l} = \text{interior}_{d-1}\left(\bar{\Omega}_i \cap \bar{\Omega}_l\right) = \text{interior}_{d-1}\left(\partial\Omega_i \cap \partial\Omega_l\right)$ with $\vec{n}_{i,l} = \vec{n}_i = -\vec{n}_l$. Hence, we write

$$\begin{aligned}
&- \sum_{i=1}^{N_{\text{dom}}-1} \sum_{l=i+1}^{N_{\text{dom}}} \int_{\partial\Omega_i\cap\partial\Omega_l} \kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_{i,l} v_h|_{\Omega_i} \,\mathrm{d}x \\
&- \sum_{i=1}^{N_{\text{dom}}-1} \sum_{l=i+1}^{N_{\text{dom}}} \int_{\partial\Omega_i\cap\partial\Omega_l} \kappa|_{\Omega_l} \nabla u_h|_{\Omega_l} \cdot \vec{n}_{l,i} v_h|_{\Omega_l} \,\mathrm{d}x \\
&= - \sum_{i=1}^{N_{\text{dom}}-1} \sum_{l=i+1}^{N_{\text{dom}}} \left(\int_{\partial\Omega_i\cap\partial\Omega_l} \left(\kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_{i,l} v_h|_{\Omega_i} + \kappa|_{\Omega_l} \nabla u_h|_{\Omega_l} \cdot \vec{n}_{l,i} v_h|_{\Omega_l} \right) \mathrm{d}x \right) \\
&= - \sum_{i=1}^{N_{\text{dom}}-1} \sum_{l=i+1}^{N_{\text{dom}}} \left(\int_{\partial\Omega_i\cap\partial\Omega_l} \left(\kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_{i,l} v_h|_{\Omega_i} - \kappa|_{\Omega_l} \nabla u_h|_{\Omega_l} \cdot \vec{n}_{i,l} v_h|_{\Omega_l} \right) \mathrm{d}x \right) \\
&= - \sum_{i=1}^{N_{\text{dom}}-1} \sum_{l=i+1}^{N_{\text{dom}}} \left(\int_{\Gamma_{i,l}} \left(\kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_i v_h|_{\Omega_i} - \kappa|_{\Omega_l} \nabla u_h|_{\Omega_l} \cdot \vec{n}_i v_h|_{\Omega_l} \right) \mathrm{d}x \right).
\end{aligned} \tag{3.52}$$

Within the hierarchical level set method used for the decomposition of the domain into subdomains, it is $\Gamma_{i,l} \subset \Gamma_i$. Thus, we write $\Gamma_{i,l} = \Gamma_{i,l} \cap \Gamma_i$ so that

$$\begin{aligned}
&- \sum_{i=1}^{N_{\text{dom}}-1} \sum_{l=i+1}^{N_{\text{dom}}} \left(\int_{\Gamma_{i,l}} \left(\kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_i v_h|_{\Omega_i} - \kappa|_{\Omega_l} \nabla u_h|_{\Omega_l} \cdot \vec{n}_i v_h|_{\Omega_l} \right) \mathrm{d}x \right) \\
&\quad = - \sum_{i=1}^{N_{\text{dom}}-1} \sum_{l=i+1}^{N_{\text{dom}}} \left(\int_{\Gamma_i\cap\Gamma_{i,l}} \left(\kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_i v_h|_{\Omega_i} - \kappa|_{\Omega_l} \nabla u_h|_{\Omega_l} \cdot \vec{n}_i v_h|_{\Omega_l} \right) \mathrm{d}x \right)
\end{aligned} \tag{3.53}$$

On each interface part $\Gamma_i \cap \Gamma_{i,l} \neq \emptyset$[‡], $i = 1, \ldots, N_{\text{dom}} - 1$ and $l > i$, we can enforce either *jump conditions*

$$[\![\kappa \nabla u]\!] \cdot \vec{n}_i = g_{i,l}, \quad [\![u]\!] = q_{i,l}, \tag{3.54}$$

or *Dirichlet conditions*

$$u|_{\Omega_i} = g_i, \quad u|_{\Omega_l} = g_l, \tag{3.55}$$

with functions $g_{i,l}$, $q_{i,l}$, and g_i, g_l that are sufficiently smooth.

Remark 3.10 (Imposing boundary conditions with Nitsche's method). Imposing Dirichlet conditions

$$u = g_{\mathrm{D}} \quad \text{on } \Gamma_{\mathrm{D}} \tag{3.56}$$

at a boundary Γ_{D} instead of an interface is obviously a special case of the conditions (3.55) since we only have to consider one boundary.

[‡]Please note that it is by definition $\Gamma_i = \bigcup_{l>i} \Gamma_{i,l}$ and,hence, $\Gamma_i \cap \Gamma_{i,l} = \emptyset$ means that $\partial\Omega_i \cap \partial\Omega_l = \emptyset$ so that the integration domain is empty.

Jump conditions: For imposing jump conditions on an interface $\Gamma_{i_0} \cap \Gamma_{i_0,l_0}$, with $i_0 \in \{1,\dots,N_{\text{dom}}-1\}$ and $l_0 \in \{i_0+1,\dots,N_{\text{dom}}\}$, we firstly summarize the corresponding terms by

$$\begin{aligned} -\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \left(\kappa|_{\Omega_{i_0}} \nabla u_h|_{\Omega_{i_0}} \cdot \vec{n}_{i_0} v_h|_{\Omega_{i_0}} - \kappa|_{\Omega_{l_0}} \nabla u_h|_{\Omega_{l_0}} \cdot \vec{n}_{i_0} v_h|_{\Omega_{l_0}}\right) \mathrm{d}x \\ = -\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} [\![\kappa \nabla u_h \cdot \vec{n}_{i_0} v_h]\!] \,\mathrm{d}x \end{aligned} \tag{3.57}$$

using the jump operator $[\![\cdot]\!]$ defined in Definition 3.3. Since $v \in V_h$ may be discontinuous across all interfaces, it is $[\![v]\!] \neq 0$ in general. Hence, we cannot extract v from the jump and include the respective condition but have to use the relation introduced in equation (3.1). Then we have

$$\begin{aligned} -\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} [\![\kappa \nabla u_h \cdot \vec{n}_{i_0} v_h]\!] \,\mathrm{d}x = & -\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} [\![v_h]\!]\{\kappa \nabla u_h \cdot \vec{n}_{i_0}\} \,\mathrm{d}x \\ & -\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \langle v_h\rangle [\![\kappa \nabla u_h \cdot \vec{n}_{i_0}]\!] \,\mathrm{d}x \end{aligned} \tag{3.58}$$

and the flux condition (3.54) (left) can now be considered by

$$-\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \langle v_h\rangle [\![\kappa \nabla u_h \cdot \vec{n}_{i_0}]\!] \,\mathrm{d}x = -\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \langle v_h\rangle g_{i_0,l_0} \,\mathrm{d}x. \tag{3.59}$$

In order to control the jump $[\![u_h]\!]$ across Γ_{i_0}, see (3.54) (right), we add the terms in form of an equation

$$\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \frac{\lambda_{i_0,l_0}}{h} [\![u_h]\!][\![v_h]\!] \,\mathrm{d}x - \int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \frac{\lambda_{i_0,l_0}}{h} q_{i_0,l_0} [\![v_h]\!] \,\mathrm{d}x = 0, \tag{3.60}$$

which stabilize the bilinear form and establish coercivity, where h denotes the mesh size and $\lambda_{i_0,l_0} \in \mathbb{R}$ the so-called stabilization parameter. In total, we then have

$$\begin{aligned} -\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} [\![\kappa \nabla u_h \cdot \vec{n}_{i_0} v_h]\!] \,\mathrm{d}x = & -\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} [\![v_h]\!]\{\kappa \nabla u_h \cdot \vec{n}_{i_0}\} \,\mathrm{d}x + \int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \frac{\lambda_{i_0,l_0}}{h} [\![u_h]\!][\![v_h]\!] \,\mathrm{d}x \\ & -\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \langle v_h\rangle g_{i_0,l_0} \,\mathrm{d}x - \int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \frac{\lambda_{i_0,l_0}}{h} q_{i_0,l_0} [\![v_h]\!] \,\mathrm{d}x, \end{aligned} \tag{3.61}$$

where the last two terms are shifted to the right-hand-side.

The definition of the operators $\{\cdot\}$ and $\langle\cdot\rangle$ depends on weights w_{i_0} an w_{l_0} with $w_{i_0} + w_{l_0} = 1$, cf. Definition 3.3. Inspired by [49], we set them as

$$w_{i_0} = \frac{|S \cap \Omega_{i_0}|}{|S \cap (\Omega_{i_0} \cup \Omega_{l_0})|} \qquad \text{respectively} \qquad w_{l_0} = \frac{|S \cap \Omega_{l_0}|}{|S \cap (\Omega_{i_0} \cup \Omega_{l_0})|} \tag{3.62}$$

so that for an element $S \in \mathcal{S}_h$ intersected by Γ_{i_0,l_0}, they correspond to the relative subvolume parts of S lying in the respective subdomain. By considering the intersecting volume $|S \cap (\Omega_{i_0} \cup \Omega_{l_0})|$ we take into account that an element can be intersected multiple times and, hence, may contain a multiple junction.

Dirichlet conditions: When imposing Dirichlet conditions, the respective interface conditions on each domain are considered separately. Hence, no equivalent relation to equation (3.1) is necessary. Instead, in order to control the boundary values, we only add the terms in form of equations

$$\begin{aligned}
&\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \frac{\lambda_{i_0,l_0}}{h}\, u_h|_{\Omega_{i_0}} v_h|_{\Omega_{i_0}}\,\mathrm{d}x - \int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \frac{\lambda_{i_0,l_0}}{h} g_{i_0} v_h|_{\Omega_{i_0}}\,\mathrm{d}x = 0, \quad \text{and}\\
&\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \frac{\lambda_{i_0,l_0}}{h}\, u_h|_{\Omega_{l_0}} v_h|_{\Omega_{l_0}}\,\mathrm{d}x - \int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \frac{\lambda_{i_0,l_0}}{h} g_{l_0} v_h|_{\Omega_{l_0}}\,\mathrm{d}x = 0
\end{aligned} \tag{3.63}$$

to

$$-\int_{\Gamma_{i_0}\cap\Gamma_{i_0,l_0}} \left(\kappa|_{\Omega_{i_0}} \nabla u_h|_{\Omega_{i_0}} \cdot \vec{n}_{i_0} v_h|_{\Omega_{i_0}} - \kappa|_{\Omega_{l_0}} \nabla u_h|_{\Omega_{l_0}} \cdot \vec{n}_{i_0} v_h|_{\Omega_{l_0}}\right)\mathrm{d}x. \tag{3.64}$$

Again linear terms in equation (3.63) that contain the boundary values g_{i_0} and g_{l_0} are shifted to the right-hand-side.

Symmetric vs. non-symmetric Nitsche method

Regardless of the type of interface conditions, the resulting system is, in contrast to the original continuous formulation, obviously not symmetric, even if we shift the linear terms containing the imposed conditions to the right-hand-side of a problem. Since symmetry is advantageous from a numerical point of view, we introduce additional terms to make the problem symmetric again.

For all interfaces $\Gamma_i \cap \Gamma_{i,l}$ on which jump conditions are enforced, we add the auxiliary zero terms

$$-\int_{\Gamma_i\cap\Gamma_{i,l}} [\![u_h]\!]\{\kappa\nabla v_h \cdot \vec{n}_i\}\,\mathrm{d}x + \int_{\Gamma_i\cap\Gamma_{i,l}} q_{i,l}\{\kappa\nabla v_h \cdot \vec{n}_i\}\,\mathrm{d}x = 0 \tag{3.65}$$

so that we end up with

$$\begin{aligned}
-\int_{\Gamma_i\cap\Gamma_{i,l}} [\![\kappa\nabla u_h \cdot \vec{n}_i v_h]\!]\,\mathrm{d}x \;=\;& \\
-\int_{\Gamma_i\cap\Gamma_{i,l}} [\![v_h]\!]\{\kappa\nabla u_h \cdot \vec{n}_i\}\,\mathrm{d}x &- \int_{\Gamma_i\cap\Gamma_{i,l}} [\![u_h]\!]\{\kappa\nabla v_h \cdot \vec{n}_i\}\,\mathrm{d}x + \int_{\Gamma_i\cap\Gamma_{i,l}} \frac{\lambda_{i,l}}{h}\, [\![u_h]\!][\![v_h]\!]\,\mathrm{d}x \\
-\int_{\Gamma_i\cap\Gamma_{i,l}} \langle v_h\rangle g_{i,l}\,\mathrm{d}x &+ \int_{\Gamma_i\cap\Gamma_{i,l}} q_{i,l}\{\kappa\nabla v_h \cdot \vec{n}_i\}\,\mathrm{d}x - \int_{\Gamma_i\cap\Gamma_{i,l}} \frac{\lambda_{i,l}}{h}\, q_{i,l}[\![v_h]\!]\,\mathrm{d}x,
\end{aligned} \tag{3.66}$$

and, if Dirichlet conditions have to be imposed, we add

$$\begin{aligned}
&-\int_{\Gamma_i\cap\Gamma_{i,l}} \kappa|_{\Omega_i} \nabla v_h|_{\Omega_i} \cdot \vec{n}_i u_h|_{\Omega_i}\,\mathrm{d}x + \int_{\Gamma_i\cap\Gamma_{i,l}} \kappa|_{\Omega_i} \nabla v_h|_{\Omega_i} \cdot \vec{n}_i g_i\,\mathrm{d}x = 0,\\
&+\int_{\Gamma_i\cap\Gamma_{i,l}} \kappa|_{\Omega_l} \nabla v_h|_{\Omega_l} \cdot \vec{n}_i u_h|_{\Omega_l}\,\mathrm{d}x - \int_{\Gamma_i\cap\Gamma_{i,l}} \kappa|_{\Omega_l} \nabla v_h|_{\Omega_l} \cdot \vec{n}_i g_l\,\mathrm{d}x = 0
\end{aligned} \tag{3.67}$$

providing us with

$$
\begin{aligned}
&-\int_{\Gamma_i\cap\Gamma_{i,l}}(\kappa|_{\Omega_i}\nabla u_h|_{\Omega_i}\cdot\vec{n}_i v_h|_{\Omega_i}-\kappa|_{\Omega_l}\nabla u_h|_{\Omega_l}\cdot\vec{n}_i v_h|_{\Omega_l})\,\mathrm{d}x\\
&=-\int_{\Gamma_i\cap\Gamma_{i,l}}\kappa|_{\Omega_i}\nabla u_h|_{\Omega_i}\cdot\vec{n}_i v_h|_{\Omega_i}\,\mathrm{d}x-\int_{\Gamma_i\cap\Gamma_{i,l}}\kappa|_{\Omega_i}\nabla v_h|_{\Omega_i}\cdot\vec{n}_i u_h|_{\Omega_i}\,\mathrm{d}x+\int_{\Gamma_i\cap\Gamma_{i,l}}\frac{\lambda_{i,l}}{h}u_h|_{\Omega_i}v_h|_{\Omega_i}\,\mathrm{d}x\\
&\quad+\int_{\Gamma_i\cap\Gamma_{i,l}}\kappa|_{\Omega_l}\nabla u_h|_{\Omega_l}\cdot\vec{n}_i v_h|_{\Omega_l}\,\mathrm{d}x+\int_{\Gamma_i\cap\Gamma_{i,l}}\kappa|_{\Omega_l}\nabla v_h|_{\Omega_l}\cdot\vec{n}_i u_h|_{\Omega_l}\,\mathrm{d}x+\int_{\Gamma_i\cap\Gamma_{i,l}}\frac{\lambda_{i,l}}{h}u_h|_{\Omega_l}v_h|_{\Omega_l}\,\mathrm{d}x\\
&\quad+\int_{\Gamma_i\cap\Gamma_{i,l}}\kappa|_{\Omega_i}\nabla v_h|_{\Omega_i}\cdot\vec{n}_i g_i\,\mathrm{d}x-\int_{\Gamma_i\cap\Gamma_{i,l}}\kappa|_{\Omega_l}\nabla v_h|_{\Omega_l}\cdot\vec{n}_i g_l\,\mathrm{d}x\\
&\quad-\int_{\Gamma_i\cap\Gamma_{i,l}}\frac{\lambda_{i,l}}{h}g_i v_h|_{\Omega_i}\,\mathrm{d}x-\int_{\Gamma_i\cap\Gamma_{i,l}}\frac{\lambda_{i,l}}{h}g_l v_h|_{\Omega_l}\,\mathrm{d}x.
\end{aligned}
\tag{3.68}
$$

In both situations, the linear terms, which are the last three terms in equation (3.66) and the last four terms in equation (3.68), have been shifted to the right-hand-side.

Unfortunately, the stabilization parameters $\lambda_{i,l}$ have to be chosen large enough as mentioned in various articles, see e.g. [24, 49, 50, 69]. In fact, the magnitude of λ depends on the polynomial degree of the approximation space and the shape regularity both of which impacts the so-called *inverse trace inequality* used in the analysis of problems formulated using this method. In contrast to this, we can choose $\lambda_{i,l} = 1$ when using the non-symmetric Nitsche variant or even drop the penalty term completely by choosing $\lambda_{i,l} = 0$, see [70]. The resulting bilinear and linear form are denoted by a_h and L_h.

Remark 3.11 (Analysis of Nitsche's method and stabilization techniques)*.* By decoupling computational mesh and physical domain boundaries and thereby, shifting the approximation issue from the mesh to the discrete function space, the challenge for guaranteeing inf-sup stability of the discrete problem arises, cf. Section 2.3.3.4. While this issue has to be addressed for each problem individually, we will briefly point out the main aspects that have to be addressed. Extensive presentations of these aspects are presented for specific problems in [24, 49, 50]. A method which overcomes not only the challenges arising due to the imposition of boundary conditions with Nitsche's method but also several numerical issues such as the ill-conditioning of discrete systems is the *ghost penalty* stabilization [60, 70, 71]. The fundamental idea of the ghost penalty approach is to introduce an additional stabilization term to the problem and, thereby, allowing for the controlling of the normal derivative on the interface by the H^1-semi-norm instead of an energy norm. Moreover, the stabilization term is independent of the intersection of interface and element.

To illustrate the derivation of the terms arising due to Nitsche's method more clearly, we consider the following example:

Example 3.4 (Symmetric variant of Nitsche's method applied to Example 3.1)**.** *Consider the example that is extensively discussed in Section 3.1: Assuming that the mesh is fitted to $\partial\Omega$, we introduce two hierarchically ordered, sufficiently smooth level set functions $\varphi_1, \varphi_2\colon \Omega \to \mathbb{R}$ with* *zero level sets* $\Gamma_1 = \Gamma_{1,2} \cup \Gamma_{1,3}$ *and* $\Gamma_2 = \Gamma_{2,3}$ *so that the problem in new notation reads: On* $\mathbb{R}^d \supset \Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Gamma_1 \cup \Gamma_2$ *with* $\partial\Omega = \Gamma_\mathrm{D} \cup \Gamma_\mathrm{N}$ *solve*

$$\xi u - \nabla \cdot (\kappa \nabla u) = f \qquad \text{in } \Omega_1 \cup \Omega_2 \cup \Omega_3, \tag{3.69}$$

$$u = g_\mathrm{D} \qquad \text{on } \Gamma_\mathrm{D}, \tag{3.70}$$

$$-\kappa \nabla u \cdot \vec{n} = g_\mathrm{N} \qquad \text{on } \Gamma_\mathrm{N}, \tag{3.71}$$

$$[\![\kappa \nabla u]\!] \cdot \vec{n}_1 = g_1 \qquad \text{on } \Gamma_1, \tag{3.72}$$

$$[\![u]\!] = q_1 \qquad \text{on } \Gamma_1, \tag{3.73}$$

$$u|_{\Omega_2} = g_2 \qquad \text{on } \Gamma_2, \tag{3.74}$$

$$u|_{\Omega_3} = g_3 \qquad \text{on } \Gamma_2 \tag{3.75}$$

with

$$g_1 = \begin{cases} g_{1,2} & \textit{on } \Gamma_{1,2} \\ g_{1,3} & \textit{on } \Gamma_{1,3} \end{cases}, \quad \text{and} \quad q_1 = \begin{cases} q_{1,2} & \textit{on } \Gamma_{1,2} \\ q_{1,3} & \textit{on } \Gamma_{1,3} \end{cases}. \tag{3.76}$$

Let V_h *be an extended approximation space on a triangulation* $\mathcal{S}_h$*. Using the symmetric variant of Nitsche's method to include the boundary conditions at* Γ_D *and interface conditions for* Γ_1 *and* Γ_2*, defined by equations* (3.70) *and* (3.72) *to* (3.75)*, the discrete variational formulation of the problem reads: For* $v_h \in V_h$ *find* $u_h \in V_h$ *such that it is*

$$\underbrace{a(u_h, v_h) + \sum_{l=0}^{4} a_l(u_h, v_h)}_{=:\, a_h(u_h, v_h)} = \underbrace{L(v_h) + \sum_{l=0}^{4} L_l(v_h)}_{=:\, L_h(v_h)}, \tag{3.77}$$

where the bilinear forms are defined via

$$a(u_h, v_h) = \int_{\Omega_1 \cup \Omega_2 \cup \Omega_3} \xi u_h v_h \, \mathrm{d}x + \int_{\Omega_1 \cup \Omega_2 \cup \Omega_3} \kappa \nabla u_h \nabla v_h \, \mathrm{d}x \tag{3.78}$$

$$\begin{aligned} a_0(u_h, v_h) = \sum_{i=1}^{3} \Bigg(&- \int_{\partial\Omega_i \cap \Gamma_\mathrm{D}} \kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_i v_h|_{\Omega_i} \, \mathrm{d}x \\ &- \int_{\partial\Omega_i \cap \Gamma_\mathrm{D}} \kappa|_{\Omega_i} \nabla v_h|_{\Omega_i} \cdot \vec{n}_i u_h|_{\Omega_i} \, \mathrm{d}x + \int_{\partial\Omega_i \cap \Gamma_\mathrm{D}} \frac{\lambda_0}{h} u_h|_{\Omega_i} v_h|_{\Omega_i} \, \mathrm{d}x \Bigg), \end{aligned} \tag{3.79}$$

$$\begin{aligned} a_1(u_h, v_h) = &- \int_{\Gamma_1 \cap \Gamma_{1,2}} \{\kappa \nabla u_h \cdot \vec{n}_1\} [\![v_h]\!] \, \mathrm{d}x - \int_{\Gamma_1 \cap \Gamma_{1,2}} \{\kappa \nabla v_h \cdot \vec{n}_1\} [\![u_h]\!] \, \mathrm{d}x \\ &+ \int_{\Gamma_1 \cap \Gamma_{1,2}} \frac{\lambda_1}{h} [\![u_h]\!] [\![v_h]\!] \, \mathrm{d}x, \end{aligned} \tag{3.80}$$

$$\begin{aligned} a_2(u_h, v_h) = &- \int_{\Gamma_1 \cap \Gamma_{1,3}} \{\kappa \nabla u_h \cdot \vec{n}_1\} [\![v_h]\!] \, \mathrm{d}x - \int_{\Gamma_1 \cap \Gamma_{1,3}} \{\kappa \nabla v_h \cdot \vec{n}_1\} [\![u_h]\!] \, \mathrm{d}x \\ &+ \int_{\Gamma_1 \cap \Gamma_{1,3}} \frac{\lambda_2}{h} [\![u_h]\!] [\![v_h]\!] \, \mathrm{d}x, \end{aligned} \tag{3.81}$$

$$\begin{aligned}a_3(u_h, v_h) = &- \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_2} \nabla u_h|_{\Omega_2} \cdot \vec{n}_2 v_h|_{\Omega_2} \,\mathrm{d}x - \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_2} \nabla v_h|_{\Omega_2} \cdot \vec{n}_2 u_h|_{\Omega_2} \,\mathrm{d}x \\ &+ \int_{\Gamma_2 \cap \Gamma_{2,3}} \frac{\lambda_4}{h} u_h v_h \,\mathrm{d}x,\end{aligned} \tag{3.82}$$

$$\begin{aligned}a_4(u_h, v_h) = &\int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_3} \nabla u_h|_{\Omega_3} \cdot \vec{n}_2 v_h|_{\Omega_3} \,\mathrm{d}x + \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_3} \nabla v_h|_{\Omega_3} \cdot \vec{n}_2 u_h|_{\Omega_3} \,\mathrm{d}x \\ &+ \int_{\Gamma_2 \cap \Gamma_{2,3}} \frac{\lambda_4}{h} u_h v_h \,\mathrm{d}x,\end{aligned} \tag{3.83}$$

and the linear forms are given by

$$L(v_h) = \int_{\Omega_1 \cup \Omega_2 \cup \Omega_3} f v_h \,\mathrm{d}x - \int_{\Gamma_{N,h}} g_{\mathrm{N}} v_h \,\mathrm{d}x, \tag{3.84}$$

$$L_0(u_h, v_h) = \sum_{i=1}^{3} \left(\int_{\partial\Omega_i \cap \Gamma_{\mathrm{D}}} \kappa|_{\Omega_i} \nabla v_h|_{\Omega_i} \cdot \vec{n}_i g_{\mathrm{D}}|_{\Omega_i} \,\mathrm{d}x + \int_{\partial\Omega_i \cap \Gamma_{\mathrm{D}}} \frac{\lambda_0}{h} g_{\mathrm{D}}|_{\Omega_i} v_h|_{\Omega_i} \,\mathrm{d}x \right), \tag{3.85}$$

$$L_1(u_h, v_h) = - \int_{\Gamma_1 \cap \Gamma_{1,2}} \{\kappa \nabla v_h \cdot \vec{n}_1\} q_{1,2} \,\mathrm{d}x + \int_{\Gamma_1 \cap \Gamma_{1,2}} \langle v \rangle g_{1,2} + \int_{\Gamma_1 \cap \Gamma_{1,2}} \frac{\lambda_1}{h} q_{1,2} [\![v_h]\!] \,\mathrm{d}x, \tag{3.86}$$

$$L_2(u_h, v_h) = - \int_{\Gamma_1 \cap \Gamma_{1,3}} \{\kappa \nabla v_h \cdot \vec{n}_1\} q_{1,3} \,\mathrm{d}x + \int_{\Gamma_1 \cap \Gamma_{1,3}} \langle v \rangle g_{1,3} + \int_{\Gamma_1 \cap \Gamma_{1,3}} \frac{\lambda_2}{h} q_{1,3} [\![v_h]\!] \,\mathrm{d}x, \tag{3.87}$$

$$L_3(u_h, v_h) = - \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_2} \nabla v_h|_{\Omega_2} \cdot \vec{n}_2 g_2 \,\mathrm{d}x + \int_{\Gamma_2 \cap \Gamma_{2,3}} \frac{\lambda_3}{h} g_2 v_h \,\mathrm{d}x, \tag{3.88}$$

$$L_4(u_h, v_h) = \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_3} \nabla v_h|_{\Omega_3} \cdot \vec{n}_2 g_3 \,\mathrm{d}x + \int_{\Gamma_2 \cap \Gamma_{2,3}} \frac{\lambda_4}{h} g_3 v_h \,\mathrm{d}x, \tag{3.89}$$

with $\lambda_i \in \mathbb{R}$ as stabilization parameters. Therein, a_0 and L_0 include condition (3.70), a_1, a_2 and L_1, L_2 include the conditions (3.72) and (3.73), a_3 and L_3 include (3.74), and a_4 and L_4 include (3.75). Please note that the sign difference in the forms $a_4(\cdot,\cdot)$ and $L_4(\cdot)$ results from the normal $\vec{n}_2$.

3.3.3 Time-dependent problems with moving interfaces

After considering situations with steady-state domains, and hence stationary discontinuities, in the previous sections, we now comment on how to solve problems with moving or evolving geometries. We suppose that for $t \in [t_0, t_f]$, we have a hold-all domain $\Omega \subset \mathbb{R}^d$ and hierarchically ordered level set functions $\varphi_i(t)$, $i = 1, \ldots, N_{\mathrm{dom}} - 1$, with zero level sets $\tilde{\Gamma}_i(t)$ that are explicitly given, cf. Remark 2.3. The interfaces separating a hold-all domain Ω into N_{dom} disjoint subdomains $\Omega_l(t)$ are given by $\Gamma_i(t) \subseteq \tilde{\Gamma}_i(t)$, where each $\Gamma_i(t)$ is characterized by equation (3.33).

As already mentioned, the introduction of space-time eXtended finite elements [50, 62] can be seen as the natural approach to discretize problems involving moving or evolving discontinuities, cf. also [29, Chap. 10 and Chap. 11]. By considering the time as an additional spatial dimension, the idea of the space-time method is to transform the problem into a $d+1$ dimensional stationary problem. For the numerical solution, the space-time domain $\Omega \times [t_0, t_f]$ is then separated into time slabs $Q^n = \Omega \times (t_{n-1}, t_n]$, which are prisms resulting from considering the domain over the time interval $(t_{n-1}, t_n]$, and the problem is defined in a variational formulation for these time slabs that are usually decoupled by applying a discontinuous Galerkin method. While the numerical costs are more or less comparable to time stepping methods, the implementation, especially in regards to the subdivision of elements, is rather complex and challenging. Moreover, concepts such as the Nitsche method need to adapted. An extensive introduction and description of space-time methods for the eXtended finite element method is given in [50] which addresses two-phase mass transport problems. However, due to the complexity of space time eXtended finite element method in regards to both analysis and implementation, we use a more classical approach and discretize problems involving time dependent discontinuities using Rothe's method.

Remark 3.12 (Maintaining of the level set function $\varphi_i(t)$). In practice, the level set functions φ_i are usually only given for the initial state $\varphi_i(t_0)$ while their evolution in time is part of the solution. This introduces additional challenges since, firstly, the advantageous signed distance property is lost over time and, secondly and more bothersome, the level set method is not volume and mass conserving. As a consequence, some maintaining methods such as a reinitialization techniques and volume correction approaches are necessary [29, Chap. 7]. In addition to this, it is known that solving the hyperbolic level set problem using finite elements may require a stabilization technique such as the Streamline-Upwind-Petrov-Galerkin (SUPG) method [72]. All these issues are addressed in Section 4.4.1, however, at this point we neglect them by assuming that all functions are explicitly given for all $t \in [t_0, t_f]$ and focus on the presentation of the hierarchical eXtended finite element approach instead.

Rothe's method

In contrast to the space-time XFEM, we need that the quantity subject to the time derivative is defined on all domains and, moreover, sufficiently smooth, so that we can approximate the time derivate by a finite element approximation. While this might seem to be a significant restriction, most problems can be scaled so that the effected function is continuous. For example, the heat equation, where we usually have the term $\rho c \partial_t u$, with ρ denoting the density, c the specific heat capacity, and u the temperature, can be reformulated by introducing the thermal diffusivity coefficient $\kappa = \frac{\lambda}{\rho c}$, where λ is the thermal conductivity. However, it has to be noted that for such scenarios no higher order can be achieved in time as we can only discretize the time derivate by an explicit or implicit Euler scheme. To illustrate the scheme when using Rothe's method, we consider the a diffusion equation, where the interface evolution is prescribed and therefore not part of the problem.

Example 3.5 (Diffusion equation with prescribed interface evolution)**.** *For $N_{\mathrm{dom}} = 3$, let $\Omega \subset \mathbb{R}^d$ be a fixed domain that is polygonally bounded with $\partial\Omega = \Gamma_{\mathrm{D}} \cup \Gamma_{\mathrm{N}}$ and consists for $t \in [t_0, t_f]$ of the disjoint subdomains $\Omega_1(t)$, $\Omega_2(t)$, and $\Omega_3(t)$ separated by sharp and sufficiently smooth interfaces $\Gamma_{1,2}(t)$, $\Gamma_{1,3}(t)$, and $\Gamma_{2,3}(t)$ with outward-pointing unit normal vectors $\vec{n}_{1,2}(t)$, $\vec{n}_{1,3}(t)$, and $\vec{n}_{2,3}(t)$. For given data which is sufficiently smooth*[§]*, find $u(\cdot, t)$, $t \in [t_0, t_f]$, s.t. it solves the problem*

$$
\begin{aligned}
\partial_t u - \nabla \cdot (\kappa \nabla u) &= f && \text{in } \Omega_1(t) \cup \Omega_2(t) \cup \Omega_3(t), && (3.90)\\
u &= g_{\mathrm{D}} && \text{on } \Gamma_{\mathrm{D}}(t), && (3.91)\\
-\kappa \nabla u \cdot \vec{n} &= g_{\mathrm{N}} && \text{on } \Gamma_{\mathrm{N}}(t), && (3.92)\\
[\![\kappa \nabla u]\!] \cdot \vec{n}_{1,2} &= g_{1,2} && \text{on } \Gamma_{1,2}(t), && (3.93)\\
[\![u]\!] &= 0 && \text{on } \Gamma_{1,2}(t), && (3.94)\\
[\![\kappa \nabla u]\!] \cdot \vec{n}_{1,3} &= g_{1,3} && \text{on } \Gamma_{1,3}(t), && (3.95)\\
[\![u]\!] &= 0 && \text{on } \Gamma_{1,3}(t), && (3.96)\\
u|_{\Omega_2} &= g_2 && \text{on } \Gamma_{2,3}(t), && (3.97)\\
u|_{\Omega_3} &= g_2 && \text{on } \Gamma_{2,3}(t), && (3.98)\\
u(\cdot, t_0) &= u_0 && \text{in } \Omega_1(t_0) \cup \Omega_2(t_0) \cup \Omega_3(t_0), && (3.99)
\end{aligned}
$$

where the situation at $t = t_0$ is similar to the setting depicted in Figure 3.2.

Time discretization: Discretizing the time interval $[t_0, t_f]$ by $N_t + 1$ time steps into $t_n = t_0 + n\Delta t$, $n = 0, \dots, N_t$, where Δt denotes the time step size, we apply the implicit Euler time discretization to the diffusion problem which then reads: For $n = 0, \dots, N_t - 1$, find $u^{n+1} \approx u(\cdot, t_{n+1})$ such that

$$
\begin{aligned}
\frac{u^{n+1}}{\Delta t} - \nabla \cdot \left(\kappa^{n+1} \nabla u^{n+1}\right) &= f^{n+1} + \frac{u^n}{\Delta t}, && \text{in } \Omega_1(t_{n+1}) \cup \Omega_2(t_{n+1}) \cup \Omega_3(t_{n+1}) && (3.100)\\
u^{n+1} &= g_{\mathrm{D}}^{n+1} && \text{on } \Gamma_{\mathrm{D}}(t_{n+1}), && (3.101)\\
-\kappa^{n+1} \nabla u^{n+1} \cdot \vec{n}^{n+1} &= g_{\mathrm{N}}^{n+1} && \text{on } \Gamma_{\mathrm{N}}(t_{n+1}), && (3.102)\\
[\![\kappa^{n+1} \nabla u^{n+1}]\!] \cdot \vec{n}_{1,2}^{n+1} &= g_{1,2}^{n+1} && \text{on } \Gamma_{1,2}(t_{n+1}), && (3.103)\\
[\![u^{n+1}]\!] &= 0 && \text{on } \Gamma_{1,2}(t_{n+1}), && (3.104)\\
[\![\kappa^{n+1} \nabla u^{n+1}]\!] \cdot \vec{n}_{1,3}^{n+1} &= g_{1,3}^{n+1} && \text{on } \Gamma_{1,3}(t_{n+1}), && (3.105)\\
[\![u^{n+1}]\!] &= 0 && \text{on } \Gamma_{1,3}(t_{n+1}), && (3.106)\\
u|_{\Omega_2}^{n+1} &= g_2^{n+1} && \text{on } \Gamma_{2,3}(t_{n+1}), && (3.107)\\
u|_{\Omega_3}^{n+1} &= g_2^{n+1} && \text{on } \Gamma_{2,3}(t_{n+1}). && (3.108)
\end{aligned}
$$

For a fixed $n \in \{0, \dots, N_t - 1\}$, we use the notation $\xi = \frac{1}{\Delta t}$ and define $\Omega_i := \Omega_i(t_{n+1})$, $i = 1, 2, 3$, $u := u^{n+1}$, et cetera. After summarizing the right-hand-side in (3.100) by $\tilde{f} := f + \xi u^n$, we end

[§]Notably, it is $[\![u]\!] = 0$ at all $\Gamma_{i,l}$ as we require continuity of u across all interfaced.

up with the (quasi-)stationary problem: Find u so that it solves the problem given by

$$\xi u - \nabla \cdot (\kappa \nabla u) = \tilde{f} \qquad \text{in } \Omega_1 \cup \Omega_2 \cup \Omega_3, \tag{3.109}$$

$$u = g_{\mathrm{D}} \qquad \text{on } \Gamma_{\mathrm{D}} \tag{3.110}$$

$$-\kappa \nabla u \cdot \vec{n} = g_{\mathrm{N}} \qquad \text{on } \Gamma_{\mathrm{N}}, \tag{3.111}$$

$$[\![\kappa \nabla u]\!] \cdot \vec{n}_{1,2} = g_{1,2} \qquad \text{on } \Gamma_{1,2}, \tag{3.112}$$

$$[\![u]\!] = 0 \qquad \text{on } \Gamma_{1,2}, \tag{3.113}$$

$$[\![\kappa \nabla u]\!] \cdot \vec{n}_{1,3} = g_{1,3} \qquad \text{on } \Gamma_{1,3}, \tag{3.114}$$

$$[\![u]\!] = 0 \qquad \text{on } \Gamma_{1,3}, \tag{3.115}$$

$$u|_{\Omega_2} = g_2 \qquad \text{on } \Gamma_{2,3}, \tag{3.116}$$

$$u|_{\Omega_3} = g_2 \qquad \text{on } \Gamma_{2,3} \tag{3.117}$$

for each time step. Obviously, the derived problem corresponds to the already presented stationary Example 3.1.

Spatial discretization: Obviously, the derived sequence of steady-state problems resembles the already presented stationary situation, in particular Example 3.1. Consequently, it can be directly discretized using the hierarchical finite element method where interface (and optionally also the Dirichlet boundary) conditions are imposed using Nitsche's method. Hence, the full discretized problem is given by:

Let V_h be an extended approximation space on a triangulation $\mathcal{S}_h$. Using the symmetric variant of Nitsche's method to include the boundary conditions at Γ_D and interface conditions for Γ_1 and Γ_2, defined by equations (3.70) and (3.72) to (3.75), the discrete variational formulation of the problem reads: For $v_h \in V_h$ find $u_h \in V_h$ such that

$$\underbrace{a(u_h, v_h) + \sum_{l=0}^{4} a_l(u_h, v_h)}_{=:\, a_h(u_h, v_h)} = \underbrace{L(v_h) + \sum_{l=0}^{4} L_l(v_h)}_{=:\, L_h(v_h)}, \tag{3.118}$$

where the bilinear forms are defined as

$$a(u_h, v_h) = \int_{\Omega_1 \cup \Omega_2 \cup \Omega_3} \xi u_h v_h \,\mathrm{d}x + \int_{\Omega_1 \cup \Omega_2 \cup \Omega_3} \kappa \nabla u_h \nabla v_h \,\mathrm{d}x, \tag{3.119}$$

$$\begin{aligned} a_0(u_h, v_h) = \sum_{i=1}^{3} \Bigg(&- \int_{\partial\Omega_i \cap \Gamma_{\mathrm{D}}} \kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_i v_h|_{\Omega_i} \,\mathrm{d}x \\ &- \int_{\partial\Omega_i \cap \Gamma_{\mathrm{D}}} \kappa|_{\Omega_i} \nabla v_h|_{\Omega_i} \cdot \vec{n}_i u_h|_{\Omega_i} \,\mathrm{d}x + \int_{\partial\Omega_i \cap \Gamma_{\mathrm{D}}} \frac{\lambda_0}{h} u_h|_{\Omega_i} v_h|_{\Omega_i} \,\mathrm{d}x \Bigg), \end{aligned} \tag{3.120}$$

$$\begin{aligned} a_1(u_h, v_h) = &- \int_{\Gamma_1 \cap \Gamma_{1,2}} \{\kappa \nabla u_h \cdot \vec{n}_1\} [\![v_h]\!] \,\mathrm{d}x - \int_{\Gamma_1 \cap \Gamma_{1,2}} \{\kappa \nabla v_h \cdot \vec{n}_1\} [\![u_h]\!] \,\mathrm{d}x \\ &+ \int_{\Gamma_1 \cap \Gamma_{1,2}} \frac{\lambda_1}{h} [\![u_h]\!] [\![v_h]\!] \,\mathrm{d}x, \end{aligned} \tag{3.121}$$

$$\begin{aligned} a_2(u_h, v_h) = &- \int_{\Gamma_1 \cap \Gamma_{1,3}} \{\kappa \nabla u_h \cdot \vec{n}_1\} [\![v_h]\!] \,\mathrm{d}x - \int_{\Gamma_1 \cap \Gamma_{1,3}} \{\kappa \nabla v_h \cdot \vec{n}_1\} [\![u_h]\!] \,\mathrm{d}x \\ &+ \int_{\Gamma_1 \cap \Gamma_{1,3}} \frac{\lambda_2}{h} [\![u_h]\!] [\![v_h]\!] \,\mathrm{d}x, \end{aligned} \tag{3.122}$$

$$\begin{aligned} a_3(u_h, v_h) = &- \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_2} \nabla u_h|_{\Omega_2} \cdot \vec{n}_2 v_h|_{\Omega_2} \,\mathrm{d}x - \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_2} \nabla v_h|_{\Omega_2} \cdot \vec{n}_2 u_h|_{\Omega_2} \,\mathrm{d}x \\ &+ \int_{\Gamma_2 \cap \Gamma_{2,3}} \frac{\lambda_3}{h} u_h v_h \,\mathrm{d}x, \end{aligned} \tag{3.123}$$

$$\begin{aligned} a_4(u_h, v_h) = &\int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_3} \nabla u_h|_{\Omega_3} \cdot \vec{n}_2 v_h|_{\Omega_3} \,\mathrm{d}x + \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_3} \nabla v_h|_{\Omega_3} \cdot \vec{n}_2 u_h|_{\Omega_3} \,\mathrm{d}x \\ &+ \int_{\Gamma_2 \cap \Gamma_{2,3}} \frac{\lambda_3}{h} u_h v_h \,\mathrm{d}x, \end{aligned} \tag{3.124}$$

and the linear forms are given by

$$L(v_h) = \int_{\Omega_1 \cup \Omega_2 \cup \Omega_3} \tilde{f} v_h \,\mathrm{d}x - \int_{\Gamma_{N,h}} g_{\mathrm{N}} v_h \,\mathrm{d}x, \tag{3.125}$$

$$L_0(u_h, v_h) = \sum_{i=1}^{3} \left(\int_{\partial\Omega_i \cap \Gamma_{\mathrm{D}}} \kappa|_{\Omega_i} \nabla v_h|_{\Omega_i} \cdot \vec{n}_i g_{\mathrm{D}}|_{\Omega_i} \,\mathrm{d}x + \int_{\partial\Omega_i \cap \Gamma_{\mathrm{D}}} \frac{\lambda_0}{h} g_{\mathrm{D}}|_{\Omega_i} v_h|_{\Omega_i} \,\mathrm{d}x \right), \tag{3.126}$$

$$L_1(u_h, v_h) = \int_{\Gamma_1 \cap \Gamma_{1,2}} \langle v \rangle g_{1,2} \,\mathrm{d}x, \tag{3.127}$$

$$L_2(u_h, v_h) = \int_{\Gamma_1 \cap \Gamma_{1,3}} \langle v \rangle g_{1,3} \,\mathrm{d}x, \tag{3.128}$$

$$L_3(u_h, v_h) = - \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_2} \nabla v_h|_{\Omega_2} \cdot \vec{n}_2 g_2 \,\mathrm{d}x + \int_{\Gamma_2 \cap \Gamma_{2,3}} \frac{\lambda_3}{h} g_2 v_h \,\mathrm{d}x, \tag{3.129}$$

$$L_4(u_h, v_h) = \int_{\Gamma_2 \cap \Gamma_{2,3}} \kappa|_{\Omega_3} \nabla v_h|_{\Omega_3} \cdot \vec{n}_2 g_2 \,\mathrm{d}x + \int_{\Gamma_2 \cap \Gamma_{2,3}} \frac{\lambda_3}{h} g_2 v_h \,\mathrm{d}x, \tag{3.130}$$

with $\lambda_i \in \mathbb{R}$ as stabilization parameters. However, the right-hand-side term $\tilde{f}$ in equation (3.125) demands particular attention in regards to the numerics as it contains a term of type

$$\int_{\Omega_1(t_{n+1}) \cup \Omega_2(t_{n+1}) \cup \Omega_3(t_{n+1})} \xi u_h(t_n) v_h(t_{n+1}) \,\mathrm{d}x, \tag{3.131}$$

which involves functions that are enriched by (potentially) different locations of the discontinuities $\Gamma_{i,l}(t_n)$ and $\Gamma_{i,l}(t_{n+1})$, $1 \leq i < l \leq 3$. We will point this out in more detail and address the implementation aspects in Section 4.3.

Chapter 4

Automated solution of multiphysics problems involving discontinuities

The numerical solution of multiphysics problems involving discontinuities is very challenging, not only with respect to the mathematical theory, but especially in regards to the implementation. While conventional finite element methods can be used for solving a wide range of physical and engineering problems, eXtended discretization methods such as the method presented in Chapter 3 are much better suited for this kind of problems, see Section 2.3.

Unfortunately, the implementation of conventional finite element methods into a more general framework is already difficult and error-prone and, thus, takes a long time. Moreover, small changes within the problem formulation or the discretization often result in a lot of code changes and, hence, require intensive testing. This makes the solution of fundamentally different PDE problems using one framework even more time consuming. All these aspects are especially true in regards to the implementation of an eXtended finite element method which, among others, needs concepts to

(XFEM 1) represent and track interfaces and unfitted boundaries,

(XFEM 2) enrich function spaces with respect to the positions of interfaces and domain boundaries,

(XFEM 3) perform quadrature on (possibly multiple-)intersected elements, and

(XFEM 4) impose conditions on the discontinuities.

A very interesting approach to overcome the implementation drawbacks of the (eXtended) finite element method is *automated code generation* which is also known as automated programming or meta-programming. The basic idea of automated code generation is to write meta-programs that can interpret code written using a higher level of abstraction and generate corresponding lower level code which the users would otherwise have to write themselves. While the implementation of a library taking advantage of automated code generation is very complex and sophisticated, using this approach for the development of a full-featured framework has several advantages. The most important benefits are that it allows for so-called abstract coding meaning that the user can write high-level code which is translated to low-level code and, therefore, for rapid prototyping. Moreover, the generated code is consistent and the maintaining is reduced to a minimum.

The idea of automated code generation for solving PDE problems with finite elements is, at least in part, used in many frameworks such as `FreeFEM++` [73], `COMSOL` [74], and the `FEniCS` project [1]. Such frameworks are usually based on the idea of separating a problem's discretization from the implementation of methods used for computing the numerical solution. Then, the discretization of problems can be considered with approaches based on automated code generation. As a result, the implementation and solution of different PDE problems from various areas becomes rather simple in such frameworks. Unfortunately, all mentioned libraries only feature the use of conventional finite element methods but do not provide eXtended discretization methods. This is due to the fact that, in addition to fundamental aspects that need to be addressed by any implementation supporting eXtended discretization methods, automating the generation of code for such methods is especially complex. In contrast to the conventional finite element method, a lot of numerical methods required by eXtended discretization methods depend on information that are only known at run-time. In particular this concerns methods that depend on the actual characteristic of the discontinuity defining the enrichment such as quadrature routines for intersected elements. Consequently, it is not obvious how the discretization of a problem can be separated from the implementation of methods used for computing the numerical solution.

While this issue has to be resolved on a technical level, it is obvious that the hierarchical eXtended finite element method is, by design, very suited to be implemented into a framework that makes use of automated code generation. Especially since the hierarchical order defines some kind of tree structure of how and where the required enrichments have to be added and, moreover, introduces a natural oder of basis functions and corresponding coefficients. Since from our point of view, the `FEniCS` framework is not only an open source project but also offers the most convenient framework to solve PDE-based problems and takes the automated code generation approach to the next level, we choose to implement the hierarchical eXtended finite element method as a `FEniCS` toolbox called `miXFEM`.

4.1 The `FEniCS` project

`FEniCS` is a collaborative project of researchers who develop tools for automated scientific computing, especially in the field of finite element methods for the solution of partial differential equations [1]. It was initially created in 2003 and has since then been extended and updated regularly. `FEniCS` offers a `python` and a `C++` interface and consists of a collection of core components such as

- the `Unified Form Language UFL` [75, 76], which is an implementation of an abstract domain-specific language used to define the finite element discretizations of differential equations in a coded notation that resembles very closely the standard mathematical notation used in variational formulations,
- the `FEniCS Form Compiler` (`FFC`) [77–79], which analyzes given `UFL` code and, in combination with the library `FIAT` [80, 81], generates low-level `C++` code, corresponding to the discretized variational forms specified in the `UFL` file, that is compatible with the `UFC` [82, 83] interface for arbitrary finite elements on simplices,
- `DOLFIN` [84, 85], the main problem solving environment and user interface whose functionality integrates the other `FEniCS` components. Additionally, it handles communication with external libraries or toolboxes such as `miXFEM`.

An overview of the relationships between the components of `FEniCS` and external software is given in Figure 4.1 [85]. Depending on the `FEniCS` version, there are some other core components that are not listed here.

Recall that the process of solving a PDE-based problem consists, in principle, of discretizing the domain, introducing a (discrete) variational formulation that includes the definition of the approximation spaces, and solving the resulting system of equations. The idea of the `FEniCS` framework is to separate a problems's discretization from the implementation of methods used for computing the numerical solution. Therefore, solving a problem with `FEniCS` consists of the following steps:

(1) Define the discrete variational formulation of a PDE-based problem using the high abstraction level of the `UFL` language, then

(2) use the `FFC` to automatically generate low-level `C++` code corresponding to the discretization of the variational problem specified in the `*.ufl` file, and finally,

(3) implement the actual problem, including the computational mesh, function definitions, et cetera in `DOLFIN`, therefore making use of the automatically generated code and the provided interfaces to third-party libraries,

see Figure 4.2. In the following paragraphs, we present a more detailed overview of the mentioned core components as we will have to enhance them to implement the hierarchical eXtended finite element method as toolbox for the `FEniCS` framework, thereby, allowing for the automated solution of problems involving discontinuities. For a better understanding, we demonstrate the

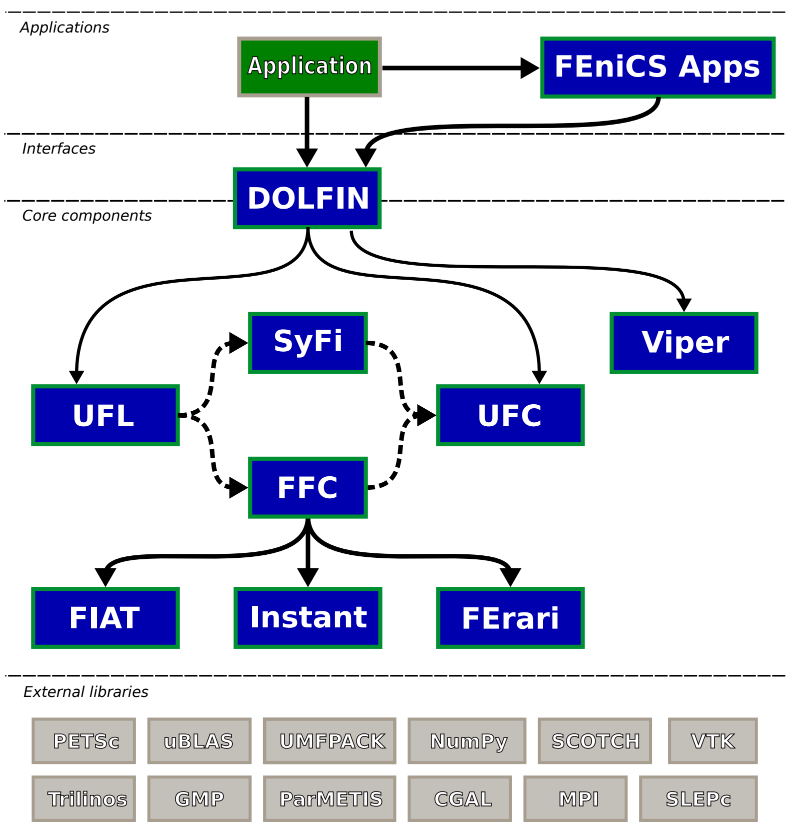


Figure 4.1 Overview of FEniCS: DOLFIN functions as the main user interface and handles the communication between all other FEniCS components and external libraries. The solid lines indicate dependencies and dashed lines indicate data flow [85].

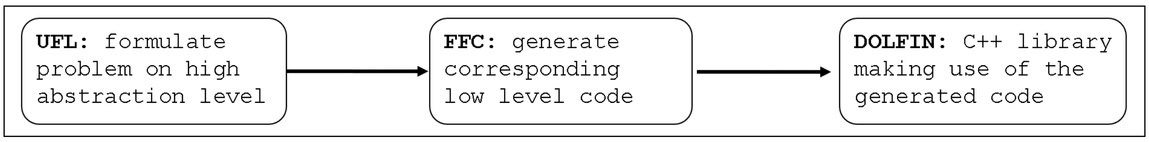

UFL: formulate problem on high abstraction level
FFC: generate corresponding low level code
DOLFIN: C++ library making use of the generated code

Figure 4.2 Overview of the FEniCS' approach to solve PDE-based problems: Formulate the discrete variational problem in UFL, use the FFC to generate the corresponding code, and solve the problem with DOLFIN.

workflow of solving a PDE problem with FEniCS by again considering Example 2.2 from Section 2.2.

4.1.1 The Unified Form Language (UFL)

The domain specific language Unified Form Language (UFL) [75, 76] can be used to define finite element discretizations of variational formulations, which resemble the mathematical notations of function spaces and variational formulations. It provides a syntax for finite element spaces declarations of a range of predefined basic element families, for example, Lagrange and Discontinuous Lagrange, representing scalar continuous or discontinuous Lagrange finite elements, respectively. All these basic scalar elements can be combined to construct vector elements, tensor elements or arbitrary mixed elements.

Recall that the discrete variational formulation of the steady-state diffusion problem Example 2.2 reads: Find $u_h \in V^1_{\mathrm{cg},h}$ s.t. for all $v_h \in V^1_{\mathrm{cg},h}$, $u_h = g_{\mathrm{D},h}$ on Γ_D and

$$\underbrace{\int_\Omega \kappa \nabla u_h \cdot \nabla v_h \,\mathrm{d}x}_{a(u_h,v_h)} = \underbrace{\int_\Omega f_h v_h \,\mathrm{d}x - \int_{\Gamma_\mathrm{N}} g_{\mathrm{N},h} v_h \,\mathrm{d}x + \int_{\Gamma_{1,2}} g_{1,2,h} v_h \,\mathrm{d}x}_{L(v_h)} \tag{4.1}$$

holds. The discrete function space $V^1_{\mathrm{cg},h}$ can be defined in `UFL` by

```
CG1 = FiniteElement("Lagrange", triangle, 1)
```

where the keyword "triangle" states the dimension and shape of the element and the value "1" defines the polynomial degree of the basis functions.

The most basic expressions in `UFL` are form arguments, which do not depend on any other expression. Form arguments include basis test and trial functions and coefficient functions which are represented by `TestFunction`, `TrialFunction`, and `Coefficient` classes. More sophisticated expressions can then be constructed using the form arguments in combination with so-called operators (which are also expressions).

For the given example, we define

```
u = TrialFunction(CG1)
v = TestFunction(CG1)
k = Coefficient(DG0)
f = Coefficient(CG1)
gN = Coefficient(CG1)
gI = Coefficient(CG1)
```

and specify the forms

```
a = k * inner(grad(u),grad(v)) * dx(0) + k * inner(grad(u),grad(v)) * dx(1)
L = f * v * dx  -  gN * v * ds(0)  +  gI * v * dS(0)
```

where `dx` denotes a volume measure or cell integral and `ds` is a boundary measure on boundary facets called exterior facet integral. In total, `UFL` supports the measures

- $\int_{\Omega_i}(\cdot)\mathrm{d}x \leftrightarrow (\cdot)$ *`dx`(i) (cell integral),
- $\int_{\partial\Omega_i}(\cdot)\mathrm{d}s \leftrightarrow (\cdot)$*`ds`$(i)$ (exterior facet integral),
- $\int_{F_i}(\cdot)\mathrm{d}S \leftrightarrow (\cdot)$*`dS`$(i)$ (interior facet integral),

on a domain Ω with external boundary $\partial\Omega$ and interior triangulation facets F, where the index i, denoting a subdomain, can be dropped if no subdomain is present or if the function is piecewise defined on the subdomains.

4.1.2 FEniCS Form Compiler (FFC)

Arguably the most essential component of the automated code generation approach of FEniCS is the FEniCS Form Compiler (FFC) [78, 79]. The FFC receives UFL files containing variational forms as input and produces the corresponding low-level C++ code for the evaluation of element tensors, finite element basis functions and their derivatives. Thereby, the FFC relies on the FInite element Automatic Tabulator (FIAT) [81] which implements a mathematical framework for the definition and evaluation of various finite element basis functions on simplices and can tabulate quadrature points and quadrature weights required for the numerical integration. The code generated by FFC is compatible with the Unified Form-assembly Code (UFC) interface [83] which is a framework for finite element assembly and works as an interface layer between problem-specific and general-purpose components of finite element programs. UFC consists of a single header file ufc.h specifying a C++ interface by very few base classes with virtual functions which must be implemented by a library or code that complies with the UFC specification. Thereby, UFC is independent of any other FEniCS component and can be used in a wide range of finite element solvers.

Generating low level code, that corresponds to a provided UFL file containing a discrete variational problem, with the FFC is very simple and can be done by the (bash) command

```
ffc -l dolfin <example>.ufl
```

Then, the work flow of the FFC consists of the following steps [79]:

(FFC 1) **Language parsing:** The user-specified form is interpreted and stored as a UFL abstract syntax tree (AST)

(FFC 2) **Code analysis:** The UFL form is preprocessed and form metadata (FormData) extracted, such as which elements were used to define the form, the number of coefficients and the cell type (intervals, triangles or tetrahedra).

(FFC 3) **Code representation:** The preprocessed form is examined and all data needed for the code generation is created. This includes the generation of finite element basis functions, extraction of data for mapping of degrees of freedom, and possible precomputation of integrals. The generated data is called intermediate representation. This step is the most complex task during the compiling process.

(FFC 4) **Optimization:** The intermediate representation is analyzed and parts are optimized.

(FFC 5) **Code generation:** Based on the (optimized) intermediate representation data, the actual C++ code for the body of each UFC function is generated. The code is stored as a dictionary which maps names of UFC functions to strings containing the C++ code.

(FFC 6) **Code formatting:** The generated code is formatted according and conforming to the UFC standard.

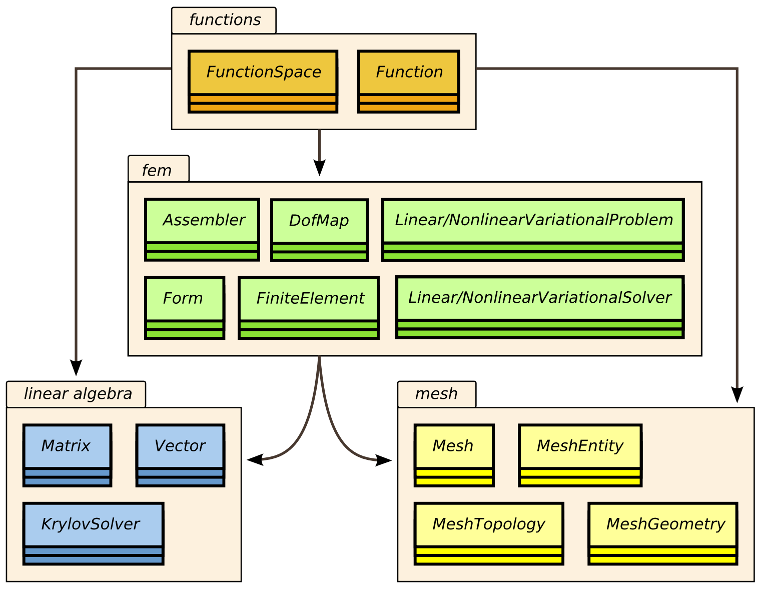


Figure 4.3 Schematic overview of some core components of DOLFIN. The arrows indicate dependencies of the modules [1, Chap. 10].

In regards to the steady-state diffusion equation, the automatically generated header file contains about 2900 lines of code.

4.1.3 DOLFIN library

DOLFIN [84, 85] is a C++/python solver library that functions as the main user interface of the FEniCS project. DOLFIN not only implements some core functionalities of FEniCS, including algorithms for manipulating meshes and finite element assembly, but also wraps the functionality of other FEniCS components as well as external software, and handles the communication between these components. Thereby, it provides a problem solving environment for PDE-based finite element models that relies on third-party libraries like PETSc [86] which can be used as linear algebra framework or VTK [87] for visualization purposes.

DOLFIN is organized in modules that each covers a certain area of functionality such as meshes, function spaces and functions, methods directly related to finite elements, linear algebra, and file input/output and visualization. A schematic overview of some core components including dependencies is illustrated in Figure 4.3 [1, Chap. 10].

The implementation structure of Example 2.2 using the automatically generated header file SteadyStateDiff.h in DOLFINs C++ interface is given in Figure 4.4. To point out the relation between DOLFIN and the automatically generated code, we exemplary highlight the finite element assembly work flow. Given the variational forms, the assembly works as follows [1, Chap. 6]: For each cell, DOLFIN maps the local indicies of the given cell's degrees of freedom to the global ones and calls code that is generated by the FFC to tabulate and compute the local tensor values. After performing the local operation, these values are then added to a global tensor.

```
#include <dolfin.h>
#include "SteadyStateDiff.h"

using namespace dolfin;

// Source term (right-hand side)
class Source : public Expression {
  [...]
};
// Normal derivative (Neumann boundary condition)
class gN : public Expression {
  [...]
};
// Flux across the interface
class gI : public Expression {
  [...]
};
// Sub domain for Dirichlet boundary condition
class DirichletBoundary : public SubDomain {
  [...]
};

int main() {
  // Create mesh and function space
  [..]
  // Define boundary condition
  [...]
  // Define variational forms using SteadyStateDiff.h
  [...]
  // Assemble matrix and right-hand-side, include boundary conditions
  [...]
  // Compute solution
  [...]
  // Save solution in VTK format
  [...]
  return 0;
}
```

Figure 4.4 Implementation structure of Example 2.2 in DOLFIN (C++ interface).

The DOLFIN code is shown in Figure 4.5 [1, Chap. 6] and some part of the code generated by the FFC for computing the local contributions of the cell to the matrix representing the bilinear form that is associated with Example 2.2 discretized by linear elements is given in Figure 4.6.

4.2 The PUM library

While the FEniCS framework offers a very convenient approach to solve PDE-based problems using (conventional) finite element methods, it lacks the features to solve problems involving discontinuities that are not resolved by the computational mesh. A first approach trying to extend the FEniCS framework for considering problems with discontinuities on unfitted meshes was the PUM[*] library [54, 88, 89]. For this purpose, the library, which is more a proof of concept

[*]The name PUM library comes from *partition-of-unity method* (PUM) which in some articles is used as synonym for the eXtended finite element method.

```
for (CellIterator cell(mesh); !cell.end(); ++cell) {
  [...]
  // Get local-to-global dofmap for each dimension
  for (uint i = 0; i < form_rank; ++i)
    dofs[i] = &(dofmaps[i]->cell_dofs(cell->index()));
  // Tabulate cell tensor
  integral->tabulate_tensor(&ufc.A[0], ufc.w(), ufc.cell);
  // Add entries to global tensor
  A.add(&ufc.A[0], dofs);
}
```

Figure 4.5 Assembly in DOLFIN [1, Chap. 6]

```
// Tabulate the tensor for the contribution from a local cell
virtual void tabulate_tensor(double* A, const double * const * w, const
    ufc::cell& c) const
{
  // Extract vertex coordinates
  const double * const * x = c.coordinates;
  // Compute Jacobian of affine map from reference cell,
  // Compute determinant of Jacobian and the inverse of the Jacobian
  [...]
  // Array of quadrature weights.
  static const double W1 = 0.5;
  // Value of basis functions at quadrature points.
  static const double FE0[1][3] = {{0.333333333333333, 0.333333333333333,
    0.333333333333333}};
  static const double FE0_D01[1][3] = {{-1.0, 0.0, 1.0}};
  static const double FE0_D10[1][3] = {{-1.0, 1.0, 0.0}};
  // Reset values in the element tensor.
  [...]
  // Coefficient declarations.
  double F0 = 0.0;
  for (unsigned int r = 0; r < 3; r++)
  {
    F0 += FE0[0][r]*w[0][r];
  }
  for (unsigned int j = 0; j < 3; j++)
  {
    for (unsigned int k = 0; k < 3; k++)
    {
      A[j*3 + k] += [...]
    }
  }
}
```

Figure 4.6 Part of the generated code for computing the local contributions to the matrix representing the bilinear form that is associated with Example 2.2 discretized by linear elements.

than a ready-to-use toolbox, uses and extends the key components UFL, FFC and DOLFIN. To, at least partly, meet the requirements (XFEM 1) to (XFEM 4), the PUM library has to address the following aspects

(Ext. 1) For the definition of variational forms, it is required to define extended function spaces and functions.

(Ext. 2) In addition, concepts for computing integrals over discontinuities, to, for example, consider fluxes or jumps, are mandatory.

(Ext. 3) Discontinuities have to be defined, e.g. by the zero level of an indicator function.

(Ext. 4) Basis functions of the conventional approximation space near the discontinuities have to be enriched accordingly to the chosen enrichment scheme and additional degrees of freedom have to be introduced.

(Ext. 5) Many routines providing the concepts related to the eXtended finite element methods have to be implemented, such as a method for performing quadrature on enriched elements.

In the following, we provide a rather detailed overview of the design and implementation of the `PUM` library. Based on this presentation, we can illustrate the new concepts and made extensions that are integrated in our toolbox `miXFEM` which is described in Section 4.3.

4.2.1 Design and implementation details of the `PUM` library

Mathematically, the `PUM` library is based on the classical eXtended finite element method where the enrichment of the approximation space is introduced via Heaviside functions, see Section 2.3.2. Within the toolbox, discontinuities can be defined by a priori knowledge of zero level sets of steady-state level set functions, and the intersection segments of an interfaces with an element is approximated by a linear segment. While the consideration of problems involving more than one discontinuity is possible in principle, the implemented approach can only handle exactly one level of enrichment. Thus, it requires a sufficiently large distance between discontinuities so that they can be considered independently and no basis function is multi-enriched. To impose interface and boundary conditions, the penalty method is used, cf. Section 2.3.

As mentioned in the beginning of Chapter 4, the main issues in regard to the automated code generation for eXtended discretization methods is the dependency of methods on run-time data. The `PUM` library resolves this issue by introducing an additional abstraction level comprised of interface objects called `GenericSurface` and `GenericPUM`. Both objects contain purely virtual methods and provide access to

(`GenericSurface` 1) the level set function, to determine the domain a given (quadrature) point lies in,

(`GenericSurface` 2) functions indicating if a cell is intersected or a given point is on an interface or facet,

(`GenericPUM` 1) data about the enriched elements, basis functions and degrees of freedom on each cell, (`GenericPUM` 2) methods for tabulating the mentioned objects, and

(`GenericPUM` 3) quadrature rules for quantities on interfaces, facets, and intersected elements,

by using the inheritance concept of `C++`. The generated code utilizes these interface objects so that the generated methods are independent of the actual representation of the discontinuity and the chosen enrichment scheme. By doing so, all required information related to the discontinuity and enrichment, that is given by the problem's implementation using the `DOLFIN-PUM` library, can be passed during run-time to the code automatically generated by the `FFC-PUM`. Hence, the implementation of methods required by the eXtended finite element method can be separated from the automated code generation used for the discretization of partial differential equations.

In the following, the most important extension of the `FEniCS`' key components `UFL`, `FFC` and `DOLFIN` are presented. Similar to the previous section, we demonstrate the workflow within the `PUM` library in the next sections using Example 2.2 for illustration purposes, where the interface $\Gamma_{1,2}$ is now unfitted to the triangulation.

4.2.1.1 Reinterpreting concepts of `UFL`

Technically, `UFL` does not offer the feature to define eXtended finite element spaces and, hence, does not explicitly meet the requirements (Ext. 1) and (Ext. 2). However, it does provide the necessary abstractions and concepts which, in combination with an adapted form compiler, are used by the `PUM` library to represent variational forms involving discontinuous quantities.

In detail, the `PUM` reinterprets the `UFL` concepts `EnrichedElement` and `RestrictedElement`, originally added to construct elements such as the Raviart–Thomas element. With this, an extended finite element function space with Heaviside enrichment, whose basis functions can be used to represent solutions with discontinuous features across a discontinuity, is introduced by

(1) defining the underlying (continuous, standard FEM) function space V^{FEM},

(2) restricting V^{FEM} onto the subdomains around discontinuity using the identifier `dc` to obtain V^{Γ},

(3) create the enriched function space $V^{\mathrm{XFEM}} = V^{\mathrm{FEM}} \bigoplus V^{\Gamma}$.

In regards to considering integrals over discontinuities, the `PUM` library introduced a new measure $\int_\Gamma (\cdot)\,\mathrm{dc} \leftrightarrow (\cdot)$ `*dc`. In combination with the `jump` operator already implemented in `UFL`, this allows to impose interface conditions using the penalty method.

Considering Example 2.2 using the mathematical framework of the `PUM` library, i.e. extending conventional function spaces via Heaviside enrichment, the construction of V_h according to equation (2.20) in `UFL` is as follows:

```
CG1_c = FiniteElement("Lagrange", cell, 1)
CG1_0 = RestrictedElement(CG1_c, dc)
# Enriched spaces $V_h$
CG1_e = CG1_c + CG1_0
```

Once an eXtended finite element function space is declared, basis and coefficient functions as well as the variational forms can be defined as before.

```
u = TrialFunction(CG1_e)
v = TestFunction(CG1_e)
k = ...
[...]
```

To include jumps or fluxes of functions over a discontinuity, the `UFL` syntax for restricting functions on interior facets is used, that is `u('+')` and `u('-')`. However, due to the newly defined measure `dc`, such integrals are interpreted differently, meaning that the function is restricted onto the positive and negative sides of a discontinuity. The discrete variational formulation of Example 2.2 when using the penalty method to enforce interface conditions is then given by: Find $u_h \in V^1_{\mathrm{cg},h}$ s.t. for all $v_h \in V^1_{\mathrm{cg},h}$ it is $u_h = g_{\mathrm{D},h}$ on Γ_{D} and

$$\int_\Omega \kappa \nabla u_h \cdot \nabla v_h \,\mathrm{d}x + \int_\Omega \lambda [\![u_h]\!][\![v_h]\!] \,\mathrm{d}x = \int_\Omega f_h v_h \,\mathrm{d}x - \int_{\Gamma_{\mathrm{N}}} g_{\mathrm{N},h} v_h \,\mathrm{d}x + \int_{\Gamma_{1,2}} g_{1,2,h} v_h \,\mathrm{d}x \qquad (4.2)$$

holds[†], which can be implemented in `UFL` by

```
a = k * inner(grad(u), grad(v)) * dx + lamb * jump(u) * jump(v) *dc
L = f * v * dx  -  gN * v * ds(0)  +  gI * v * dS(0)
```

4.2.1.2 The `FFC-PUM`

The `PUM` library uses already existing concepts in `UFL` to specify problems involving a discontinuity as high-level inputs, however, the automated code generation for such problems relies on an extended form compiler reinterpreting the additional abstractions correctly. Hence, the `PUM` compiler `FFC-PUM` has been developed to support new functionalities specific to the partition of unity framework. The fundamental extensions are

(`FFC-PUM` 1) the addition of mechanisms for generating an intermediate code representation for forms and function spaces involving discontinuities, and

(`FFC-PUM` 2) the implementation of routines to separate standard FEM terms and enriched expressions for volume integrals, interior and exterior facet integrals and integrals over discontinuities,

[†]Please note that due to $q_{1,2} = 0$, there is no penalty term on the right-hand-side.

```
// STANDRAD FENICS
class Form_0: public dolfin::Form
{
  public:
  // Constructor
  Form_0(const dolfin::FunctionSpace& V1, const dolfin::FunctionSpace& V0):
  dolfin::Form(2, 2), k(*this, 0), w(*this, 1)
  {
    _function_spaces[0] = reference_to_no_delete_pointer(V0);
    _function_spaces[1] = reference_to_no_delete_pointer(V1);

    _ufc_form = boost::shared_ptr<const ufc::form>(new poisson_form_0());
  }
}
```

```
// PUM LIBRARY
class Form_0: public dolfin::Form
{
  public:
  // Constructor
  Form_0(const pum::FunctionSpace& V0, const pum::FunctionSpace& V1):
  dolfin::Form(2, 2), k(*this, 0), w(*this, 1)
  {
    _function_spaces[0] = reference_to_no_delete_pointer(V0);
    _function_spaces[1] = reference_to_no_delete_pointer(V1);

    _ufc_form = boost::shared_ptr<const ufc::form>(new
   poisson_form_0(V0.pum_objects())); // <== THIS IS NEW //
  }
}
```

Figure 4.7 Definition of the bilinear form for Example 2.2 in FEniCS (top) and the PUM library (bottom).

While the extension concerns all steps (FFC 1) to (FFC 6), we only provide a very rough overview of the changes and describe just the most important aspects. For more details, especially in regards to the implementation, we refer to [54].

In principle, the FFC-PUM uses the same operational procedure as the FFC, see Section 4.1.2. In order to keep the implementation of new methods to a minimum, the FFC-PUM makes use of the fact that within the context of the eXtended finite element method, only a small subset of basis functions is enriched and, therefore, needs special treatment. With respect to the code generation this means that special code is required only for a small number of enriched cells which are close to the discontinuity. All non-enriched cells away from the discontinuity can be considered with the methods introduced for the conventional finite element method without any modifications. Consequently, one of the first steps performed within the FFC-PUM is to analyze the given UFL input and separate all expressions into standard parts that can be processed using the FFC methods and enriched parts which require the development and use of new concepts.

These new concepts need to consider that all methods related to enriched approximation spaces such as variational forms, finite element spaces, mapping degrees of freedom, and quadrature now depend on information only known during run-time. As mentioned, this data is passed

Algorithm 1 Computing local tensors for cell integrals and integrals over intersection segments [54, Modified version of Algorithm 1]

Input: current cell, coefficients
Output: local tensor

```
1: procedure COMPUTE AND MAP STANDARD AND ENRICHED ENTRIES TO THE LOCAL ELEMENT TENSOR FOR A GIVEN CELL
2:     tabulate values of the basis functions and/or their derivatives
3:     compute and map standard entries by using standard quadrature routines
4:     if no basis function is enriched then
5:         end
6:     else
7:         if given cell is intersected then
8:             compute the modified Gauss quadrature rule on the physical domain
9:         else
10:            map the standard Gauss quadrature rule to the physical domain
11:        end if
12:        for all quadrature points on given cell do
13:            tabulate values of the basis functions and/or their derivatives
14:            if any coefficient is defined on the test function's discontinuous space then
15:                compute the standard entries of the test/trial function that are
16:                affected by the enriched part of the discontinuous coefficient
17:            end if
18:            compute the enriched entries of test and trial functions affected by the
19:            enriched part of the discontinuous coefficient
20:        end for
21:        if given cell is affected by interface integral (*dc) then
22:            compute the modified Gauss quadrature rule for the intersection part
23:            on the physical domain
24:            for all quadrature points on the intersection part do
25:                tabulate values of the basis functions and/or their derivatives
26:                compute the enriched entries
27:            end for
28:        end if
29:        tabulate values into local tensor with minimum dimension
30:    end if
31: end procedure
```

using the interface objects `GenericPUM` and `GenericSurface`. These objects, which are implemented using the concept of inheritance in the `C++` library `DOLFIN-PUM` that is described in the next section, contain all information that concern the enrichment and the location of the discontinuity. In order to make the objects and their content accessible for all routines generated by the `FFC-PUM`, they are passed to the generated code within the definition of the forms, see Figure 4.7.

We illustrate the use of the interface objects and the issue of generating code that is sufficiently generic by considering the generated routines used for computing volume integrals on elements, see Algorithm 1. As described in Section 4.1.3, the `DOLFIN` assembling routines use the generated code to locally perform quadrature on each element individually before the computed values are combined within a global tensor such as the system matrix or the right-hand-side vector. When using the eXtended finite element method, the quadrature rules depend on the discontinuity and the enrichment that are only known at run-time. For the automated code to work, the routines therefore have to be sufficiently general to be able to consider the possible enrichment (and non-enrichment) of basis functions. Due to this, all routines generated with the `FFC-PUM` depend on restricting all data locally for each element and initializing the corresponding quantities with the maximum number of enrichments that may arise within the computation.

4.2.1.3 The `DOLFIN-PUM` library

Consisting of `C++` object oriented classes, `DOLFIN-PUM` implements the routines required by the eXtended finite element method. Essentially, the library consists of

(`PUM` 1) the mentioned interface layer between the XFEM implementation and the code generated by the `FFC-PUM` to transfer the data related to enriched degrees of freedom to the generated code,

(`PUM` 2) methods to represent a discontinuity by the zero level set of a given scalar function,

(`PUM` 3) the (actual) XFEM implementation, that consists of, among others, methods for enriching basis functions, introducing additional degrees of freedom, evaluating enriched functions as well as creating and managing data for enriched elements, degrees of freedom, their mapping, and the corresponding (modified) quadrature rules, and

(`PUM` 4) many other routines containing helper functions to compute subtriangulations, volumes, and allowing for plotting discontinuities and results.

By overloading classes and objects already implemented in `DOLFIN`, the `DOLFIN-PUM` library makes use of `DOLFIN`'s modular structure and allows for using the implemented interfaces to all external libraries and toolboxes such as `PETSc` and `VTK`. The basis of the `DOLFIN-PUM` library are the previously mentioned interface objects `GenericSurface` and `GenericPUM` (`PUM` 1). Both are implemented as abstract bases classes with purely virtual member functions that are implemented in derived subclasses. In regards to the `GenericSurface` object, all related methods, such as routines for computing intersection points or determining the sign of a point with respect to a given indicator function, are provided by a `Surface` class (`PUM` 2). The (abstract) `GenericSurface` object is passed to the `GenericPUM` class whose subclass `PUM` introduces an enrichment that is based on the location of the discontinuity. Moreover, this class addresses (`PUM` 3) and implements the management of data related to the enriched basis functions and degrees of freedom, the definition of enriched approximation spaces and the evaluation of corresponding quantities. For this purpose, the methods mentioned in (`PUM` 4) are used.

4.2.2 Drawbacks and missing features of the PUM library

By enhancing the FEniCS framework, the PUM library in principle allows for an easy and rapid implementation of models for problems with discontinuities. Unfortunately, the library is only applicable to a very limited class of problems due to several issues which, essentially, prevent its application to multiphysics problems.

The first and most obvious drawback of the PUM library is that the framework cannot handle multiple-enriched basis functions. Hence, it is not possible to consider problems with multiple junctions, where interfaces meet or intersect each other. Instead, a sufficiently large distance between discontinuities containing at least one non-enriched element is required. This also prevents the application of the toolbox to problems with moving or evolving discontinuities.

Another at least equally fundamental issue of the toolbox's design is the fact that all methods generated by FFC-PUM can only take into account the enrichment of the test function's approximation space. For a correct evaluation and computation of quantities, this prevents processing data that lives in a different approximation space. Instead, it restricts us to use exactly one enriched function space for approximating all quantities used in a variational form. On first glance, this primarily has an effect on the performance as we may have to use an unnecessarily high polynomial degree for the approximation of quantities such as the piecewise constant coefficient κ in Example 2.2. On second glance, this issue results in many more limitations since it suppresses that data may depend on a different time step which, in problems with evolving subdomains, means that it relies on a different interface position. Another consequence is that it is not possible to project or interpolate data from one enriched function space onto another.

Last but not least, the PUM library does not provide a concept to identify discontinuities and, hence, lacks the possibility for the imposition of different conditions on different discontinuities.

4.3 miXFEM - a multiple interfaces eXtended finite element method based on hierarchical enrichment

For a framework to be suited for the rapid implementation and solution of multiphysics problems, we need not only to meet the conditions (XFEM 1) to (XFEM 4) but the following more increased requirements:

(Req. 1) As multiphysics problems may consists of several subdomains and, hence, discontinuities, it is mandatory that the framework can process basis functions that are enriched by multiple discontinuities.

(Req. 2) Terms in variational formulations may consist of any combination of functions and coefficients that live in different (conventional or enriched) function spaces. Any numerical method has to accurately consider these terms, e.g., in regards to quadrature.

(Req. 3) For each discontinuity, it has to be possible to impose different conditions. Therefore, all discontinuities need to be represented individually and must be uniquely identifiable. For imposing boundary and interface conditions, we may need operators such as $\{\}$ and $\langle\rangle$, cf. Definition 3.3.

(Req. 4) Discontinuities may move or evolve in time and, hence, intersect each other. As a consequence, we need methods that can handle all possible intersections of interfaces with elements or with other interfaces. Moreover, the approach has to be robust with respect to the position of the intersection segment of interface and element.

(Req. 5) Mandatory methods such as interpolation onto all possible (enriched) function spaces as well as approaches for modifying quadrature rules for all possible situations have to be provided.

(Req. 6) Ideally, the framework should also allow a rapid implementation of problems from different areas and provide methods for the analysis of the convergence behavior and numerical errors.

The hierarchical eXtended finite element method presented in Chapter 3 is a flexible method that meets our requirements from a theoretical point of view. In regards to its implementation, the method offers a clear structure and is, by construction, well suited to be implemented in a framework using automated code generation, whose advantages have already been illustrated at length in the previous sections.

4.3.1 Design and implementation details of the toolbox `miXFEM`

Notwithstanding the mentioned drawbacks, the `PUM` library's design approach of introducing interface objects and, hence, an additional layer of abstraction between automatically generated code and XFEM implementation is very thought-out. In combination with its implementation of various basic methods and concepts, the `PUM` library therefore provides an interesting starting point to implement the hierarchical eXtended finite element method into a general framework. This framework, which we call `miXFEM` (multiple interface eXtended finite element method), uses essentially the same design approach as the `PUM` library, which means it is based on extending the core functionalities of `FEniCS` by

(Appr. 1) reinterpreting concepts already implemented in `UFL` to introduce multiple enriched function spaces and quantities,

(Appr. 2) implementing an extended form compiler, capable of interpreting the abstractions correctly, to automatically generate problem related code, and

(Appr. 3) providing an extensive `C++` framework making use of the generated code and providing a full-featured interface to third-party libraries to allow for a rapid implementation of various types of problems.

However, to meet all requirements (Req. 1) to (Req. 6) and, thus, to provide a framework for the rapid solution of multiphysics problems involving an arbitrary number of possibly evolving discontinuities, we need to heavily enhance the `PUM` library and add a lot of additional concepts and methods. In the Sections 4.3.1.1 to 4.3.1.3, we briefly point out the most important changes and extensions of the core components of `FEniCS` and the `PUM` library. In the subsequent Section 4.3.2, we provide more details on the actual implementation of methods for the hierarchical eXtended finite element method. Since in `miXFEM` all discontinuities are described by zero levels of level set functions whose evolution is part of the solution in many multiphysics problems, we additionally implemented a full-featured level set toolbox. This toolbox described in Section 4.4.1 is associated with `miXFEM` and provides methods for maintaining the interfaces during the solution process. As this thesis is motivated by applications involving melting and solidification, we also comment on deriving a non-material velocity field in Section 4.4.2.

4.3.1.1 Variational formulation of multiphysics problems in `UFL`

Recall that in mathematical terms, an extended approximation space V_h, constructed by extending a conventional finite element space V^{FEM} via hierarchical Heaviside enrichment as presented in Section 3.3.1, is given by

$$V_h := V^{\text{FEM}} \bigoplus_{i=1,\dots,N_{\text{dom}}-1} V^{\Gamma_i}, \tag{4.3}$$

cf. equation (3.37), where V^{Γ_i} is the span of all basis functions enriched by Γ_i. To define multiphysics problems with various discontinuities in `UFL`, we further abstract the concepts of `RestrictedElement` and `EnrichedElement` similar to the approach taken by the `PUM`. Starting with a conventional finite element space, we define subsets of this space by recursively restricting it using the measure `dc`. Therein, each restriction corresponds to one discontinuity and one hierarchy level and the enriched approximation space V_h is constructed by adding all restricted elements together. The procedure is exemplary shown for $V^{\text{FEM}} = V^1_{\text{cg},h}$ with three enrichments in the following `UFL` code:

```
CG1_c = FiniteElement("CG", cell, 1)
CG1_0 = RestrictedElement(CG1_c, dc)
CG1_1 = RestrictedElement(CG1_0, dc)
DG0_c = FiniteElement("DG", cell,01)
DG0_0 = RestrictedElement(DG0_c, dc)
DG0_1 = RestrictedElement(DG0_0, dc)
# Enriched spaces
CG1_e = CG1_c + CG1_0 + CG1_1
DG0_e = DG0_c + DG0_0 + DG0_1
```

Using the function spaces defined by this means, we can define test and trial functions as well as coefficients as before, i.e.

```
u = TrialFunction(CG1_e)
v = TestFunction(CG1_e)
k = Coefficient(DG0_e)
[...]
```

and use them to define the problem at hand in variational formulation using the standard UFL language. Thereby we meet the requirements (Req. 1) and (Req. 2) on the UFL level.

In order to comply with (Req. 3), we reuse the subdomain identification method provided by FEniCS for specifying different volumes and facets so that discontinuities can be uniquely identified within the variational formulation such as

```
a = [...]*dx + [...]*dc(0) + [...]*dc(1)
```

however, we have to make sense of this numbers in our problem implementation, see Section 4.3.2.3.

Since we want to impose conditions at interfaces and boundaries which are unfitted to the computational mesh by using Nitsche's method, we additionally introduce the operators as defined in Definition 3.3. While the jump operator is already defined[‡], we can easily add the weighted average `wavg` and the cross-over average `cavg` by

```
subvolume = Coefficient(DG0_e)
def wavg(v):
  return (subvolume('+')*v('+') + subvolume('-')*v('-'))/(subvolume('+') +
   subvolume('-'))
def cavg(v):
  return (subvolume('-')*v('+') + subvolume('+')*v('-'))/(subvolume('+') +
   subvolume('-'))
```

Therein, the `subvolume` on each side is defined accordingly to equation (3.62), which for an element $S \in \mathcal{S}_h$ intersected by Γ_{i_0,l_0} is given by

$$w_{i_0} = \frac{|S \cap \Omega_{i_0}|}{|S \cap (\Omega_{i_0} \cup \Omega_{l_0})|} \qquad \text{resp.} \qquad w_{l_0} = \frac{|S \cap \Omega_{l_0}|}{|S \cap (\Omega_{i_0} \cup \Omega_{l_0})|}. \tag{4.4}$$

Please note that just as in the framework of the PUM library, all requirements (Req. 1) to (Req. 3) are up to now only met on the abstract UFL level. For tackling problems involving multiple discontinuities with different interface conditions these abstractions not only need to be interpreted by a form compiler, but we also have to implement a C++ framework providing the actual implementation.

[‡]In UFL the $[\![\cdot]\!]$ operator is already implemented as `u('+') - u('-')`. However due to choice in Definition 3.3, we may have to either introduce our own variant or alter the sign in some cases.

4.3.1.2 `miXFFC` - A `FEniCS Form Complier` to consider multiphysics problems with discontinuities

All abstractions made within `UFL` have to be interpreted correctly by a form compiler. As our approach is based on an extended use of `UFL`'s concepts and methods, we implemented a new form compiler called `miXFFC`, which is a reworked and significantly extended version of the `FFC-PUM`. While it follows the same workflow as the original `FFC` and `FFC-PUM`, several new concept and significant extensions of the `FFC-PUM` are necessary to meet (Req. 1) to (Req. 6). In particular, we now have to consider variational forms involving quantities living in different approximation spaces which are multiple enriched by hierarchically ordered discontinuities. Moreover, each interface may require different conditions that have to be imposed. As a result, the code generated by `miXFFC` depends even more on information that is not available at the stage when the code is generated.

This is also reflected in the most important design difference of the `miXFFC` in comparison to the `FFC-PUM`: In `miXFFC`, we introduce individual interface objects `GenericMIXFEM` and `GenericLevelSets` for each quantity used within the variational forms. These interface objects contain all relevant information concerning the eXtended approximation space, such as the hierarchy levels and locations of the discontinuities. All interface objects corresponding to quantities of a form are then incorporated into a new structure that is integrated as an abstract object in the implementation generated by `miXFFC`, see Figure 4.8. In contrast to the `FFC-PUM`, where the generated code only receives the interface objects of the test function's approximation space, we now provide the data for all quantities to all methods requiring information regarding the location of the discontinuities, hierarchy, and the chosen enriched approximation space[§].

In the following, we briefly address the most important concepts and extensions of `miXFFC` that mostly rely on these interface objects. Please note that some concepts rely on additional methods that are implemented in `miXDOLFIN` which is presented in the next section. Moreover, we want to stress that many routines generated by `miXFFC` operate locally on given mesh entities, just as it is the case when using the `FFC` or the `FFC-PUM`. As a result, we have to implement local-to-global (and vice-versa) mapping routines.

Hierarchical enrichment: As described in Section 4.3.1.1, we now can define variational forms that involve multiple enriched quantities on the `UFL` level. Unfortunately, we cannot extract much information aside from the maximum number of possibly arising discontinuities from the `UFL` input when generating the code with `miXFFC`. Hence, all generated routines and structures have to be sufficiently general to process up to $N_{\text{dom}} - 1$ enrichments. Consequently, all local tensors and structures of the automatically generated code are of maximum size which

[§]Since we use Nitsche's method to impose interface and boundary conditions, we also have more coefficients than before, see `line 7`.

```
// PUM LIBRARY
class Form_0: public dolfin::Form
{
  public:
  // Constructor
  Form_0(const pum::FunctionSpace& V0, const pum::FunctionSpace& V1):
  dolfin::Form(2, 2), k(*this, 0), w(*this, 1)
  {
    _function_spaces[0] = reference_to_no_delete_pointer(V0);
    _function_spaces[1] = reference_to_no_delete_pointer(V1);

    _ufc_form = boost::shared_ptr<const ufc::form>(new
   pum_form_0(V0.pum_objects()));
  }
}
```

```
// MIXFEM TOOLBOX
class Form_0: public dolfin::Form
{
  public:
  // Constructor
  Form_0(const mixfem::FunctionSpace& V0, const mixfem::FunctionSpace& V1):
  dolfin::Form(2, 4), k(*this, 0), nitsche(*this, 1), phi_0(*this, 2),
   subvolume(*this, 3)
  {
    _function_spaces[0] = reference_to_no_delete_pointer(V0);
    _function_spaces[1] = reference_to_no_delete_pointer(V1);

    _ufc_form = boost::shared_ptr<const ufc::form>(new
   mixfem_form_0(V0.mixfem_objects(), this->coefficients_ptr())); // <==
   THIS IS NEW //
  }
  // THE FOLLOWING IS NEW //
  /// Return a pointer to the coefficients vector of the dolfin::form
  const std::vector<boost::shared_ptr<const dolfin::GenericFunction> >*
   coefficients_ptr() const
  {
    return &_coefficients;
  }
}
```

Figure 4.8 Definition of the bilinear form for Example 2.2 in the `PUM` library (top) and `miXFEM` (bottom).

is given by[¶]

$$N_{\text{loc}} = N_S \times N_{\text{dom}}. \tag{4.5}$$

with N_S denoting the number of standard basis functions of the conventional function space with support on any element $S \in \mathcal{S}_h$. In our implementation, the local representation of an enriched function $u_h \in V_h$ on $S \in \mathcal{S}_h$ is therefore hierarchically ordered and given by[‖]

$$u_h|_S = \boldsymbol{u}_h(S) \cdot \boldsymbol{v}_h(S)|_S, \tag{4.6}$$

[¶]Please note that, in addition to the $N_{\text{dom}} - 1$ enrichments, we also have to take into account the number of conventional basis functions, such that it is $N_{\text{loc}} = N_S \times N_{\text{dom}} - 1 + N_S$.

[‖]In practical examples, most of the values in these structures for an element $S \in \mathcal{S}_h$ are zero since only a very few discontinuities meet in one element.

with

$$\boldsymbol{u}_h(S) = [\underbrace{u_1, \ldots, u_{N_S}}_{\text{std. coefficients}}, \underbrace{u_{1,1}, \ldots, u_{1,N_S}}_{\text{coefficients for } \Gamma_1}, \ldots, \underbrace{u_{N_{\text{dom}}-1,1}, \ldots, u_{N_{\text{dom}}-1,N_S}}_{\text{coefficients for } \Gamma_{N_{\text{dom}}-1}}]^T \tag{4.7}$$

and

$$\boldsymbol{v}_h(S)|_S = [\underbrace{v_1, \ldots, v_{N_S}}_{\text{std basis functions}}, \underbrace{v_{1,1}, \ldots, v_{1,N_S}}_{\text{basis functions for } \Gamma_1}, \ldots, \underbrace{v_{N_{\text{dom}}-1,1}, \ldots, v_{N_{\text{dom}}-1,N_S}|}_{\text{basis functions for } \Gamma_{N_{\text{dom}}-1}}]^T|_S. \tag{4.8}$$

Introducing this order provides us with a simple way to identify the index ranges that correspond to a given hierarchy level.

Algebraic decomposition of enriched quantities: The introduced hierarchical order of enriched basis functions and the degrees of freedom has several advantages. For one, it allows us to easily identify the index range containing the basis functions and coefficients belonging to a given hierarchy level. This can be used to efficiently evaluate jumps or similar expressions. Moreover, it provides us with a simple way to algebraically decompose each quantity into its hierarchy levels which is important for the evaluation of quantities and the computation of terms involving different (enriched) approximation spaces.

Evaluation of coefficients and processing terms involving different (enriched) approximation spaces: In many situations, we need to process terms involving quantities that live in different (enriched) approximation spaces, cf. (Req. 2). One example for such a situation is the quadrature method. In principle, there are two reasons why the approximation spaces of two quantities differ, that is either

(Case 1) the type or the polynomial degree of the conventional finite element space that is enriched is different, or

(Case 2) the locations of the discontinuities, and hence the enrichment, differ.

While (Case 1) naturally occurs in almost every problem, for example, in terms such as $\int_{\Omega_h} \kappa \nabla u \nabla v \, \mathrm{d}x$, (Case 2) is a result of the time discretization of terms using Rothe's method and is therefore related to (Req. 4). Using the same notation as in Section 3.3.3, the time derivative of a quantity $u(\cdot, t)$ (which of course has to be sufficiently smooth in time) can be discretized by a finite difference approximation $\partial_t u \approx \frac{u^{n+1}-u^n}{\Delta t}$. Shifting the part u^n to the right hand side and introducing the variational formulation results in terms of type $\int_{\Omega^{n+1}} u^n v^{n+1} \, \mathrm{d}x$, where v^{n+1} is the test function considering the new interface position(s).

Irrespective of the reason which cause approximation spaces to differ, by using the interface objects of each quantity, we can provide all data required for evaluating quantities with respect to all approximation spaces that are of interest. By adapting the quadrature rules according to the interface positions and the polynomial degree of the quantities' finite element approximation, we are able to accurately compute terms involving quantities of any (enriched) approximation space. The modified quadrature rules and the subdivision of elements are implemented in `miXDOLFIN` and addressed in more detail in Section 4.3.2.

Interpolation and projection of functions: Since functions living in an enriched approximation space usually do not fulfill the Kronecker-δ property[**], implementing nodal interpolation schemes needs additional effort. In contrast to this, the L^2 approximation $u_h \in V_h$ of a function f given by

$$\int_\Omega u_h v_h \,\mathrm{d}x = \int_\Omega f v_h \,\mathrm{d}x, \quad v_h \in V_h, \tag{4.9}$$

can be easily computed at first glance since it is just an additional variational form that is to be defined within `UFL` and can be generated using a form compiler. Unfortunately, `FEniCS`, as well as the `PUM` library, always performs an implicit nodal interpolation of the given function when assembling the right-hand-side. Thereby all information regarding the discontinuous feature are dropped.

For functions $f \in W_h$, where W_h is any (enriched or conventional) finite element function space, `miXFEM` can correctly assemble the right-hand-side of equation (4.9) as this is included in (Case 1) described in the previous paragraph. In order to also consider situations where f is analytically defined, `miXFFC` generates an alternate routine to assemble the right-hand-side, wherein the provided function f is simply evaluated accordingly to the enrichment of V_h.

Integrals over discontinuities: To consider integrals over discontinuities and impose (individual) interface or boundary conditions, see (Req. 3), we face several challenges. Recalling that $\Gamma_i(t) := \bigcup_{\tilde{l}>i} \Gamma_{i,\tilde{l}}(t)$, cf. (Cond. 3) in Section 3.2.2, each zero level of an indicator function may consist of several domain boundaries and interfaces. Thus, we first introduce a *domain id mapping* concept to uniquely identify all (parts of) discontinuities.

This concept provides a mapping between the indices used within the subdomain identification method for discontinuities on the `UFL` level, described in Section 4.3.1.1, and the adjacent subdomains and the affected hierarchy levels. Since it is, for the most part, implemented in `miXDOLFIN` and therefore explained in more detail in the next section, the mapping data is passed via the interface objects to the generated code. Based on the hierarchy levels involved, we can easily identify the basis functions and degrees of freedom that are of interest for computing values that are e.g. required by the operators $\{\cdot\}$, $\langle\cdot\rangle$ and $[\![\cdot]\!]$, see Definition 3.3. Altogether, we are able to meet (Req. 3) by this extension.

While these extensions affect several parts of the generated code, they are especially important for computing and evaluating enriched quantities. Therefore, we close this section by pointing out the differences in the quadrature routines of `miXFEM`, presented in Algorithm 2, compared to the quadrature method as implemented in the `PUM` library, see Algorithm 1. Aside from allowing basis functions to be multi-enriched with respect to different hierarchy levels, the introduction

[**]In fact, the Heaviside enrichment used in this thesis can be shifted so that the Kronecker-δ property is retained. However, we want to present a more general approach that can be naturally used in the `FEniCS` framework.

Algorithm 2 Computing local tensors for cell integrals and integrals over interface segments with `miXFEM`

Input: current cell, coefficients (including list of corresponding `GenericMIXFEM` objects)
Output: local tensor

```
1:  procedure COMPUTE AND MAP STANDARD AND ENRICHED ENTRIES TO THE LOCAL ELEMENT TENSOR FOR A GIVEN CELL
2:      tabulate values of the basis functions and/or their derivatives
3:      compute and map standard entries by using standard quadrature routines
4:      if no basis function is enriched then
5:          end
6:      else
7:          if given cell is intersected then
8:              compute the modified Gauss quadrature rule on the physical domain
9:          else
10:             map the standard Gauss quadrature rule to the physical domain
11:         end if
12:         for all quadrature points on given cell do
13:             tabulate values of the basis functions and/or their derivatives
14:             compute the (standard) entries of all coefficients
15:             for all hierarchy levels $> 0$ (with 0 denoting the non-enriched level) do
16:                 for all coefficients do
17:                     if curr. coefficient is enriched by curr. hierarchy level on given cell then
18:                         determine the enriched entries of the coefficient with respect to
19:                         their own enrichment corresponding to current hierarchy level
20:                     end if
21:                 end for
22:             end for
23:             compute the enriched entries of tensor with respect to the determined coefficients
24:         end for
25:         for all interface integrals (*dc($i$) do
26:             if given cell is affected by curr. interface integral then
27:                 get hierarchy level and indices of adjacent domains
28:                 determine affected index ranges
29:                 compute the modified Gauss quadrature rule for the intersection part
30:                 on the physical domain
31:                 for all quadrature points on the intersection part do
32:                     tabulate values of the basis functions and/or their derivatives with respect
33:                     to the determined index ranges and compute the enriched entries
34:                 end for
35:             end if
36:         end for
37:         tabulate values into local tensor with minimum dimension
38:     end if
39: end procedure
```

of individual interface objects for each coefficient is reflected in the code lines `17-20` for volume integrals and in the lines `28` and `31-34` for interface integrals in Algorithm 2. In contrast to the `PUM` library, the coefficients of a variational form are considered with respect to the enrichment of their own eXtended approximation space, not the space associated with the test or trial function, cf. the lines `14-19` and `25-26` in Algorithm 1. While these seem to be minor changes, they significantly broaden the scope of the methods as we are now able to process terms involving quantities that are defined on different (eXtended) approximation spaces.

4.3.1.3 `miXDOLFIN` library

The main part of the implementation of the hierarchical eXtended finite element method is done in a `C++` framework called `miXDOLFIN`. Just as within the `PUM` library, `miXDOLFIN` consists of several classes providing

(`miXDOLFIN` 1) methods to use the interface objects `GenericMIXFEM` and `GenericLevelSets` to pass information between the problem solving environment and the code generated by `miXFFC` to transfer the data related all enriched basis functions and degrees of freedom to the automatically generated code,

(`miXDOLFIN` 2) methods to represent hierarchically defined and ordered discontinuities by the zero level sets Γ_i of given scalar functions φ_i,

(`miXDOLFIN` 3) a mapping to identify interfaces $\Gamma_{i,l} \subseteq \Gamma_i$, $i < l$, so that individual interface conditions can be imposed for each $\Gamma_{i,l}$,

(`miXDOLFIN` 4) the (actual) XFEM implementation for considering multiple discontinuities, consisting of, i.a., methods for enriching basis functions, introducing additional degrees of freedom, evaluating enriched functions as well as creating and managing data for enriched elements, degrees of freedom, their mapping and the corresponding (modified) quadrature rules, and

(`miXDOLFIN` 5) many other routines containing, for example, helper functions to process an arbitrary number of discontinuities for

- computing subtriangulations and subvolumes,
- performing convergence analysis,
- interpolating functions onto eXtended approximation spaces, and for
- plotting discontinuities and results.

4.3.2 Implementation details of `miXDOLFIN`

Since a detailed technical description of the entire toolbox would go beyond the scope of this section and the thesis, we only highlight some aspects of and present details regarding implementation in `miXFEM` in the following paragraphs. A detailed technical description of the entire toolbox will be given in an upcoming publication.

For the presentation, we assume that the level set functions φ_i and their zero levels $\tilde{\Gamma}_i$, $i = 1, \ldots, N_{\text{dom}} - 1$, separating the domain Ω into disjoint subdomains Ω_l, $l = 1, \ldots, N_{\text{dom}}$, are given. While the level set functions and, hence, the subdomains may change in time, we consider a steady-state situation in all upcoming sections for convenience, with the exception of the paragraph addressing the handling of time dependency. In regards to the discretization, we further assume that the following assumptions, see for example [49], hold:

(Assumption 1) The triangulation $\mathcal{S}_h$, $h > 0$, is non-degenerate that means for all $S \in \mathcal{S}_h$ it is $\frac{h_S}{\rho_S} \leq c \in \mathbb{R}$, with ρ_S denoting the diameter of the largest ball contained in the element S.

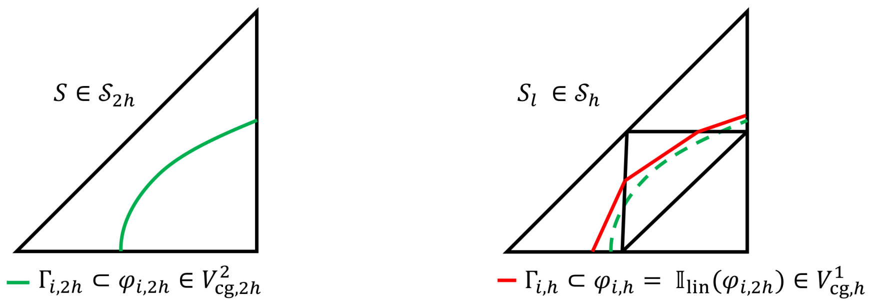
(Assumption 2) The resolution of the triangulation $\mathcal{S}_h$ in the vicinity of each zero level $\tilde{\Gamma}_i$ is sufficiently high so that for each element $S \in \mathcal{S}_h$ with $S \cap \tilde{\Gamma}_i \neq \emptyset$ no element edge is intersected twice.

Both assumptions are very common and not restrictive so that they can be easily satisfied in all scenarios.

4.3.2.1 Discrete interface representation

We begin this section by addressing the discrete representation of Γ_i. Recalling that $\Gamma_i \subseteq \tilde{\Gamma}_i$, with $\tilde{\Gamma}_i$ being the zero level set of φ_i in the continuous setting, discrete approximations $\varphi_{i,h}$ of φ_i and $\tilde{\Gamma}_{i,h}$ of $\tilde{\Gamma}_i$ are needed first and foremost. To avoid the handling of complex intersections of interfaces and elements, especially in regards to multi-junctions, we currently approximate $\tilde{\Gamma}_i$ linearly. Hence, we use $\varphi_{i,h} \in V^1_{\text{cg},h}$ as finite element approximation of φ_i. Unfortunately, the low polynomial degree of the approximation space has turned out to be unfavorable for problems involving terms related to second derivatives of the interface such as curvature. Hence for considering such problems, we use an approach very similar to [29] and introduce a second triangulation $\mathcal{S}_{2h}$ such that $\mathcal{S}_h$ can be obtained by regular refining $\mathcal{S}_{2h}$[††]. The idea is now to approximate φ_i by $\varphi_{i,2h} \in V^2_{\text{cg},2h}$ and define $\varphi_{i,h} = \mathbb{I}_{\text{lin}}(\varphi_{i,2h})$ by linearly interpolating $\varphi_{i,2h}$ on $\mathcal{S}_h$, cf. Figure 4.9. Let $\tilde{\Gamma}_{i,h}$ denote the corresponding zero level set, the discrete and

[††]In practice, one first choose $\mathcal{S}_{2h}$ and then construct $\mathcal{S}_h$.


$S \in \mathcal{S}_{2h}$
$\Gamma_{i,2h} \subset \varphi_{i,2h} \in V^2_{\mathrm{cg},2h}$
$S_l \in \mathcal{S}_h$
$\Gamma_{i,h} \subset \varphi_{i,h} = \mathbb{I}_{\mathrm{lin}}(\varphi_{i,2h}) \in V^1_{\mathrm{cg},h}$

(a) Quadratic approximation $\Gamma_{i,2h}$ of the interface Γ_i on $\mathcal{S}_{2h}$.

(b) Linear approximation $\Gamma_{i,h}$ of the interface Γ_i on $\mathcal{S}_h$.

Figure 4.9 2D visualization of $\Gamma_{i,2h}$ on an element $S' \in \mathcal{S}_{2h}$ and $\Gamma_{i,h} = \mathbb{I}_{\mathrm{lin}}(\varphi_{i,2h})$ on the elements $S_l \in \mathcal{S}_h$, $l = 1, 2, 3, 4$, obtained by regular refining $S' \in \mathcal{S}_{2h}$.

hierarchically active interface $\Gamma_{i,h}$ is then given by

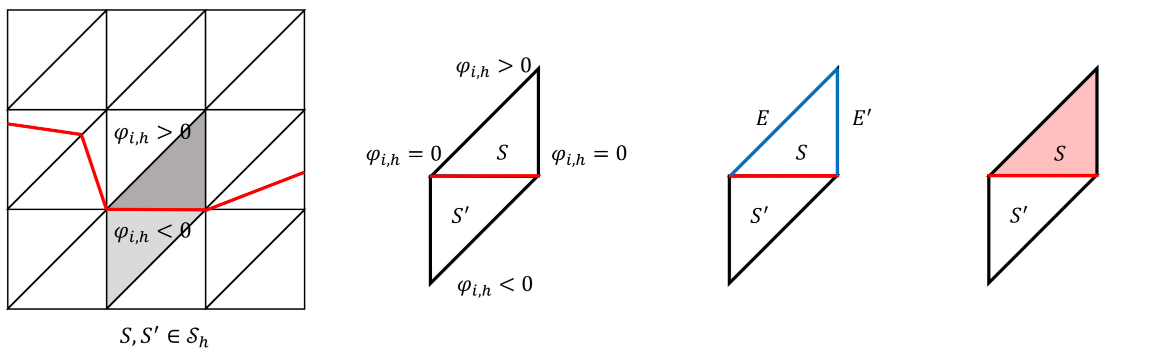
$$\begin{aligned}\Gamma_{i,h} :&= \{\boldsymbol{x} \in \tilde{\Gamma}_{i,h} : \prod_{l=1}^{i-1} H(\varphi_{l,h}(\boldsymbol{x})) = 1\} \\ &= \{\boldsymbol{x} \in \Omega : \varphi_{i,h}(\boldsymbol{x}) = 0 \wedge \prod_{l=1}^{i-1} H(\varphi_{l,h}(\boldsymbol{x})) = 1\},\end{aligned} \tag{4.10}$$

cf. equation (3.33). For practical reasons, we always compute the complete zero levels $\tilde{\Gamma}_{i,h}$ and later define the active parts $\Gamma_{i,h} \subseteq \tilde{\Gamma}_{i,h}$ by evaluating the product of the Heaviside functions as shown in (4.10). The zero levels $\tilde{\Gamma}_{i,h}$ are uniquely given by the linear intersection segments $\tilde{\Gamma}_{i,h,S}$, with $\tilde{\Gamma}_{i,h} := \bigcup_{S \in \mathcal{S}_h} \tilde{\Gamma}_{i,h,S}$, which can be computed by the following two-step algorithm:

(Step 1) First, we construct the set $\mathcal{S}_h^{\tilde{\Gamma}_{i,h}}$ containing all elements intersected by $\tilde{\Gamma}_{i,h}$. For this purpose, we compute the values $z_E = \varphi_{i,h}(\boldsymbol{x}_{E,1})\varphi_{i,h}(\boldsymbol{x}_{E,2})$, where $x_{E,l}$ denotes the l-th vertex of edge $E \in \mathcal{E}(S)$ and $\mathcal{E}(S)$ contains all edges of the simplex $S \in \mathcal{S}_h$. If we have $z_E \leq 0$ and $\varphi_{i,h}(\boldsymbol{x}_{E,l}) > 0$ for some $l \in \{1, 2\}$, we mark the edge E and the element $S \in \mathcal{S}_h$ as intersected and add it to the set $\mathcal{S}_h^{\tilde{\Gamma}_{i,h}}$.

(Step 2) In a second step, we loop over all elements $S \in \mathcal{S}_h^{\tilde{\Gamma}_{i,h}}$ and compute the intersection points which can be easily done since $\tilde{\Gamma}_{i,h}$ is linear.

Please note that (Step 1) allows for an unique assignment of intersection points to edges. While such an assignment is obvious for edges $E \in \mathcal{E}(S)$ with $z_E < 0$, we also need to consider situations with $z_E = 0$, where a zero level set $\tilde{\Gamma}_{i,h}$ directly intersects a vertex or aligns with an element edge or facet so that $\varphi_{i,h}(\boldsymbol{x}_{E,l}) = 0$ for least one $l \in \{1, 2\}$ of the edge E. In order to also have an unique assignment for such situations, we only consider edges E as intersected, where it is $\varphi_{i,h}(\boldsymbol{x}_{E,l}) > 0$ and $\varphi_{i,h}(\boldsymbol{x}_{E,\tilde{l}}) \leq 0$, for l, $\tilde{l} \in \{1, 2\}$ with $\tilde{l} \neq l$, but neglect edges with $\varphi_{i,h}(\boldsymbol{x}_{E,l}) = 0$ and $\varphi_{i,h}(\boldsymbol{x}_{E,\tilde{l}}) < 0$. This approach is visualized in Figure 4.10. Among others, the resulting unique assignment is important for generating a subtriangulation for intersected elements for performing quadrature, as we will see in the next paragraph.



(a) Interface $\Gamma_{i,h}$ aligns with the facet of the elements S and S'.
(b) Values in the endpoints of edges.
(c) Edges E and E' are marked as intersection by our approach.
(d) Our approach marks S as intersected element.

Figure 4.10 Visualization of unique assignment of intersections to element edges for an interface aligning with an element edge (2D).

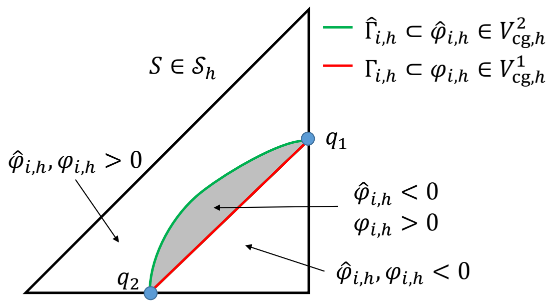
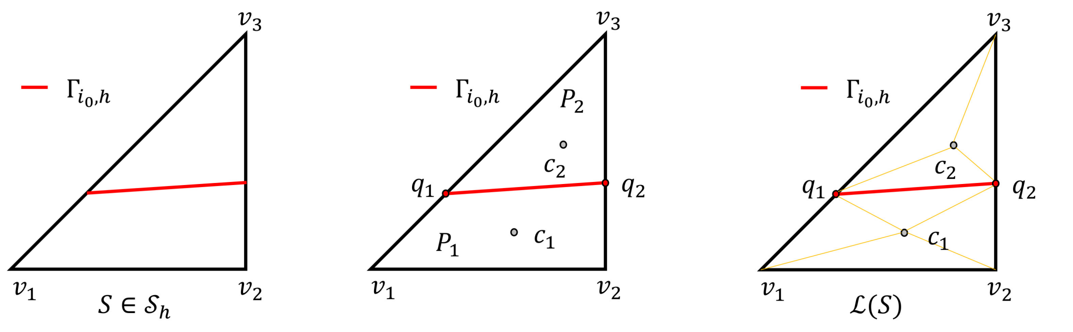

$\hat{\Gamma}_{i,h} \subset \hat{\varphi}_{i,h} \in V^2_{\mathrm{cg},h}$
$\Gamma_{i,h} \subset \varphi_{i,h} \in V^1_{\mathrm{cg},h}$
$S \in \mathcal{S}_h$
q_1
$\hat{\varphi}_{i,h}, \varphi_{i,h} > 0$
$\hat{\varphi}_{i,h} < 0$
$\varphi_{i,h} > 0$
$\hat{\varphi}_{i,h}, \varphi_{i,h} < 0$
q_2

Figure 4.11 2D visualization of the approximation error when constructing a linear representation of a discontinuity using $\varphi_{i,h} \in V^2_{\mathrm{cg},h}$.

Remark 4.1 (Comment on the efficiency). Actually, the idea in [29] is based on introducing a regularly refined mesh $\mathcal{S}_{\tilde{h}/2}$ only for elements $S \in \mathcal{S}_{\tilde{h}}$ intersected by the zero level set of a quadratic function $\varphi_{\tilde{h}}$. On these elements, there is a one-to-one relation between the degrees of freedom $\varphi_{\tilde{h}}$ and its linear approximation $\mathbb{I}_{\mathrm{lin}}(\varphi_{\tilde{h}/2})$ which allows for an efficient computation of the quantities. Alternatively (or as an addition to this), it is much more efficient to introduce a narrow band around the zero level set of each $\varphi_{i,\tilde{h}}$. Then we only have to compute all level set related quantities and maintain each level set function only on these subdomains instead of considering the complete domain Ω. We comment on the narrow band approach in Section 4.4.1.

Remark 4.2 (Conformity between the polynomial degree of $\varphi_{i,h}$ and $\tilde{\Gamma}_{i,h}$). Naively, one could try to approximate φ_i by $\varphi_{i,h} \in V^m_{\mathrm{cg},h}$, $m \geq 2$, and just use the intersection points q_m, $m = 1, \ldots, N_{\mathrm{q}}$, of the function $\varphi_{i,h}$ and the element edges to construct linear intersection segments $\tilde{\Gamma}_{i,h,S}$ and define $\tilde{\Gamma}_{i,h} := \bigcup_{S \in \mathcal{S}_h} \tilde{\Gamma}_{i,h,S}$. However, such a construction leads to an undefined behavior if we try to determine the domain containing $\boldsymbol{x} \in S$ by the evaluation of $\varphi_{i,h}(\boldsymbol{x})$ as the values and, in particular, signs may be not consistent with the constructed $\tilde{\Gamma}_{i,h}$, see Figure 4.11.



(a) Initial state, $S \in \mathcal{S}_h$ is intersected by $\Gamma_{i,h}$.

(b) Steps (Subtr. 1) and (Subtr. 2).

(c) Steps (Subtr. 3) and (Subtr. 4).

Figure 4.12 2D visualization of the scheme for the generation of a subtriangulation.

4.3.2.2 Subtriangulation and quadrature

An important aspect when using eXtended discretization methods is the subdivision of intersected elements $S \in \mathcal{S}_h^{\Gamma_{i,h}}$ to perform quadrature. To illustrate the scheme which extends the idea of [54] and is implemented in `miXFEM` for generating subtriangulations of such elements, we first consider an element $S \in \mathcal{S}_h^{\Gamma_{i_0,h}}$ intersected by only one discrete interface $\Gamma_{i_0,h}$. Afterwards, we comment on how to extend the approach to situations where S is intersected by multiple interfaces. Now, given an intersected element $S \in \mathcal{S}_h^{\Gamma_{i_0,h}}$ and the interface $\Gamma_{i_0,h}$, we perform the following steps to generate a corresponding subtriangulation:

(Subtr. 1) Subdivide an element S into two parts P_1 and P_2 using $\Gamma_{i_0,h,S} \subset \Gamma_{i_0,h}$. The parts P_l are now defined as polygon by the original vertices v_k, $k = 1, \dots, d+1$, of S and the intersection points q_m, $m = 1, \dots, N_\mathrm{q}$.

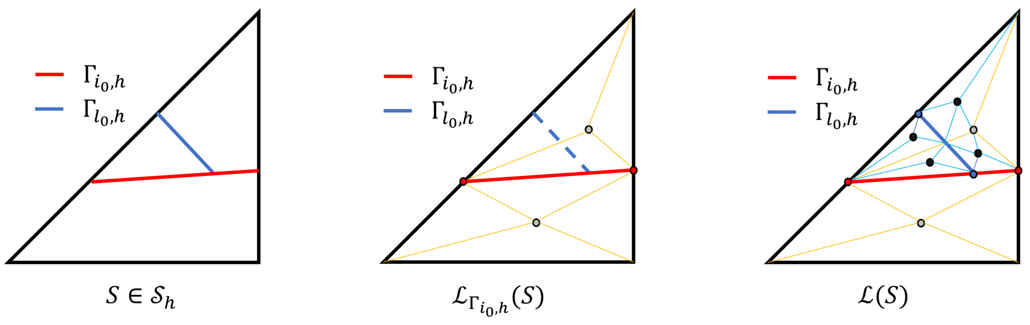
(Subtr. 2) Compute the center of gravity c_l for each part P_l, $l = 1, 2$.

(Subtr. 3) Create simplicial subtriangulation $\mathcal{L}_l(S)$ of the parts P_l by connecting the original vertices v_k, $k = 1, \dots, d+1$, of the simplex S contained in P_l with the intersection points q_m, $m = 1, \dots, N_\mathrm{q}$, and the respective center c_l of gravity.

(Subtr. 4) A local subtriangulation called *local mesh* is then given by $\mathcal{L}(S) = \mathcal{L}_1(P_1) \cup \mathcal{L}_2(P_2)$.

All steps are visualized for a 2D situation in Figure 4.12. The presented subdivision approach can easily be extended to consider multiple-intersected elements by performing the subdivision recursively, starting with the interface of highest hierarchy, cf. Figure 4.13. All following subdivisions then have to be performed respective to the previous generated subelements of the local mesh. The complete procedure is given in Algorithm 3.

Using the generated subtriangulation, we now can perform quadrature by transforming the quadrature schemes onto the subelements. Just as in the conventional finite element method,



(a) Initial state, $S \in \mathcal{S}_h$ is intersected by $\Gamma_{i,h}$ and $\Gamma_{l,h}$.

(b) Perform subdivision scheme for $\Gamma_{i,h}$.

(c) Perform subdivision scheme for $\Gamma_{l,h}$ considering the already generated local mesh.

Figure 4.13 2D visualization of the subtriangulation scheme for the generation of a subtriangulation of an multi-intersected element.

Algorithm 3 Generation of a local mesh for each intersected element

Input: $\Gamma_{i,h}$, $\mathcal{S}_h^{\Gamma_{i,h}}$, $i = 1, \dots, N_{\text{dom}} - 1$
Output: local meshes $\mathcal{L}(S)$, $S \in \mathcal{S}_h^{\Gamma_{i,h}}$

```
1:  procedure Generation of a local mesh for each intersected element
2:      Define $\mathcal{S}_h^{\Gamma} := \bigcup_{i=1,\dots,N_{\text{dom}}-1} \mathcal{S}_h^{\Gamma_{i,h}}$
3:      for all $S \in \mathcal{S}_h^{\Gamma}$ do
4:          initialize local mesh $\mathcal{L}(S)$ by $\mathcal{L}(S) := S$
5:          for all $i = 1, \dots, N_{\text{dom}} - 1$ do
6:              for all $\tilde{S} \in \mathcal{L}(S)$ do
7:                  if $\Gamma_{i,h} \cap \tilde{S} \neq \emptyset$ then
8:                      refine $\tilde{S}$ and generate the local mesh $\mathcal{L}(\tilde{S})$ by performing the
9:                      steps (Subtr. 1) to (Subtr. 4)
10:                 end if
11:             end for
12:             update local mesh $\mathcal{L}(S)$
13:         end for
14:     end for
15: end procedure
```

we can use the usual quadrature scheme and transform the points and weights to the generated subelements. The approach guarantees that the quadrature points of each subdivided element are located in exactly one domain so that functions can be evaluated in these points, however, we have to correctly scale the weights with the volume of the elements $S' \in \mathcal{L}(S)$. The presented scheme for the generation of a local mesh for intersected elements is also important, among others, for evaluating functions, considering time-dependent problems, and plotting the results as we will see in the following sections.

4.3.2.3 Interface id mapping and imposing boundary and interface conditions

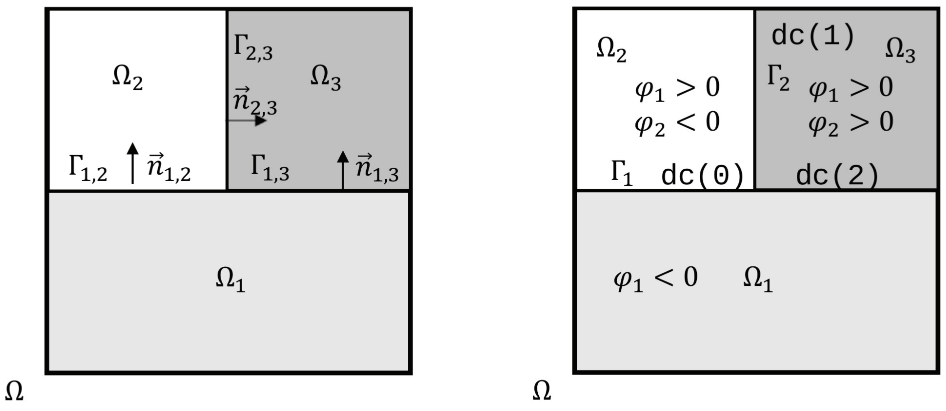
Using the hierarchical level set method to decompose a domain Ω into subdomains, we have $\Gamma_i = \bigcup_{l>i} \Gamma_{i,l}$, see (Cond. 3) in Section 3.2.2, which means that we represent the internal boundaries $\Gamma_{i,l}$, $l > i$, by (parts of) one zero level set $\Gamma_i \subseteq \tilde{\Gamma}_i$, $i = 1, \ldots, N_{\text{dom}} - 1$. As we usually have to impose a different condition on each interface $\Gamma_{i,l} \subseteq \Gamma_i$ when modeling multiphysics problems, we need an approach to (re-)identify the respective parts $\Gamma_{i,l} \subseteq \Gamma_i$ and the adjacent subdomains Ω_i and Ω_l during run-time. In order to do this, a domain id mapping concept has been implemented which affects all components of `miXFEM`.

Given a discrete variational formulation of a problem with several interfaces, the concept consists of two levels. Firstly, we define (in ascending order) `dc(k)` measures, $k \in \mathbb{N}_0$, within the `UFL` file that implements the discrete variational formulation of the problem. The indices of the measures are then passed to the code generated by `miXFFC`. Secondly, we introduce a mapping in the implementation of the problem in `miXDOLFIN` to identify the interfaces for imposing the (correct) interface condition. Therefore, we define a vector and begin with assigning the zero level sets to the interface measures. The following code shows how to assign the zero level set $\Gamma_{i,h}$ to the interface measure `dc(l)`

```
[...]
// create domain_id_map
std::vector<uint > domain_id_map(NumberOfMeasures);
[...]
domain_id_map[l] = i; // Assign zero level set number i to dc measure k
[...]
```

For all hierarchically active zero level sets $\Gamma_{i,h}$ that consists of multiple interfaces $\Gamma_{i,l,h}$ with $\Gamma_{i,h} = \bigcup_{l>i} \Gamma_{i,l,h}$, we additionally use the information provided by the respective level set function $\varphi_{l,h}$ to separate $\Gamma_{i,h}$ into parts with $\varphi_{l,h}(\boldsymbol{x}) < 0$ and $\varphi_{l,h}(\boldsymbol{x}) > 0$, $\boldsymbol{x} \in \Gamma_{i,h}$. The corresponding implementation where an interface $\Gamma_{i,h}$ is separated into two parts $\Gamma_{i,l_0,h}$ and $\Gamma_{i,l_1,h}$ by the level set function $\varphi_{l_0,h}$ is given by

```
[...]
// create domain_id_map
std::vector<uint > domain_id_map(NumberOfDc);
[...]
domain_id_map[l0] = i; // Assign zero level set number i to dc measure l0
domain_id_map[l1] = i; // Assign zero level set number i to dc measure l1
[...]
// create restriction vector
std::vector<std::vector<std::pair<uint,bool> > >
    domain_id_restrictions(NumberOfDc,std::vector<std::pair<uint, bool> >(0));
[...]
domain_id_restrictions[l0].push_back(std::make_pair(i,false));
domain_id_restrictions[l1].push_back(std::make_pair(i,true));
[...]
```



(a) Setting. (b) Level set functions and `dc` measures.

Figure 4.14 Visualization a domain id mapping possibility for Example 3.2.

Therein, we use boolean values to identify the sign of $\varphi_{l,h}(\boldsymbol{x})$ with false corresponding to $\varphi_{l,h}(\boldsymbol{x}) < 0$ and true corresponding to $\varphi_{l,h}(\boldsymbol{x}) > 0$. Both vectors are passed to the code generated by `miXFFC` to impose the correct interface condition. In addition, the concept allows for identifying the adjacent domains which is crucial for the operators $[\![\cdot]\!]$, $\{\cdot\}$, and $\langle\cdot\rangle$ that are required by Nitsche's method.

Remark 4.3 (Simple domain id map). Using the restriction vector and the sign of the level set functions to separate a zero level set $\Gamma_{i,h}$ into its components $\Gamma_{i,l,h}$ is only necessary, if we want to consider a different type of interface condition on parts $\Gamma_{i,l_0,h}, \Gamma_{i,l_1,h} \subset \Gamma_{i,h}$, i.e., Dirichlet conditions on $\Gamma_{i,l_0,h} \subset \Gamma_{i,h}$ and flux/jump conditions on $\Gamma_{i,l_1,h} \subset \Gamma_{i,h}$. Otherwise, we can simply introduce piecewise defined functions to impose different conditions of the same type on all interfaces. However, the use of the domain id mapping concept can improve the readability and understandability of the code.

Example 4.1. *Considering the setting in Example 3.2, the configuration* $\Gamma_1 = \Gamma_{1,2} \cup \Gamma_{1,3}$ *and* $\Gamma_2 = \Gamma_{2,3}$ *can be realized using the domain id mapping concept as depicted in Figure 4.14*[‡‡] *by*

```
std::vector<uint > domain_id_map(3);
domain_id_map[0] = 0; // Assign zero level set number 0 to dc measure 0
domain_id_map[1] = 1; // Assign zero level set number 1 to dc measure 1
domain_id_map[2] = 0; // Assign zero level set number 0 to dc measure 2 =>
    we need to include a restriction:
std::vector<std::vector<std::pair<uint, bool> > > domain_id_restrictions(3,
    std::vector<std::pair<uint, bool> >(0));
// Restrict measure dc(0) to the region with $\varphi_1<0$
domain_id_restrictions[0].push_back(std::make_pair(1,false));
// Restrict measure dc(2) to the region with $\varphi_1>0$
domain_id_restrictions[2].push_back(std::make_pair(1,true));
```

[‡‡]Please note that since `C++` begins numbering with 0 instead of 1, all indices of subdomains and interfaces are shifted so that Γ_1 is denoted by the index 0 in the code snippet.

4.3.2.4 Evolving interfaces

Currently, we discretize time-dependent problems with moving or evolving interfaces by applying Rothe's method, see Section 3.3.3[§§]. Therefore, we first discretize the time derivative by the implicit Euler scheme and then apply spatial discretization. By doing so, two very different issues arise in regards to the implementation: firstly, we need to efficiently redefine the time levels of data and, secondly, we have to process terms involving different time levels in the discrete variational forms, cf. Section 3.3.3.

General implementation approach: For an efficient handling of data, we make extensive use of boost shared pointers when implementing problems in `miXDOLFIN`. Approximating the time derivative with the implicit Euler method, we only have to introduce two versions of a time-dependent object $O(t)$, and "old" version $O(t_n)$ and a "new" version $O(t_{n+1})$. By using and redirecting pointers to these objects in the computational scheme, we only have to create the new data at each time level, as shown in the following code snippet

```
[...]
boost::shared_ptr<myObject> object_new;
boost::shared_ptr<myObject> object_old;
[...]
object_new = boost::make_shared<myObject>(t0);
[...]
while (t < T)
{
  [...]
  object_old = object_new;
  object_new = boost::make_shared<myObject>(t);
  [...]
}
```

Processing terms involving different time levels: Using Rothe's method can yield integrals such as $\int_{\bigcup_{i=1}^{N_{\text{dom}}} \Omega_i(t_{n+1})} f(t_n)v(t_{n+1})\,\mathrm{d}x$. As the positions of $\Gamma_i(t_n)$ and $\Gamma_i(t_{n+1})$ are usually different, we need to evaluate and process the quantities f and v considering different enrichments. For this purpose, we interpret the different locations at the time levels as different interfaces (with corresponding different enrichments), which is a situation which can be naturally handled within our framework and the presented subdivision approach, see Section 4.3.2.2. On these subtriangulation, we can evaluate functions correctly with respect to their enrichment and perform quadrature for the assembling of tensors.

4.3.2.5 Miscellaneous

After highlighting some methods and features of `miXFEM` in some detail, we now briefly describe some other important aspects.

[§§]Of course, we only consider problems (or a scaled variant) which are sufficiently smooth in time.

Managing enriched data: Managing the data corresponding to basis functions and the degrees of freedom is mandatory in any implementation of the finite element method. However, designing such concepts for eXtended discretization methods is complex as the enriched data may change due to the evolution or dissolution of discontinuities. By implementing the hierarchical eXtended finite element method into the automated code generation framework of `FEniCS`, matters become significantly more challenging. Reasons for this are that the compiler knows nothing about the actual discontinuities and enrichment but only the maximum number of possible arising discontinuities and, in addition, almost all methods generated by `miXFFC` operate locally on cells or facets and therefore need to be identified in the global structures. In our approach, we introduce *offsets* to define local and global structures that order quantities based on the hierarchy levels. By doing so, basis functions and coefficients are represented as described in Section 3.3.1 and Section 4.3.1.2. The concept is sufficiently general which means that it can also consider vector-valued elements or combined resp. mixed elements such as the Taylor-Hood element.

Additional assembler: The evaluation of quantities and the assembling of tensors is not straight forward when using an eXtended discretization method since the Kronecker-δ property is usually lost. This also makes the interpolation of functions given in terms of symbolic expressions onto eXtended function spaces more difficult. To take this into account in our framework, several new assembler methods are implemented in `miXFEM` which can pass given functions represented by `dolfin::Expressions` and data corresponding to the enrichment that is summarized to `GenericMIXFEM` objects to the `tabulate_tensor` routines generated by `miXFFC`, cf. Section 4.3.1.2. Using the expressions, we can simply generate approximations of given functions by computing the L^2-projection, cf. Section 4.3.1.2. By passing the `GenericMIXFEM` objects of all coefficients involved in a discrete variational formulation to the generated code, we can evaluate and process terms involving different (enriched) approximation spaces.

Plotting: For the visualization of computed data, we store our results in the `VTK` output format so that they can be processed using programs such as `ParaView` [90]. To correctly visualize discontinuous features within functions, we have to introduce multi-valued vertices. Moreover, the data needs to be adapted to fit into the structures provided by `VTK`. For this we make use of the already generated subtriangulations, see Section 4.3.2.2, which are stored as a local mesh for each intersected element.

Error computation and visualization: The `miXFEM` framework provides methods to compute numerical approximation errors such as $\|u_{\text{ref}} - u_h\|$ which are of interest, for example, when convergence or parameter studies are performed. In general, such error can be easily computed when the reference solution u_{ref} is a known analytical function by projecting it onto the enriched approximation space of the numerical solution and computing the difference. However, more effort is required if u_{ref} is also a computed approximation which has been obtained by, e.g., using a finer mesh or different polynomial degree than the current numerical solution u_h. This is because now we need to compare functions of different approximation spaces where the enrichment data differs. To also compute approximation errors for such situations, we can define $\|u_{\text{ref}} - u_h\|$ as `dolfin::Expressions` and interpret it as a functional which can be computed using one of previously mentioned new designed assemblers. To make the analysis easier, the library `Gnuplot` [91] can be linked to `miXFEM` to automatically plot convergence results.

4.4 Additional numerical methods

Aside from the actual hierarchical eXtended finite element method and its implementation, the numerical solution of multiphysics problems often requires additional methods. One reason for this is that, for example, the evolution of the discontinuities is part of the solution in many multiphysics problems. As mentioned before, the location of discontinuities in `miXFEM` are given by zero levels of signed distance functions and their evolution is taken into account by the level set method. Unfortunately, the level set method introduces additional challenges when used within the finite element method since it neither preserves the signed distance property nor does it conserve the volume. Moreover, additional stabilization techniques are necessary when problems are dominated by an advection term. For these reasons, we implemented a level set toolbox, which can be integrated into `miXFEM`, that provides methods for stabilizing the level set problem, reinitializing a given level set function, and correcting the arising volume defect. For efficiency, we also implemented structures to restrict the level set problem to a small region called *narrow band*. All this is presented in Section 4.4.1

The level set method describes the evolution of function in a given velocity field. In the applications motivating this thesis, we consider problems involving phase changes but not material flow. Hence, the corresponding velocity may result from the Stefan condition [15, 92] and is therefore non-material. As a consequence, we need methods to evaluate the energy fluxes, compute the (normal) interface velocity, and extend this velocity into the vicinity. The approaches used within this thesis are described in Section 4.4.2.

4.4.1 Level set toolbox

While the level set toolbox does not directly belong to `miXFEM`, it provides methods for moving and evolving discontinuities represented by zero levels of level set functions as well as routines for maintaining the level set functions with respect to the signed distance property and volume conservation. The level set toolbox, which has been implemented in collaboration with former master student T. Klock, has been presented in more detail in [2, 93]. At this point, we sum up the most important concepts that are adapted from [29, 94, 95] but omit detailed implementation aspects.

Since all level set functions can be considered independent from each other, we simplify the notation and describe all methods using only one level set function φ with zero level Γ for the presentation. Please keep in mind that this function can be any φ_i, $i = 1, \dots, N_{\text{dom}} - 1$. Moreover, we use the notation $\tilde{h}$ instead of h to describe the mesh size. This is because of the possible difference in the mesh size used for solving the level set and the actual problem, see Section 4.3.2.1.

4.4.1.1 Discretization

Recall that the evolution of an interface separating two domains can be described by the level set method as shown in Section 2.1. Then, the strong formulation of the problem reads: Given a domain Ω and a time interval $[t_0, t_f]$, find $\varphi \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$ such that

$$\frac{\partial \varphi}{\partial t} + \vec{V} \cdot \nabla \varphi = 0 \qquad \text{in } \Omega \times [t_0, t_f], \tag{4.11}$$

$$\varphi(\cdot, t_0) = \varphi_0 \qquad \text{in } \Omega, \tag{4.12}$$

hold. As usual, the problem is transfered into a variational formulation which can be discretized by either variant of the method of lines.

Using Rothe's method, the time interval $[t_0, t_f]$ is as before discretized by $N_t + 1$ time steps $t_n = t_0 + n\Delta t$, $n = 0, \dots, N_t$, with Δt denoting the time step size. Let $\theta \in [0, 1]$ be a parameter[‖] and $\varphi^n(\cdot) \approx \varphi(\cdot, t_n)$ be an approximation of the level set function φ at time t_n. The full discrete level set problem on a triangulation $\mathcal{S}_{\tilde{h}}$ using the θ-scheme for time discretization then reads: Find $\varphi_{\tilde{h}}^{n+1} \in V_{\text{cg},\tilde{h}}^m$, with $m \geq 1$, such that

$$\sum_{S \in \mathcal{S}_{\tilde{h}}} \left(\frac{\varphi_{\tilde{h}}^{n+1} - \varphi_{\tilde{h}}^n}{\Delta t} + \theta \vec{V}^{n+1} \cdot \nabla \varphi_{\tilde{h}}^{n+1} + (1 - \theta) \vec{V}^n \cdot \nabla \varphi_{\tilde{h}}^n, v_{\tilde{h}} \right)_{L^2(S)} = 0, \tag{4.13}$$

holds for all $v_{\tilde{h}} \in V_{\text{cg},\tilde{h}}^m$. In our implementation and, hence, the following methods, we choose $m = 2$ and, in most cases, $\tilde{h} = 2h$ with h denoting the mesh size for the actual problem. The interface is then represented as described in Section 4.3.2.1.

4.4.1.2 Stabilization

It is well known that solving hyperbolic PDEs with standard finite element methods can be unstable, especially for high velocities $\vec{V}$. This issue can be resolved by using a stabilization method [96], which slightly reformulates the discretized problem. A well known method for this is the *Streamline-Upwind/Petrov-Galerkin* (SUPG) stabilization [97] which is also called *Streamline diffusion* stabilization [29].

[‖]Note that $\theta = 0$ leads to the explicit Euler-scheme while $\theta = 1$ results in the implicit Euler-scheme.

As proposed in [29], special test functions $\tilde{v}_{\tilde{h}} \in L^2(\Omega)$ of the form

$$\tilde{v}_{\tilde{h}}|_S := v_{\tilde{h}} + \delta_S \vec{V}^{n+1} \cdot \nabla v_{\tilde{h}}, \quad S \in \mathcal{S}_{\tilde{h}},\ v_{\tilde{h}} \in V^m_{\mathrm{cg},\tilde{h}}$$

with $\delta_S \in [0,1]$, can be used for stabilization so that equation (4.13) reads

$$\sum_{S \in \mathcal{S}_{\tilde{h}}} \left(\frac{\varphi^{n+1}_{\tilde{h}} - \varphi^n_{\tilde{h}}}{\Delta t} + \theta \vec{V}^{n+1} \nabla \varphi^{n+1}_{\tilde{h}} + (1-\theta)\vec{V}^n \nabla \varphi^n_{\tilde{h}}, v_{\tilde{h}} + \delta_S \vec{V}^{n+1} \cdot \nabla v_{\tilde{h}} \right)_{L^2(S)} = 0. \tag{4.14}$$

In literature, it is suggested to use a δ_S that depends on the velocity $\vec{V}$ and the diameter of the simplex $\tilde{h}_S = \mathrm{diam}(S)$, for $S \in \mathcal{S}_{\tilde{h}}$,

$$\delta_S = c \frac{\tilde{h}_S}{\max\{\delta_0, ||\vec{V}^{n+1}||_{\infty,S}\}}, \tag{4.15}$$

with $0 < \delta_0 \ll 1$ and $c \in [0,1]$ with $c = 0$ corresponding to the case that no stabilization is applied.

4.4.1.3 Reinitialization

From a numerical point of view, it is beneficial to have a level set function $\varphi_{\tilde{h}}$ which is close to a signed distance function, see Section 2.1. Unfortunately, this property may be lost during the evolution of the level set function in time due to various reasons, such as discretization errors, insufficient approximation of the curvature, and topological changes. To regain the signed distance property, the level set function is reinitialized with a variant [29] of the Fast Marching Method (FMM) [98]. This approach provides a signed distance approximation $\tilde{\varphi}_{\tilde{h}}$ of $\varphi_{\tilde{h}}$.

Given $\varphi_{\tilde{h}} \in V^2_{\mathrm{cg},\tilde{h}}$ on $\mathcal{S}_{\tilde{h}}$, we first compute the linear interpolation $\varphi_{\tilde{h}/2} = \mathbb{I}_{\mathrm{lin}}(\varphi_{\tilde{h}})$ of $\varphi_{\tilde{h}}$ on the regularly refined triangulation $\mathcal{S}_{\tilde{h}/2}$. Let $\mathcal{V}(S)$ denote the set of vertices given on a simplex $S \in \mathcal{S}_{\tilde{h}/2}$ and $\mathcal{V} := \mathcal{V}(\mathcal{S}_{\tilde{h}/2})$ be the (discrete) set of all vertices of $\mathcal{S}_{\tilde{h}/2}$[**]. The patch of order $l+1$, $l \geq 1$, of elements related to a vertex $v \in \mathcal{V}(S)$, $S \in \mathcal{S}_{\tilde{h}/2}$ is given by

$$\mathcal{P}^{l+1}(v) := \{S \in \mathcal{S}_{\tilde{h}/2} : \mathcal{V}(S) \cup \mathcal{V}(\mathcal{P}^l(v)) \neq \emptyset\} \quad \text{with} \quad \mathcal{P}^1(v) := \{S \in \mathcal{S}_{\tilde{h}/2} : v \in \mathcal{V}(S)\}. \tag{4.16}$$

The basic idea of the FMM, which consists of two phases, is to compute distance values $\hat{d}(v)$ for any vertex $v \in \mathcal{V}$ thereby taking advantage of the fact that all information will only propagate outwards from the zero level set of the function.

[**]Please note that by choosing a linear Lagrangian basis, vertices and degrees of freedom coincide.

Initialization phase: Firstly, distance values $\hat{d}(v)$ for $v \in \mathcal{V}(\mathcal{S}^{\Gamma}_{\tilde{h}/2})$ are computed by

$$\hat{d}(v) = \min_{S \in \mathcal{P}^2(v)} \operatorname{dist}(v, \Gamma_{\tilde{h}/2,S}),$$

where the extended patch $\mathcal{P}^2(v)$ including also all second neighbor simplices is considered for stability reasons, see [29].

Extension phase: The initial computed values are propagated into the far field. Therefore, the finished set

$$\mathcal{V}_F = \{v \in \mathcal{V}(\mathcal{S}_{\tilde{h}/2}) \,:\, \hat{d}(v) \text{ is computed}\},$$

is introduced which stores already processed vertices. Additionally, an active set is defined by

$$\mathcal{V}_A = \{v \in \mathcal{V}(\mathcal{S}_{\tilde{h}/2}) \,:\, \text{At least one neighbor vertex of } v \text{ is in } \mathcal{V}_F\}$$

which stores vertices that are likely to be dealt with next since they are neighbors of already processed vertices. The process of choosing the vertex $v \in \mathcal{V}_A$ which is considered next is based on a tentative distance function

$$\tilde{d}(v) = \min\{\tilde{d}_S(v) \,:\, S \in \mathcal{P}^1(v) \text{ with } \mathcal{V}(S) \cap \mathcal{V}_F \neq \emptyset\}$$

with $\tilde{d}_S(v)$ defined as

$$\tilde{d}_S(v) = \hat{d}(P_{W(S)}(v)) + \|v - P_{W(S)}(v)\|.$$

Therein,

$$P_{W(S)}(v) = \operatorname{argmin}_{x \in \operatorname{conv}(W(S))} \|v - x\|.$$

is the minimum distance projection of v onto the convex hull $W(S) = \mathcal{V}(S) \cap \mathcal{V}_F$. With this construction, $\tilde{d}_S(v)$ approximates the distance of v to the discrete interface $\Gamma_{\tilde{h}}$ by using the simplex S (which has at least one vertex in $\mathcal{V}_F$) as an information propagator. From these values, the minimum value for $\tilde{d}_S(v)$ is used as the tentative distance value if all processed vertices/simplices are considered as information propagators. Once values $\tilde{d}(v)$ can be computed, the extension phase works by extracting the current nearest vertex $v^* = \operatorname{argmin}_{v \in \mathcal{V}_A} \tilde{d}(v)$, setting $\hat{d}(v^*) = \tilde{d}(v^*)$, removing this vertex from the active set $\mathcal{V}_A$ and adding it to the finished set $\mathcal{V}_F$. Then the active set $\mathcal{V}_A$ and the tentative values $\tilde{d}(v)$, $v \in \mathcal{V}_A$ are updated according to the updated finished set and the procedure is repeated until all vertices are processed.

As the result of the FMM, we obtain unsigned distance values $\tilde{d}(v)$ for every vertex v of the refined triangulation $\mathcal{S}_{\tilde{h}/2}$. These uniquely define a reinitialized piecewise quadratic level set function $\tilde{\varphi}_{\tilde{h}} \in V^2_{\mathrm{cg},\tilde{h}}$ due to the one-to-one relation between vertices in $\mathcal{S}_{\tilde{h}/2}$ and degrees of freedom in $V^2_{\mathrm{cg},\tilde{h}}$. Note that the values $\tilde{d}$ are unsigned, hence a multiplication with the correct sign is necessary before using them as new coefficients of $\tilde{\varphi}_{\tilde{h}}$.

4.4.1.4 Volume correction

As analyzed in [99], the level set method does not preserve the volume, although the loss in volume decreases with decreasing mesh size. The same is true for the reinitialization method. As a consequence, a volume correction method is needed to overcome this issue. This method can be used, for example, after the initialization phase of the FMM in which the new (unsigned) distance values $\hat{d}(v)$ for $v \in \mathcal{V}(\mathcal{S}^{\Gamma}_{\tilde{h}/2})$ are computed. In general, there are two possibilities for such methods, a global volume correction method or a local approach to preserve the volume. In the following, we describe the more elaborated local approach, which is an adapted version of the localized correction algorithm presented in [95], and comment on how to implement a global method afterwards. For a more detailed description, we refer to [95] and [2].

Consider the linear level set functions $\varphi_{\tilde{h}/2} = \mathbb{I}_{\text{lin}}(\varphi_{\tilde{h}})$ and its reinitialized version $\tilde{\varphi}_{\tilde{h}/2} = \mathbb{I}_{\text{lin}}(\tilde{\varphi}_{\tilde{h}})$ on $\mathcal{S}_{\tilde{h}/2}$. For arbitrary functions ϕ, ψ, we define the volume functional

$$\Delta V(\phi_{\tilde{h}}, \psi_{\tilde{h}}, S) = \int\limits_{S \cap \{x \in \Omega_{\tilde{h}} \,:\, \phi < 0\}} dx - \int\limits_{S \cap \{x \in \Omega_{\tilde{h}} \,:\, \psi < 0\}} dx, \quad S \in \mathcal{S}_{\tilde{h}/2}.$$

Let $\Omega_{\Gamma_{\tilde{h}}} = \bigcup_{S \in S^{\Gamma_{\tilde{h}}}_{\tilde{h}/2}} S$ be the domain of all intersected simplices and $V^1_{\text{cg},\tilde{h}/2}(\Omega_{\Gamma_{\tilde{h}}})$ be the corresponding function space. Assuming that the values $\hat{d}(v)$, computed in the initialization phase of the FMM, define a tentative level set function $\tilde{\phi}^{\text{ten}} \in V^1_{\text{cg},\tilde{h}/2}(\Omega_{\Gamma_{\tilde{h}}})$, we adjust the values of this function in the following four steps to preserve the volume:

(Step 1) Compute an offset $c_S \in \mathbb{R}$ for every $S \in \mathcal{S}^{\Gamma_{\tilde{h}}}_{\tilde{h}/2}$ [††] such that

$$\Delta V\left(\mathbb{I}_{\text{lin}}(\varphi_{\tilde{h}}), \tilde{\phi}^{\text{ten}}(\cdot) + c_S, S\right) = 0$$

holds. Thereby the addition $\tilde{\phi}^{\text{ten}}(\cdot) + c_S$ is understood as an addition in the degrees of freedom, i.e. c_S is added to every coefficient of $\tilde{\phi}^{\text{ten}}$.

(Step 2) Compute a continuous, piecewise linear offset function $\phi^{\text{corr}} \in V^1_{\text{cg},\tilde{h}/2}(\Omega_{\Gamma_{\tilde{h}}})$ by averaging values c_S on $\mathcal{P}^1(v)$ so that a value $\phi^{\text{corr}}(v)$ for v connected to $S \in \mathcal{S}^{\Gamma}_{\tilde{h}/2}$ is given by

$$\phi^{\text{corr}}(v) := \frac{1}{|S \in \mathcal{P}^1(v) \cap \mathcal{S}^{\Gamma_{\tilde{h}}}_{\tilde{h}/2}|} \sum_{S \in \mathcal{P}^1(v) \cap \mathcal{S}^{\Gamma_{\tilde{h}}}_{\tilde{h}/2}} c_S.$$

(Step 3) Compute a global multiplier $m_\Omega \in \mathbb{R}$ so that

$$\Delta V\left(\mathbb{I}_{\text{lin}}(\varphi_{\tilde{h}}), \tilde{\phi}^{\text{ten}}(\cdot) + m_\Omega \phi^{\text{corr}}(\cdot), \Omega_{\tilde{h}}\right) = 0.$$

Since this is the second optimization step, the resulting m_Ω is usually close to 1.

[††] In principle, this corresponds to computing a function $\Psi \in V^0_{\text{dg},\tilde{h}/2}$.

(Step 4) Adjust the $\hat{d}(v)$ values for vertices that are processed in the FMM initialization phase by

$$\hat{d}(v) := \begin{cases} \hat{d}(v) - m_\Omega \phi^{\text{corr}}(v), & \text{for } v \in \{x \in \Omega_{\tilde{h}} \,:\, \mathbb{I}_{\text{lin}}(\varphi_{\tilde{h}}) < 0\}, \\ \hat{d}(v) + m_\Omega \phi^{\text{corr}}(v), & \text{for } v \in \{x \in \Omega_{\tilde{h}} \,:\, \mathbb{I}_{\text{lin}}(\varphi_{\tilde{h}}) > 0\}, \end{cases}$$

and proceed to the extension phase of the FMM with these modified $\hat{d}(v)$ values.

The roots in steps 1 and 3 can be computed by using the regula falsi algorithm in the Anderson/Björk variant [100] as shown in [2].

Remark 4.4 (Global approach for volume conservation). A simpler approach to correct the volume defect is based on computing only a global offset $c_\Omega \in \mathbb{R}$ in (Step 1). Using this value, we can easily define $\phi^{\text{corr}} \in V^1_{\text{cg},\tilde{h}/2}(\Omega_{\Gamma_{\tilde{h}}})$ and proceed with (Step 4). Unfortunately, numerical studies show that this global approach tends to also preserves the shape of the geometry which is undesirable in many applications.

4.4.1.5 Narrow band approach

A major drawback of the level set method as an interface representation technique is that a higher dimensional object (the level set function) is used to represent a lower dimensional object (the interface). To overcome this drawback, the narrow band level set method [94] can be used whose basic idea is to restrict the task of solving the level set problem, the reinitialization, and the volume correction to a narrow band around the current interface.

Construction of the narrow band(s): The narrow band method is based on the assumption that the given discrete level set function $\varphi_{\tilde{h}}$ is approximately a signed distance function so that the values of the degrees of freedom are approximately equal to the exact distance of a vertex to the interface. Then, the *inner* narrow band is defined by

$$\mathcal{V}_{\text{INB}} = \{v \in \mathcal{V}(\mathcal{S}_{\tilde{h}}) \;:\; \varphi_{\tilde{h}}(v) < \gamma_{\tilde{h}}\}, \tag{4.17}$$

with $\gamma_{\tilde{h}} = \gamma\tilde{h}$, $\gamma \in \mathbb{Z}^+$, and $\tilde{h} = \max_{S \in \mathcal{S}_{\tilde{h}}} \text{diam}(S)$, and the *outer* narrow band is given by

$$\mathcal{V}_{\text{ONB}} = \mathcal{V}_{\text{INB}} \cup \left(\bigcup_{v \in \mathcal{V}_{\text{INB}}} \bigcup_{S \in \mathcal{P}^1(v)} \mathcal{V}(S) \right). \tag{4.18}$$

The outer narrow band includes all vertices of the inner narrow band set as well as all vertices of the first neighbor patch of all simplices in the inner narrow band domain. With these set, the corresponding domains $\Omega_{\text{INB}} := \{S \in \mathcal{S}_{\tilde{h}} \,:\, \mathcal{V}(S) \subset \mathcal{V}_{\text{INB}}\}$ and $\Omega_{\text{NB}} = \Omega_{\text{ONB}} := \{S \in \mathcal{S}_{\tilde{h}} \,:\, \mathcal{V}(S) \subset \mathcal{V}_{\text{ONB}}\}$ can be defined.

Introduction of a cut-off function: Since solving the original level set problem on the narrow band often leads to oscillations at the narrow band boundaries, cf. [94], a discrete cut-off function

$$\zeta_{\tilde{h}}(\varphi_{\tilde{h}}) = \zeta_{\tilde{h}}(\varphi_{\tilde{h}}(\boldsymbol{x},t)) = \begin{cases} 1 & \text{for } \left|\varphi_{\tilde{h}}(\boldsymbol{x},t)\right| \leq \beta_{I,\tilde{h}}, \\ \hat{\zeta}_{\tilde{h}}(\varphi_{\tilde{h}}, \beta_{I,\tilde{h}}, \beta_{O,\tilde{h}}) & \text{for } \beta_{I,\tilde{h}} < \left|\varphi_{\tilde{h}}(\boldsymbol{x},t)\right| \leq \beta_{O,\tilde{h}}, \\ 0 & \text{for } \left|\varphi_{\tilde{h}}(\boldsymbol{x},t)\right| > \beta_{O,\tilde{h}}, \end{cases} \tag{4.19}$$

with

$$\hat{\zeta}_{\tilde{h}} = \left(\varphi_{\tilde{h}}(\boldsymbol{x},t) - \beta_{O,\tilde{h}}\right)^2 \frac{2\varphi_{\tilde{h}}(\boldsymbol{x},t) + \beta_{O,\tilde{h}} - 3\beta_{I,\tilde{h}}}{(\beta_{O,\tilde{h}} - \beta_{I,\tilde{h}})^3} \tag{4.20}$$

is introduced, which slowly decreases the influence of the advection term towards these boundaries. Therein, the parameters $\beta_{I,\tilde{h}} = \beta_I \tilde{h}$ and $\beta_{O,\tilde{h}} = \beta_O \tilde{h}$ with $\beta_I < \beta_O < \gamma$ divide the inner narrow band layer into three sublayers. A viable choice for these parameters is for example $\beta_I = 2,\ \beta_O = 4,\ \gamma = 6$, see [94].

Adapted discretized problem: Based on these definitions, the concept of the narrow band level set method is now to solve the adapted, completely discretized level set equation

$$\begin{aligned} &\sum_{S \in \mathcal{S}_{\tilde{h}}} \left(\frac{\varphi_{\tilde{h}}^{n+1} - \varphi_{\tilde{h}}^{n}}{\Delta t} + \theta \zeta_{\tilde{h}}^{n+1} \vec{V}^{n+1} \cdot \nabla \varphi_{\tilde{h}}^{n+1} + (1-\theta)\zeta_{\tilde{h}}^{n} \vec{V}^{n} \cdot \nabla \varphi_{\tilde{h}}^{n}, v_{\tilde{h}} + \delta_S \vec{V}^{n+1} \cdot \nabla v_{\tilde{h}} \right)_{L^2(S)} \\ &\quad = 0, \quad \forall v_{\tilde{h}} \in V^2_{\mathrm{cg},\tilde{h}}, \quad S \in \Omega_{\mathrm{INB}}, \end{aligned} \tag{4.21}$$

reinitialize the solution on Ω_{ONB}, and extend the function with a constant value $\pm(\gamma_{\tilde{h}} + \varepsilon)$ outside of Ω_{ONB}, $\varepsilon \in \mathbb{R}^+$. Please note that in equation (4.21), we treat (4.19) explicitly with respect to time by defining $\zeta_{\tilde{h}}^{n+1} = \zeta(\varphi_{\tilde{h}}^{n})$ to avoid the task of solving a non-linear equation. For doing so, the innermost narrow band layer width $\beta_I \tilde{h}$ has to be sufficiently large.

CFL conditions: When using the narrow band level set method, one has to consider two Courant–Friedrichs–Lewy (CFL) conditions:

- Since ζ decreases the transport of the level set function outside the innermost band, we have to guarantee that the interface $\Gamma_{\tilde{h}}$ does not leave this region. Therefore, the CFL condition
$$\Delta t \| \|\vec{V}^{n+1}\|_2 \|_{L_\infty(\Omega_{\mathrm{INB}})} < \beta_I \tilde{h}, \quad \forall n \in \{0, \dots, N_t - 1\} \tag{4.22}$$
must hold.

- For the method to work, we need to ensure $\Omega_{\mathrm{INB}}^{n+1} \subset \Omega_{\mathrm{ONB}}^{n}$ and, hence, that $\varphi_{\tilde{h}}$ is close to a signed distance function on $\Omega_{\mathrm{ONB}}^{n}$. If the velocity transporting the interface and the time step size are too large compared to the mesh size, we may end up considering the constant values $\pm(\gamma_{\tilde{h}} + \varepsilon)$ during the solution and reinitialization process. To avoid this, the condition
$$\Delta t \| \|\vec{V}^{n+1}\|_2 \|_{L_\infty(\Omega_{\mathrm{INB}})} < \tilde{h}, \quad \forall n \in \{0, \dots, N_t - 1\}. \tag{4.23}$$
has to hold.

Remark 4.5. A typical parameter choice includes $\beta_I > 1$ so that (4.22) is automatically fulfilled, if (4.23) holds, making this the limiting condition. Please also note that even though the method's description assumes the reinitialization procedure to be applied after *every* time step, it might be better to apply reinitialization and update the narrow band after every m-th time step instead. This results in a more restrictive CFL condition given by

$$m\Delta t \| \|\vec{V}^{n+1}\|_2 \|_{L_\infty(\Omega_{\mathrm{INB}})} < \tilde{h}, \quad \forall n \in \{0, \ldots, N_t - 1\}. \tag{4.24}$$

4.4.2 Construction of a non-material velocity field

The solution of the level set problem for the function φ_i is based on knowing its evolution velocity $\vec{V}_i$. In general, there are two situations possible, first, that the velocity $\vec{V}_i$ is material, meaning that particles move with the given velocity as it is, for example, the case when considering fluid dynamics. Alternatively, the velocity can be non-material, which occur in problems involving phase transitions. While using a material velocity for the evolution of an interface is straightforward, non-material velocities have to be computed by other quantities and, moreover, are usually only locally defined. As a consequence, an extension of the computed velocity field onto the whole domain is required. Since this thesis is motivated by considering melting and solidification processes which can be modeled by the Stefan condition

$$[\![\lambda \nabla u]\!] \cdot \vec{n}_i = L\vec{V}_i \cdot \vec{n}_i \quad \text{on } \Gamma_i, \tag{4.25}$$

where u denotes the temperature, λ the thermal conductivity and L the latent heat, which is the energy that is released or absorbed during the phase change, we focus on this condition for the presentation of an approach to obtain a global velocity field based on the energy fluxes at an interface. For ease of notation, we reduce the setting to a two-phase scenario where a level set function $\varphi(\cdot, t)$ with zero level set $\Gamma(t)$ separates a domain Ω into the subdomains $\Omega_1(t)$ (with $\varphi(\cdot, t) < 0$) and $\Omega_2(t)$ (with $\varphi(\cdot, t) > 0$).

Our approach, published in [2, 93] and summarized here, computes the velocity $\vec{V}^n \in \left(V^1_{\mathrm{cg},\tilde{h}}\right)^d$ for $n = 0, \ldots, N_t - 1$ in two steps: In the first step, the Stefan condition (4.25) is evaluated to compute the normal velocity at the interface. In a second step, this velocity is extended to the whole narrow band, making this approach very similar to the previously presented Fast Marching Method. However, please note that the velocity field is not calculated on the regularly refined mesh since we do not need a piecewise quadratic velocity function.

4.4.2.1 Initialization phase

We begin with computing the projections w_j, $j \in \mathcal{N}_{\mathrm{P}}$, of all $v \in \mathcal{V}(\mathcal{S}^{\Gamma}_{\tilde{h}})$ onto the discrete interface $\Gamma_{\tilde{h}}$ such that $\|v - w_j\|_2 = \mathrm{dist}(v, \Gamma_{\tilde{h}})$ holds. As there can be multiple points that satisfy the

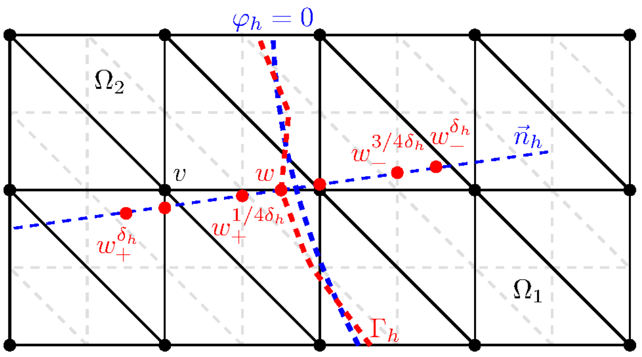


Figure 4.15 Evaluation points of in the DSCE velocity computation method [2].

minimum distance requirement, we may have multiple projections w_j. Now, we present two approaches that can be used to compute the corresponding values $\vec{V}^n_{\Gamma_{\tilde{h}}}(v)$ for $v \in \mathcal{V}(S),\ S \in \mathcal{S}^{\Gamma}_{\tilde{h}}$:

(DGE) Direct gradient evaluation: We compute the discrete temperature gradient $\nabla u^n_{\tilde{h}}$, which is a piecewise constant, vector-valued XFEM function, and the velocity field at the projections w_j *directly* using

$$\left(\vec{V}^n_{\Gamma_{\tilde{h}}}(w_j)\right)_k = \left(\nabla(u^n_{1,\tilde{h}})(w_j)\right)_k - \left(\nabla(u^n_{2,\tilde{h}})(w_j)\right)_k,$$

with index $k = 1, \ldots, d$ denoting the respective component. The velocity vector at v is then defined by averaging over all contributions $\left(\vec{V}^n_{\Gamma_{\tilde{h}}}(w_j)\right)_k$ of all projections w_j that are found in the previous step.

(DSCE) Discretized Stefan condition evaluation [101]: For every projection $w \in \{w_j : j \in \mathcal{N}_{\mathrm{P}}\}$, we use point-value tuples

$$\left(w_-^{\frac{l}{4}\delta_{\tilde{h}}}, u^n_{1,\tilde{h}}\left(w_-^{\frac{l}{4}\delta_{\tilde{h}}}\right)\right) \qquad \text{and} \qquad \left(w_+^{\frac{l}{4}\delta_{\tilde{h}}}, u^n_{2,\tilde{h}}\left(w_+^{\frac{l}{4}\delta_{\tilde{h}}}\right)\right), \qquad l = 0, \ldots, 4,$$

with

$$w_{+,-}^{r\delta_{\tilde{h}}} = w \pm r\delta_{\tilde{h}}\vec{n}_{\tilde{h}}(w)$$

and $\delta_{\tilde{h}} = \delta\tilde{h}_{\max}$ a step-width parameter to perform a linear least-squares regression through these five points on each separate side, cf. Figure 4.15. The slope of these regressions is taken to approximate the gradient in normal direction at w and the resulting normal velocity is given as

$$(\vec{V}^n_{\Gamma_{\tilde{h}}} \cdot \vec{n}_{\tilde{h}})(w) = \frac{2}{L}\left[\frac{\lambda_1}{5}\frac{2u^n_{1,\tilde{h}}(w) + u^n_{1,\tilde{h}}\left(w_-^{\frac{1}{4}\delta_{\tilde{h}}}\right) - u^n_{1,\tilde{h}}\left(w_-^{\frac{3}{4}\delta_{\tilde{h}}}\right) - 2u^n_{1,\tilde{h}}\left(w_-^{\delta_{\tilde{h}}}\right)}{\delta_{\tilde{h}}}\right.$$
$$\left.+\frac{\lambda_2}{5}\frac{2u^n_{2,\tilde{h}}(w) + u^n_{2,\tilde{h}}\left(w_+^{\frac{1}{4}\delta_{\tilde{h}}}\right) - u^n_{2,\tilde{h}}\left(w_+^{\frac{3}{4}\delta_{\tilde{h}}}\right) - 2u^n_{2,\tilde{h}}\left(w_+^{\delta_{\tilde{h}}}\right)}{\delta_{\tilde{h}}}\right].$$

By multiplying with $\vec{n}_{\tilde{h}}$, a velocity field $\vec{V}^n_{\Gamma_{\tilde{h}}}(w) = (\vec{V}^n_{\Gamma_{\tilde{h}}} \cdot \vec{n}_{\tilde{h}})(w) \cdot \vec{n}_{\tilde{h}}(w)$ can be obtained from this expression. At the end, we set the velocity field at v to the average of all contributions from the projections $w \in \{w_j : j \in \mathcal{N}_\mathrm{P}\}$.

4.4.2.2 Extension phase

To propagate the initialized velocity values into the far field, the idea of the already presented FMM algorithm is used. Since this algorithm effectively propagates (distance) values into the far field in the reinitialization, we can basically step through the same procedure and propagate velocity values instead of distance values. In detail, we perform the following two steps:

(Vel. step 1) Calculate distance values for vertices $v \in \mathcal{V}(S)$, $S \in \mathcal{S}^\Gamma_{\tilde{h}}$ so that initialized distance and velocity field values are given after this step.

(Vel. step 2) Propagate the distance values into the far field through the FMM extension phase and additionally propagate the velocity values. If a vertex v^* regarding $\hat{d}$ in the distance propagation is processed, we set the velocity to

$$\vec{V}^n_{\Gamma_{\tilde{h}}}(v^*) = \vec{V}^n_{\Gamma_{\tilde{h}}}\left(P_{W(S_\mathrm{min})}(v^*)\right)$$

where the value $\vec{V}^n\left(P_{W(S_\mathrm{min})}(v^*)\right)$ is again calculated through the barycentric coordinates of the projection $P_{W(S_\mathrm{min}}(v^*)$ and already known velocity values.

Remark 4.6. Note that although (2) resembles what we do in the reinitialization process, there is actually a minor difference that needs to be considered: there can be multiple minimizing simplices S_min that minimize the current $\tilde{d}(v^*)$ function for the vertex that is processed next. While this does not matter in the reinitialization, since the distance value coincide for all these minimizing simplices, the velocity values can differ. In this case, we need to set the velocity to be the average of all contributions, i.e.,

$$\vec{V}^n_{\Gamma_{\tilde{h}}}(v^*) = \frac{1}{|\mathcal{S}_\mathrm{min}|}\sum_{S\in\mathcal{S}_\mathrm{min}} \vec{V}^n_{\Gamma_{\tilde{h}}}\left(P_{W(S)}(v^*)\right),$$

where $\mathcal{S}_\mathrm{min}$ captures all simplices that minimize the tentative distance function. Also note that there is no such thing as a tentative velocity function in this procedure. The vertex that is processed next is still defined by the vertex that minimizes the tentative distance function $\tilde{d}$ on the current active set $\mathcal{V}_A$. In other words, the velocity field values is information that is propagated alongside but does not interfere with the FMM procedure itself.

Chapter 5

Numerical results and applications

The presented hierarchical eXtended finite element method and its implementation into the framework `miXFEM` have been validated using several examples, see, i.a., [2, 93, 102–104]. Moreover, `miXFEM` has been used for the numerical simulation of several applications with industrial background, see, for example, [105, 106]. In the following sections, we briefly summarizing some validation results and then focus on the modeling and simulation of the applications presented in Section 1.1 which motivate this thesis.

Due to its importance for solving complex coupled problems, we first consider the level set toolbox, including the reinitialization and volume correction techniques, in Section 5.1. Because of the extent of the performed numerical studies, we only give an overview of the considered examples and present a summary of numerical tests that have been performed to validate the implementation. Afterwards, we focus on the hierarchical eXtended finite element method. The modeling of problems involving multiple subdomains, where several elements are multi-enriched, is considered in Section 5.2. For this example, we also briefly take a look on the approximation quality and convergence order of the hierarchical eXtended finite element method. Due to the lack of analytically known solutions of multiphase Stefan problems, we validate our approach for this kind of problem for different two-phase scenarios in Section 5.3. There, we again present several examples and summarize some validation results before we take a quick look on the convergence behavior of the methods by considering one example.

After considering the components of `miXFEM` and the validation examples, we focus on the (reduced) multiphysics problems that are motivated by the industrial applications already mentioned in Section 1.1. The laser welding process for the generation of hybrid joints is modeled and simulated in Section 5.4 and we use this process to demonstrate `miXFEM`'s capabilities to handle changes in the topology and techniques to handle new arising boundaries. Section 5.5 focuses on the thermal upsetting process where we, in addition to topological changes, have to consider an evolving geometry in a hold-all domain. Within this section, the enhanced modeling potential when using the hierarchical eXtended finite element method due to the decoupling of mesh and physical domain is illustrated. Afterwards, the keyhole-based laser welding process is considered in much detail in Section 5.6. Therein, we again highlight the advantages and flexibility of the hierarchical eXtended finite element method to model complex multiphysics problems. Using an analytical approach to determine the keyhole geometry a priori, the presented method and `miXFEM` are used to model and simulate the process. The numerical results are then compared to experimental data generated at the Bremer Institut für angewandte Strahltechnik. Finally, we briefly consider a multiphase flow problem to illustrate the generality of the developed method and the framework `miXFEM`.

5.1 The level set method

When using eXtended discretization methods, interfaces and physical domain boundaries are usually represented using the level set method, see Section 2.1. This is especially true when the evolution of boundaries or interfaces is part of the solution, which makes the level set toolbox crucial for tackling multiphysics problems. Therefore, we first address the validation of the algorithms implemented into the level set toolbox. There are various aspects implemented within the toolbox which require intensive testing:

(Tests 1) the actual level set solver,

(Tests 2) the SUPG stabilization,

(Tests 3) the reinitialization method,

(Tests 4) all volume correction approaches, and

(Tests 5) the narrow band approach and the corresponding adapted level set problem.

Moreover, we have to perform thorough parameter studies to quantify the influence of

(Study 1) the SUPG stabilization,

(Study 2) the narrow band regions (respectively the values of β_I, β_O, and γ),

(Study 3) the reinitialization method and the reinitialization frequency,

(Study 4) the different volume correction approaches, and

(Study 5) all possible combinations of the methods

on the solution process and the solution's accuracy.

However, since the level set method is not directly a part of the hierarchical eXtended finite element method on which this thesis focuses, we omit a detailed presentation of our parameter studies but refer to our corresponding publications [2, 93]. Instead, we only mention the considered examples in Section 5.1.1, comment on the simulation setup and computational approach in Section 5.1.2, and briefly summarize the obtained results in Section 5.1.3.

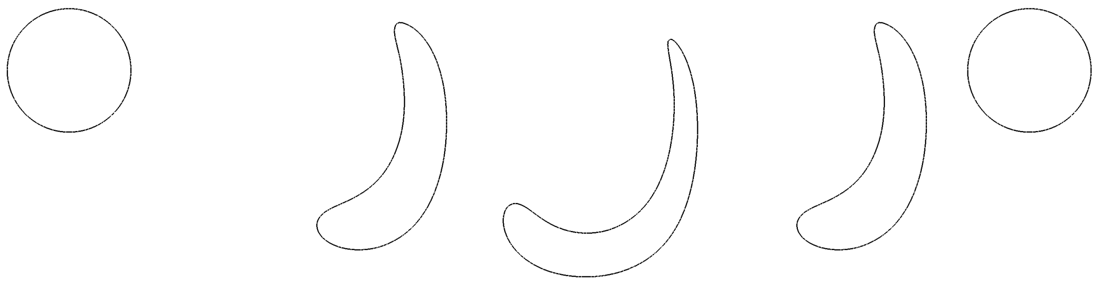


Figure 5.1 2D example swirling flow vortex: Zero level set $\Gamma_h(t)$ of the reference solution $\varphi_h(\boldsymbol{x}, t)$ for $t = 0,\ t = 0.5,\ t = 1,\ t = 1.5$ and $t = 2$.

5.1.1 Examples

For the validation of the methods described in Section 4.4.1 and their implementation, detailed studies have been performed for, among other, the following two examples in both, 2D and 3D. The characteristic of all mentioned examples is that the zero level set $\Gamma(t_0)$ of $\varphi(\boldsymbol{x}, \boldsymbol{c}, t_0)$ is a sphere with center of gravity in $\boldsymbol{c}$ that is deformed for $t \in (t_0, \frac{1}{2}t_f]$ while this deformation is reversed for $t \in (\frac{1}{2}t_f, t_f]$ so that $\varphi(\boldsymbol{x}, t_0) = \varphi(\boldsymbol{x}, t_f)$. Hence, we can compare the functions $\varphi(\boldsymbol{x}, t_0)$ and $\varphi(\boldsymbol{x}, t_f)$ and compute the approximation errors introduced by the discretization and the maintaining methods implemented in the toolbox.

5.1.1.1 2D example: Swirling flow vortex

On $\Omega = (0, 1)$, we introduce the divergence-free velocity field

$$\vec{V}(\boldsymbol{x}) = \vec{V}(x, y) := \begin{pmatrix} -\sin(\pi x)^2 \sin(2\pi y) \cos(\frac{\pi t}{t_f}) \\ \sin(2\pi x)^2 \sin(\pi y)^2 \cos(\frac{\pi t}{t_f}) \end{pmatrix} \tag{5.1}$$

and consider the level set problem for the time interval $[t_0, t_f] = [0, 2]$, where the initial function is given by

$$\varphi(\boldsymbol{x}, \boldsymbol{c}, t_0) := \|\boldsymbol{x} - \boldsymbol{c}\|_2 - r_0, \tag{5.2}$$

with $\boldsymbol{c} = (0.5, 0.75)^T$ and $r_0 = 0.15$ [95]. A visualization of the zero level set $\Gamma_h(t)$ for some points in time is given in Figure 5.1.

5.1.1.2 2D example: Deforming droplet

This example describes the movement and deformation of a circle and is adapted from [29]. For $t \in [0, 20]$ let $\Omega = (0, 1)^2$ be the considered domain and

$$\varphi(\boldsymbol{x}, \boldsymbol{c}, t_0) := \|\boldsymbol{x} - \boldsymbol{c}\|_2 - r_0, \tag{5.3}$$

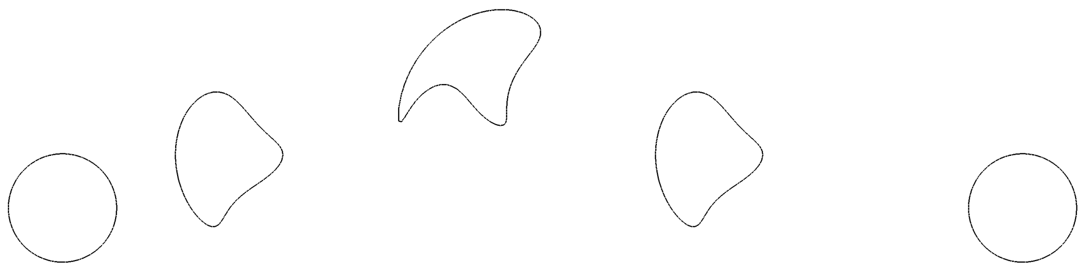


Figure 5.2 2D example rising deforming droplet: Zero level set $\Gamma_h(t)$ of the reference solution $\varphi_h(\boldsymbol{x}, t)$ for $t = 0,\ t = 5,\ t = 10,\ t = 15$ and $t = 20$.

with $\boldsymbol{c} = (0.5, 0.25)^T$ and $r_0 = 0.15$, be the initial level set function with zero level set $\Gamma(t_0)$. We consider the evolution of $\varphi(\boldsymbol{x}, t)$ for the velocity field given by

$$\vec{V}(\boldsymbol{x}, t) = \vec{V}(x, y, t) = \begin{cases} a(\boldsymbol{x}) \cdot (y - 0.5, -(x - 0.5)), & \text{for } t \leq \frac{1}{2} t_f, \\ -a(\boldsymbol{x}) \cdot (y - 0.5, -(x - 0.5)), & \text{for } t > \frac{1}{2} t_f, \end{cases} \tag{5.4}$$

with

$$a(\boldsymbol{x}) = \begin{cases} 4\|(\boldsymbol{x} - (0.5, 0.5)^T)\|_2 \ (0.5 - \|\boldsymbol{x} - (0.5, 0.5)^T\|_2), & \text{for } \|\boldsymbol{x} - (0.5, 0.5)^T\|_2 \leq 0.5, \\ 0, & \text{otherwise} \end{cases} .$$

Figure 5.2 visualizes $\Gamma_h(t)$ for different points in time.

5.1.1.3 3D example: Swirling flow vortex

The swirling flow vortex example described in Section 5.1.1.1 can be extended to a 3D scenario [95]. Therefore, we choose $\Omega = (0, 1)^3$ and define

$$\varphi(\boldsymbol{x}, \boldsymbol{c}, t_0) := \|\boldsymbol{x} - \boldsymbol{c}\|_2 - r_0, \tag{5.5}$$

with $\boldsymbol{c} = (0.35, 0.35, 0.35)^T$ and $r_0 = 0.15$ whose zero level set $\Gamma(t_0)$ is a a sphere centered at $c(\boldsymbol{x})$ with radius r_0. The considered velocity field is given by

$$\vec{V}(\boldsymbol{x}, t) = \vec{V}(x, y, z, t) = \begin{pmatrix} 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/t_f) \\ -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/t_f) \\ -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/t_f) \end{pmatrix}, \tag{5.6}$$

where the considered time interval is $[t_0, t_f] = [0, 2]$. The zero level set $\Gamma_h(t)$ is visualized in Figure 5.3 for some points in time.

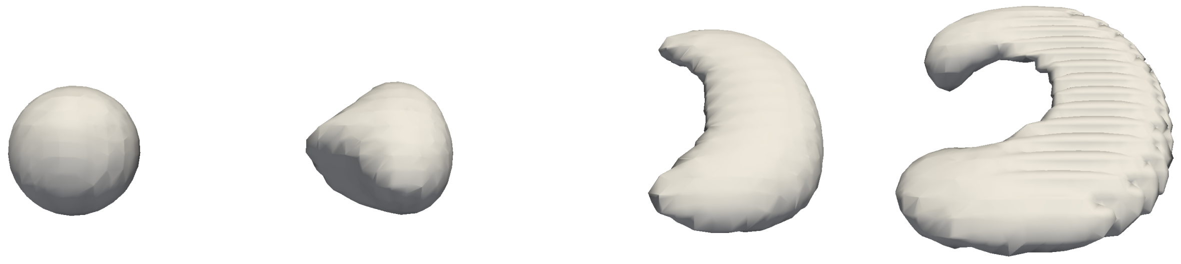


Figure 5.3 3D example deformation flow: Zero level set $\Gamma_h(t)$ of the reference solution $\varphi_h(\boldsymbol{x}, t)$ for $t = 0,\ t = 0.25$, $t = 0.5$, and $t = 1.0$,.

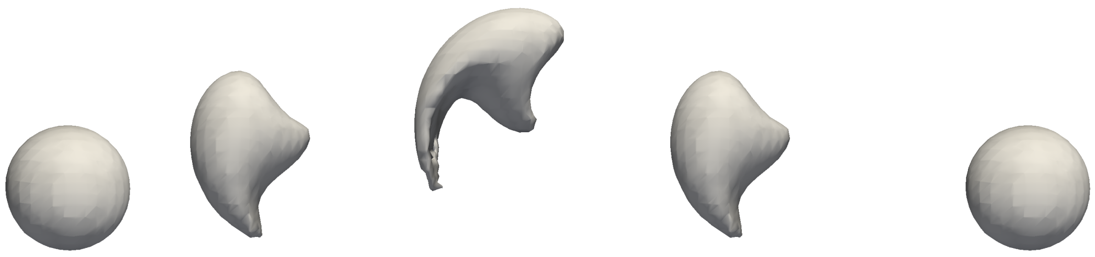


Figure 5.4 3D example rising deforming droplet: Zero level set $\Gamma_h(t)$ of the reference solution $\varphi_h(\boldsymbol{x}, t)$ for $t = 0,\ t = 5$, $t = 10$, $t = 15$ and $t = 20$.

5.1.1.4 3D example: Deforming droplet

The last scenario considered in the studies extends the example considered in Section 5.1.1.2 to three dimensions [29]. For $\Omega = (0, 1)^3$ and $t \in [0, 20]$, we consider the evolution of

$$\varphi(\boldsymbol{x}, \boldsymbol{c}, t_0) := \|\boldsymbol{x} - \boldsymbol{c}\|_2 - r_0, \tag{5.7}$$

with $\boldsymbol{c} = (0.5, 0.25, 0.5)^T$ and $r_0 = 0.2$ for the velocity field given by

$$\vec{V}(\boldsymbol{x}, t) = \vec{V}(x, y, z, t) = \begin{cases} a(\boldsymbol{x}) \cdot (y - 0.5, -(x - 0.5)), & \text{for } t \leq \frac{1}{2}t_f, \\ -a(\boldsymbol{x}) \cdot (y - 0.5, -(x - 0.5)), & \text{for } t > \frac{1}{2}t_f, \end{cases} \tag{5.8}$$

with

$$a(\boldsymbol{x}) = \begin{cases} 4\|(\boldsymbol{x} - 0.5(1, 1, 1)^T)\|_2(0.5 - \|\boldsymbol{x} - 0.5(1, 1, 1)^T\|_2), & \text{for } \|\boldsymbol{x} - 0.5(1, 1, 1)^T\|_2 \leq 0.5, \\ 0, & \text{otherwise} \end{cases}.$$

The movement and deformation of the droplet given by the zero level set $\Gamma(t)$ is shown for different times in Figure 5.4.

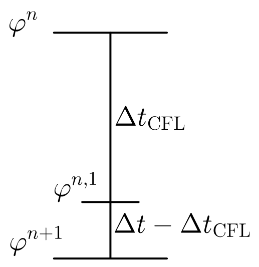

φ^n
Δt_{CFL}
$\varphi^{n,1}$
$\Delta t - \Delta t_{\mathrm{CFL}}$
φ^{n+1}

Figure 5.5 Time stepping synchronization: Intermediate time steps for the solution of the level set problem are synchronized with the time step size for the thermal problem.

5.1.2 Simulation setup and computational approach

All examples are considered on a uniform triangulation $\mathcal{S}_h$ with $\bigcup_{S\in\mathcal{S}_h} S = \bar{\Omega}$. The used approximation spaces are $V^2_{\mathrm{cg},h}$ for the level set function and $\left(V^2_{\mathrm{cg},h}\right)^d$, $d = 2, 3$, for the discrete velocity fields $\vec{V}_h$. The time discretization scheme used for the detailed parameter studies is the implicit Euler method, but the Crank-Nicolson scheme, the explicit Euler method and some total variation diminishing Runge-Kutta schemes have also been tested. The number of elements N_{el} in our studies varies from $N_{\mathrm{el}} = 2\times 10^d$ to $N_{\mathrm{el}} = 2\times 80^d$, $d = 2, 3$. The narrow band parameters are either $\beta_I = 2$, $\beta_O = 4$, $\gamma = 6$, or $\beta_I = 3$, $\beta_O = 6$, $\gamma = 9$ and the time step sizes are chosen as 2^{-k} with $k \in \{4, 5, 6, 7, 8, 9\}$. However, due to the CFL conditions arising by using the narrow band approach, we may have to adapt the time step size or use intermediate time steps. Hence, we define the time step size Δt as major time step size and introduce an adjusted time step size Δt_φ, if necessary, to take into account the CFL condition(s) with time step size Δt_{CFL}. Consequently, we may have to introduce potentially non-equidistant intermediate time steps $t_{n,i}$ in order to reach t_{n+1}. The described procedure is illustrated in Figure 5.5 and the full computational approach is shown in Algorithm 4.

5.1.3 Results

For validating the methods and their implementation as well as analyzing the quality of the computed solutions, the errors

$$e^{\infty}_{\mathrm{vol}} := \max_{n=0,\dots,N_t} \frac{V(\Omega^n_{i,h}) - V(\Omega^n_i)}{V(\Omega^n_i)}, \quad i = 1 \text{ or } i = 2, \tag{5.9}$$

$$e^{t_f}_{\mathrm{dist}} := \max_{\boldsymbol{x}\in\Gamma^{N_t}_h} \min_{\tilde{\boldsymbol{x}}\in\Gamma^{N_t}} \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2, \tag{5.10}$$

Algorithm 4 Level set solver.

Input: Ω, Γ_{in}, $\mathcal{S}_h$, $\varphi(t_0)$, $\vec{V}(t)$, g_{in}, t_0, t_f, Δt, $\Delta t_{\max}$, $\Delta t_{\min}$, β_O, β_I, γ.
Output: φ^n, Γ_h^n for $n > 0$.

```
1: procedure Level set solver.
2:     Initialization (1): $t_n = t_0$, $\varphi_h^n = \varphi(t_0)$, $\vec{V}^n = \vec{V}(t_0)$.
3:     Initialization (2): Define $V^2_{\text{cg},h,g_D}(t_0)$
4:     if Narrow band used then
5:         Initialization (3): Initialize $\Omega_{\text{INB}}$ and $\Omega_{\text{ONB}}$ for $\varphi_h^0$.
6:     end if
7:     for $n = 0, \dots, N_t - 1$ do
8:         Assign $t_\varphi = t_0 + n\Delta t$ (current simulation time) and $\varphi_h^{t_\varphi} = \varphi_h^n$.
9:         while ( do$t_\varphi < (n + 1\Delta t)$)
10:            Compute $\Delta t_\varphi := \min\{\max\{\Delta t_{\text{CFL}}, \Delta t_{\min}\}, (n+1)\Delta t - t_\varphi, \Delta t_{\max}\}$.
11:            Solve level set problem to get $\varphi_h^{t_\varphi+\Delta t_\varphi}$
12:            if Reinitialization required then
13:                Perform initialization phase, cf. Section 4.4.1.3.
14:                if Volume correction desired then
15:                    Perform volume correction, cf. Section 4.4.1.4.
16:                end if
17:                Perform extension phase, cf. Section 4.4.1.3, to get new $\varphi_h^{t_\varphi+\Delta t_\varphi}$.
18:                if Narrow band used then
19:                    Update $\Omega_{\text{INB}}$ and $\Omega_{\text{ONB}}$.
20:                end if
21:            end if
22:            Set $t_\varphi = t_\varphi + \Delta t_\varphi$.
23:        end while
24:        Set $\varphi_h^{n+1} = \varphi_h^{t_\varphi}$.
25:    end for
26: end procedure
```

are considered. If we do not use the narrow band[*], we also analyze the convergence behavior for the error

$$e^n_{L^2} := \|\varphi^n - \varphi^n_h\|_{L^2(\Omega)}, \quad n = 0, \dots, N_t. \tag{5.11}$$

To put it in a nutshell, the results of the validation tests and parameter studies when considering the introduced examples are as follows:

(Result 1) The level set solver without using any maintaining method converges with the optimal rates in (almost) every example[†].

(Result 2) Using SUPG stabilization worsens the numerical results somewhat, if the velocity is rather small. For high velocities, where the conventional finite element method

[*]Please note that we do not use errors such as $\|\varphi^n - \varphi^n_h\|_{L^2(\Omega)}$ when applying the narrow band method, since the reinitialization reestablishes the signed distance property only in a proximity of the interface. Consequently, such errors do not necessarily converge.

[†]However, the absolute errors are rather large compared to simulation where the maintaining methods have been used.

becomes unstable, using the stabilization method allows for retaining the stability so that reasonable solutions can be computed.

(Result 3) The reinitialization should always be performed with a volume correction step after the initialization phase.

(Result 4) The global and local volume correction methods both significantly reduce the volume defect. However, the (more efficient) global approach has a rather large impact on the solution since, by design, it tries to preserve the shape of the zero level set. In contrast to this, the local approach has only a very small influence on the solution process and the convergence order.

(Result 5) Even with volume correction, the reinitialization frequency has an impact on the accuracy of the solution. Depending on the CFL condition, and hence the mesh size h, it is advantageous to choose the maximum Δt_{CFL} possible to reduce the total number of reinitializations.

(Result 6) The choice of the narrow band parameters β_I, β_O, and γ has almost no influence on the solution itself but of course on the computational time.

5.2 Multiphase steady-state diffusion equation

To demonstrate the capability of the `miXFEM` framework to consider problems involving multiple discontinuities and multi-junctions, we solve the following diffusion problem: Let $\Omega = (0,1)^2$ with $\partial\Omega = \Gamma_\mathrm{D} \cup \Gamma_\mathrm{N}$, where $\Gamma_\mathrm{D} = [0,1] \times \{0\}$ and $\Gamma_\mathrm{N} = \partial\Omega \setminus \Gamma_\mathrm{D}$, be a domain that consists of the subdomains

$$\begin{aligned}
\Omega_1 &:= \left\{\boldsymbol{x} \in \Omega \,:\, y < \tfrac{1}{4}\right\}, \\
\Omega_2 &:= \left\{\boldsymbol{x} \in \Omega \,:\, x < \tfrac{1}{4} \wedge y > \tfrac{1}{4}\right\}, \\
\Omega_3 &:= \left\{\boldsymbol{x} \in \Omega \,:\, y > \tfrac{1}{4} \wedge y < \tfrac{2}{3}x + \tfrac{1}{12}\right\}, \\
\Omega_4 &:= \left\{\boldsymbol{x} \in \Omega \,:\, x > \tfrac{1}{4} \wedge y > \tfrac{2}{3}x + \tfrac{1}{12} \wedge y^2 > \tfrac{1}{25} - (x-0.66)^2 + 0.66^2\right\}, \\
\Omega_5 &:= \left\{\boldsymbol{x} \in \Omega \,:\, y > \tfrac{2}{3}x + \tfrac{1}{12} \wedge y^2 < \tfrac{1}{25} - (x-0.66)^2 + 0.66^2\right\},
\end{aligned} \tag{5.12}$$

that are separated by the interfaces

$$\begin{aligned}
\Gamma_{1,2} &:= \left\{\boldsymbol{x} \in \Omega \,:\, x \leq \tfrac{1}{4} \wedge y = \tfrac{1}{4}\right\}, \\
\Gamma_{1,3} &:= \left\{\boldsymbol{x} \in \Omega \,:\, x > \tfrac{1}{4} \wedge y = \tfrac{1}{4}\right\}, \\
\Gamma_{2,4} &:= \left\{\boldsymbol{x} \in \Omega \,:\, x = \tfrac{1}{4} \wedge y > \tfrac{1}{4}\right\}, \\
\Gamma_{3,4} &:= \left\{\boldsymbol{x} \in \Omega \,:\, y > \tfrac{1}{4} \wedge y = \tfrac{2}{3}x + \tfrac{1}{12} \wedge y^2 > \tfrac{1}{25} - (x-0.66)^2 + 0.66^2\right\}, \\
\Gamma_{3,5} &:= \left\{\boldsymbol{x} \in \Omega \,:\, y > \tfrac{1}{4} \wedge y = \tfrac{2}{3}x + \tfrac{1}{12} \wedge y^2 < \tfrac{1}{25} - (x-0.66)^2 + 0.66^2\right\}, \\
\Gamma_{4,5} &:= \left\{\boldsymbol{x} \in \Omega \,:\, y > \tfrac{2}{3}x + \tfrac{1}{12} \wedge y^2 = \tfrac{1}{25} - (x-0.66)^2 + 0.66^2\right\},
\end{aligned} \tag{5.13}$$

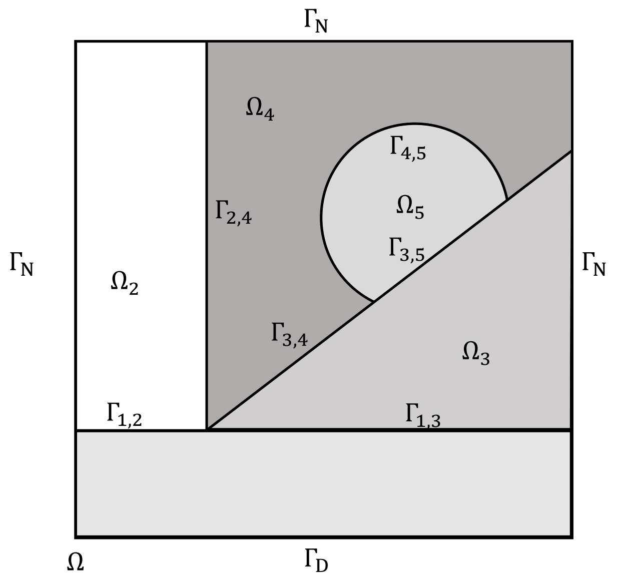


Figure 5.6 Problem setting for the multiphase example: Hold-all domain Ω is decomposed into subdomains Ω_l, $l = 1, \dots, 5$ by the (active parts) of the zero level sets Γ_i of hierarchically ordered level set functions.

see Figure 5.6.

We introduce the sets $\mathcal{F} := \{(2,4),(3,4),(3,5),(4,5)\}$ and $\mathcal{D} := \{(1,2),(1,3)\}$ containing index pairs of interfaces at which we impose different conditions and consider the steady-state diffusion problem given by

$$-\nabla \cdot (\kappa \nabla u) = f \qquad \text{in } \bigcup_{i=1}^{5} \Omega_i, \tag{5.14}$$

$$u = g_\mathrm{D} \qquad \text{on } \Gamma_\mathrm{D}, \tag{5.15}$$

$$-\kappa \nabla u \cdot \vec{n} = g_\mathrm{N} \qquad \text{on } \Gamma_\mathrm{N}, \tag{5.16}$$

$$[\![\kappa \nabla u]\!] \cdot \vec{n}_{i,l} = g_{i,l} \qquad \text{on } \Gamma_{i,l}, \quad (i,l) \in \mathcal{F}, \tag{5.17}$$

$$[\![u]\!] = q_{i,l} \qquad \text{on } \Gamma_{i,l}, \quad (i,l) \in \mathcal{F}, \tag{5.18}$$

$$u|_{\Omega_i} = g_i, \qquad \text{on } \Gamma_{i,l}, \quad (i,l) \in \mathcal{D}, \tag{5.19}$$

$$u|_{\Omega_l} = g_l, \qquad \text{on } \Gamma_{i,l}, \quad (i,l) \in \mathcal{D}, \tag{5.20}$$

where the data for g_D, g_N, $g_{i,l}$, $q_{i,l}$, g_i, and g_l is chosen so that the function

$$u(\boldsymbol{x}) = u(x,y) := \begin{cases} x^2 + 2y^2 & \text{for } x \in \Omega_1 \\ \frac{(1-\cos(4\pi(x-0.5)))(1-\cos(4\pi(y-0.5)))}{4} & \text{for } x \in \Omega_2 \\ \exp\left(\sqrt{(x+0.1)} + y^2\right) & \text{for } x \in \Omega_3 \\ \frac{1-\cos\left(4\pi(y-\frac{3}{4})\right)}{9} & \text{for } x \in \Omega_4 \\ 1 - \sin\left(4\pi(x-0.5)\right)\left(1-\exp(y-0.5)\right) & \text{for } x \in \Omega_5 \end{cases} \tag{5.21}$$

is a solution to the problem given by equations (5.14) to (5.20) for $\kappa|_{\Omega_i} = \kappa_i$ and $\kappa_1 = 5$, $\kappa_2 = 10$, $\kappa_3 = 15$, $\kappa_4 = 20$, and $\kappa_5 = 30$.

5.2.1 Modeling and implementation in `miXFEM`

We model the given problem using the hierarchical level set method with $N_{\text{dom}} = 5$. Therefore, we define hierarchically ordered level set functions[‡]

$$\begin{aligned} \varphi_1(\boldsymbol{x}) &:= y - \frac{1}{4}, \\ \varphi_2(\boldsymbol{x}) &:= x - \frac{1}{4}, \\ \varphi_3(\boldsymbol{x}) &:= y - \frac{2}{3}x + \frac{1}{12}, \\ \varphi_4(\boldsymbol{x}) &= \frac{1}{5} - \sqrt{(x-0.66)^2 + (y-0.66)^2}, \end{aligned} \tag{5.22}$$

where the (active parts of the hierarchical) zero level sets

$$\begin{aligned} \Gamma_1 &:= \Gamma_{1,2} \cup \Gamma_{1,3}, \\ \Gamma_2 &:= \Gamma_{2,4}, \\ \Gamma_3 &:= \Gamma_{3,4} \cup \Gamma_{3,5}, \quad \text{and} \\ \Gamma_4 &:= \Gamma_{4,5}, \end{aligned} \tag{5.23}$$

decompose Ω into the five subdomains Ω_l, $l = 1, 2, 3, 4, 5$. Now, we approximate the solution of this problem numerically with the hierarchical eXtended finite element method.

Let $\mathcal{S}_h$ be a triangulation covering Ω_h with $\Omega_h = \Omega$ since Ω is polygonally bounded which also means that we have $\Gamma_{\text{D},h} = \Gamma_{\text{D}}$ and $\Gamma_{\text{N},h} = \Gamma_{\text{N}}$. Using Nitsche's method as described in Section 3.3.2, the discrete problem reads: Find $u_h \in V_h^m := \{v_h \in C^0(\Omega_i) : v_h|_{S\cap\Omega_i} \in$

[‡]The number of the level set function corresponds to the hierarchy level where the lowest number denotes the highest hierarchy level

$\mathcal{P}^m(S \cap \Omega_i),\ \forall S \in \mathcal{S}_h,\ i = 1, \ldots, N_{\mathrm{dom}}\}$ with $u_h = g_{\mathrm{D}}$ on Γ_{D} such that

$$\begin{aligned}
&\int_{\bigcup_{i=1}^5 \Omega_i} \kappa \nabla u_h \cdot \nabla v_h \,\mathrm{d}x \\
&\quad - \sum_{(i,l)\in\mathcal{F}} \left(\int_{\Gamma_i \cap \Gamma_{i,l}} \left([\![v_h]\!]\{\kappa \nabla u_h \cdot \vec{n}_i\} + [\![u_h]\!]\{\kappa \nabla v_h \cdot \vec{n}_i\} - \frac{\lambda_{i,l}}{h} [\![u_h]\!][\![v_h]\!] \right) \mathrm{d}x \right) \\
&\quad - \sum_{(i,l)\in\mathcal{D}} \left(\int_{\Gamma_i \cap \Gamma_{i,l}} \left(\kappa|_{\Omega_i} \nabla u_h|_{\Omega_i} \cdot \vec{n}_i v_h|_{\Omega_i} + \kappa|_{\Omega_i} \nabla v_h|_{\Omega_i} \cdot \vec{n}_i u_h|_{\Omega_i} - \frac{\lambda_{i,l}}{h} u_h|_{\Omega_i} v_h|_{\Omega_i} \right) \mathrm{d}x \right. \\
&\qquad\qquad \left. - \int_{\Gamma_i \cap \Gamma_{i,l}} \left(\kappa|_{\Omega_l} \nabla u_h|_{\Omega_l} \cdot \vec{n}_i v_h|_{\Omega_l} + \kappa|_{\Omega_l} \nabla v_h|_{\Omega_l} \cdot \vec{n}_i u_h|_{\Omega_l} + \frac{\lambda_{i,l}}{h} u_h|_{\Omega_l} v_h|_{\Omega_l} \right) \mathrm{d}x \right) \\
&= \int_{\bigcup_{i=1}^5 \Omega_i} f_h v_h \,\mathrm{d}x - \int_{\Gamma_{\mathrm{N}}} g_{\mathrm{N}} v_h \,\mathrm{d}x \\
&\quad + \sum_{(i,l)\in\mathcal{F}} \left(\int_{\Gamma_i \cap \Gamma_{i,l}} \left(\langle v_h \rangle g_{i,l} \,\mathrm{d}x - q_{i,l}\{\kappa \nabla v_h \cdot \vec{n}_i\} + \frac{\lambda_{i,l}}{h} q_{i,l} [\![v_h]\!] \right) \mathrm{d}x \right) \\
&\quad + \sum_{(i,l)\in\mathcal{D}} \left(\int_{\Gamma_i \cap \Gamma_{i,l}} \left(\kappa|_{\Omega_i} \nabla v_h|_{\Omega_i} \cdot \vec{n}_i g_i + \frac{\lambda_{i,l}}{h} g_i v_h|_{\Omega_i} \right) \mathrm{d}x \right. \\
&\qquad\qquad \left. - \int_{\Gamma_i \cap \Gamma_{i,l}} \left(\kappa|_{\Omega_l} \nabla v_h|_{\Omega_l} \cdot \vec{n}_i g_l - \frac{\lambda_{i,l}}{h} g_l v_h|_{\Omega_l} \right) \mathrm{d}x \right)
\end{aligned} \tag{5.24}$$

holds for all $v_h \in V_h^m$. The corresponding part of the discretized variational formulation implemented in `UFL` when using `miXFEM` is shown in Figure 5.7.

In regards to the decomposition of zero level set Γ_i into interfaces $\Gamma_{i,l}$, please recall that a sophisticated domain id mapping is only necessary in situations where we want to impose a different type of interface conditions on different interface parts $\Gamma_{i,l}$ of the same zero level set Γ_i. Since that is not the case in the given setting, we can simply define the mapping by[§]

```
[...]
std::vector<uint > domain_id_map(4);
domain_id_map[0] = 0;
domain_id_map[1] = 1;
domain_id_map[2] = 2;
domain_id_map[3] = 3;
std::vector<std::vector<std::pair<uint, bool> > >
  domain_id_restrictions(4, std::vector<std::pair<uint, bool> >(0));
[...]
```

To take into account different fluxes and jump conditions or Dirichlet values at the same zero level set Γ_i, we introduce piecewise defined functions. Alternatively, we could also introduce additional measures `dc(i)` and use different function for imposing the required conditions.

[§]Please note that the indexing in `C++` starts with 0 while our notation begins with 1.

```
[...]
a = k*dot(grad(v), grad(u))*dx \
    - k('-') * dot(grad(u('-')), n(phi_0('-'))) * v('-') * dc(0) \
    + k('+') * dot(grad(u('+')), n(phi_0('+'))) * v('+') * dc(0) \
    - k('-') * dot(grad(v('-')), n(phi_0('-'))) * u('-') * dc(0) \
    + k('+') * dot(grad(v('+')), n(phi_0('+'))) * u('+') * dc(0) \
    - jump(u) * wavg(k*inner(nabla_grad(v),n(phi_1)) ) * dc(1) \
    - jump(u) * wavg(k*inner(nabla_grad(v),n(phi_2)) ) * dc(2) \
    - jump(u) * wavg(k*inner(nabla_grad(v),n(phi_3)) ) * dc(3) \
    - jump(v) * wavg(k*inner(nabla_grad(u),n(phi_1)) ) * dc(1) \
    - jump(v) * wavg(k*inner(nabla_grad(u),n(phi_2)) ) * dc(2) \
    - jump(v) * wavg(k*inner(nabla_grad(u),n(phi_3)) ) * dc(3) \
    + nitsche('+') * u('-') * v('-') * dc(0) \
    + nitsche('+') * u('+') * v('+') * dc(0) \
    + nitsche('+') * jump(u) * jump(v) * dc(1) \
    + nitsche('+') * jump(u) * jump(v) * dc(2) \
    + nitsche('+') * jump(u) * jump(v) * dc(3)
L = - k('-') * dot(grad(v('-')), n(phi_0('-'))) * u_exact('-') * dc(0) \
    + k('+') * dot(grad(v('+')), n(phi_0('+'))) * u_exact('+') * dc(0) \
    - jump(u_exact) * wavg(k*inner(nabla_grad(v),n(phi_1)) ) * dc(1) \
    - jump(u_exact) * wavg(k*inner(nabla_grad(v),n(phi_2)) ) * dc(2) \
    - jump(u_exact) * wavg(k*inner(nabla_grad(v),n(phi_3)) ) * dc(3) \
    + jump(k*dot(nabla_grad(u_exact),n(phi_1)) ) * cavg(v) * dc(1) \
    + jump(k*dot(nabla_grad(u_exact),n(phi_2)) ) * cavg(v) * dc(2) \
    + jump(k*dot(nabla_grad(u_exact),n(phi_3)) ) * cavg(v) * dc(3) \
    + nitsche('+') * u_exact('-') * v('-') * dc(0) \
    + nitsche('+') * u_exact('+') * v('+') * dc(0) \
    + nitsche('+') * jump(u_exact) * jump(v) * dc(1) \
    + nitsche('+') * jump(u_exact) * jump(v) * dc(2) \
    + nitsche('+') * jump(u_exact) * jump(v) * dc(3) \
    - nabla_div(k*nabla_grad(u_exact)) * v * dx \
    + k * dot(N,nabla_grad(u_exact)) * v * ds(0)
[...]

```

Figure 5.7 Part of the UFL implementation of the discretized variational formulation of the steady-state diffusion problem in miXFEM.

5.2.2 Approximation quality and convergence order

Using the approximation space $V_h = \{v_h \in C^0(\Omega_i) : v_h|_{S\cap\Omega_i} \in \mathcal{P}^1(S \cap \Omega_i),\ \forall S \in \mathcal{S}_h,\ i = 1, \dots, 5\}$, we run the multiphase example on a uniform triangulation $\mathcal{S}_h$ with $N_{\text{el}} = \{2\times 5^2, 2\times 9^2, 2\times 17^2, 2\times 33^2, 2\times 65^2\}$ elements. As expected, the convergence rates of the L^2 and H^1 approximation errors for the respective projection, as well as the solution, are optimal and do not depend on the interfaces' positions, see Tables 5.1 and 5.2. The solution of the problem is visualized for $N_{\text{el}} = 2\times 17^2$ in Figure 5.8.

5.3 Two-phase Stefan problem

Motivated by the mentioned engineering applications, we now consider the two-phase Stefan problem as an example process for validating the framework miXFEM for a problem with time-dependent discontinuity. Therefore, we use the Stefan condition to couple the heat equation

N_{el}	$\inf_{v_h \in V^1_{\mathrm{cg}}} \|u - v_h\|_{L^2(\Omega)}$	eoc	$\inf_{v_h \in V^1_{\mathrm{cg}}} \|u - v_h\|_{H^1(\Omega)}$	eoc
2×5^2	0.051359	-	1.774387	-
2×9^2	0.015551	2.0326	0.998924	0.9780
2×17^2	0.003824	2.2057	0.508045	1.0632
2×33^2	0.000969	2.0697	0.256290	1.0316
2×65^2	0.000247	2.0164	0.127814	1.0263

Table 5.1 Projection errors and estimated order of convergence for the specified example using the approximation space V_h.

N_{el}	$\|u - u_h\|_{L^2(\Omega)}$	eoc	$\|u - u_h\|_{H^1(\Omega)}$	eoc
2×5^2	0.238422	-	3.743190	-
2×9^2	0.083712	1.7807	1.612254	1.4342
2×17^2	0.018471	2.3761	0.683200	1.3516
2×33^2	0.003809	2.3803	0.287689	1.3044
2×65^2	0.000655	2.5970	0.130304	1.1685

Table 5.2 Approximation errors and estimated order of convergence for the specified example using the approximation space V_h.

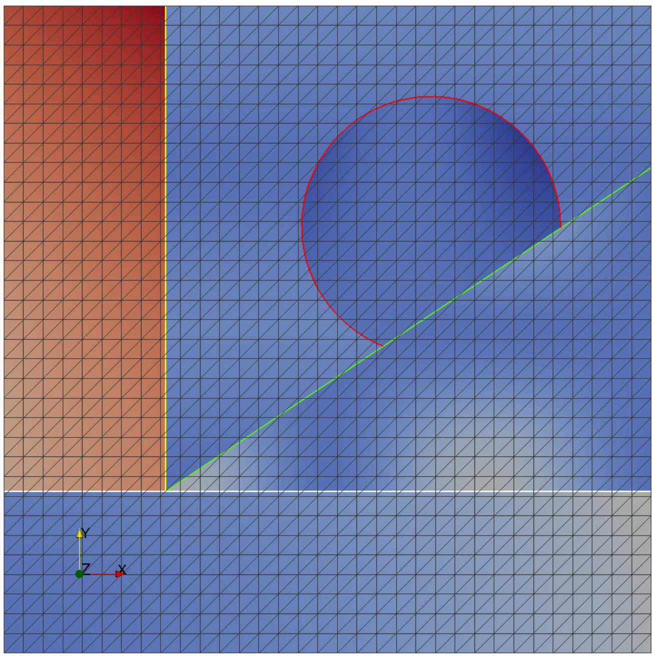
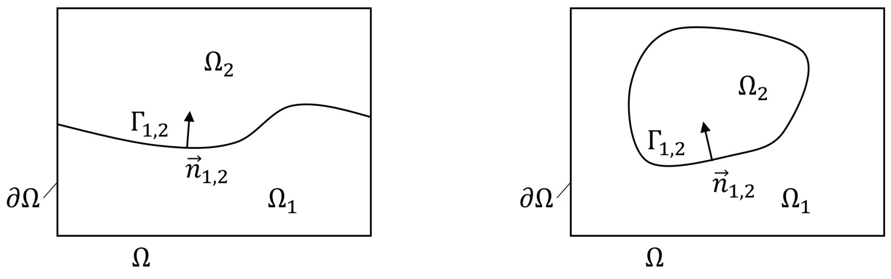


Figure 5.8 Visualization of the solution u_h for $N_{\mathrm{el}} = 2 \times 17^2$: Γ_1 is shown in white, Γ_2 is shown in yellow, Γ_3 is shown in green, and Γ_4 is shown in red.

with the conventional level set problem. This problem has been considered in more detail for several examples with known analytical solution in our publications [2, 93].

5.3.1 Model

Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, with $\partial\Omega$ polygonal, be a fixed domain consisting for $t \in [t_0, t_f]$ of the disjoint regions $\Omega_1(t)$ (the solid domain) and $\Omega_2(t)$ (the liquid domain) that are separated by



(a) Setting with open interface. (b) Setting with closed interface.

Figure 5.9 Sketches of different settings for the two-phase Stefan problem.

an internal boundary $\Gamma_{1,2}(t)$ (the solid-liquid interface). We assume that $\Gamma_{1,2}(t)$ is sharp and sufficiently smooth for all $t \in [t_0, t_f]$ and introduce the unit normal vector $\vec{n}_{1,2}(t, \boldsymbol{x}) = \vec{n}_1 = -\vec{n}_2$ to $\Gamma_{1,2}(t)$ pointing from Ω_1 into Ω_2. Sketches of different settings are visualized in Figure 5.9.

Heat equation

The temperature field is given by $u\colon \Omega \times [t_0, t_f] \to \mathbb{R}$ with $u|_{\Omega_i} = u_i,\ i \in \{1, 2\}$ and its evolution is described by

$$\rho c \frac{\partial u}{\partial t} - \nabla \cdot (\lambda \nabla u) = f, \quad \text{in } \Omega_1(t) \cup \Omega_2(t),\ t \in (t_0, t_f), \tag{5.25}$$

in which we assume that the density ρ is constant in Ω while the specific heat capacities c_i and the thermal conductivities λ_i are assumed as constant in each subdomain $\Omega_i(t)$.

For the boundary $\partial\Omega = \Gamma_\mathrm{D} \cup \Gamma_\mathrm{N} \cup \Gamma_\mathrm{R}$ with $\Gamma_\mathrm{D} \cap \Gamma_\mathrm{N} \cap \Gamma_\mathrm{R} = \emptyset$, the following conditions are given

$$u = g_\mathrm{D}, \qquad \text{on } \Gamma_\mathrm{D} \times (t_0, t_f], \tag{5.26}$$

$$-\lambda \nabla u \cdot \vec{n} = g_\mathrm{N}, \qquad \text{on } \Gamma_\mathrm{N} \times (t_0, t_f], \tag{5.27}$$

$$-\lambda \nabla u \cdot \vec{n} = g_\mathrm{R}(u), \qquad \text{on } \Gamma_\mathrm{R} \times (t_0, t_f], \tag{5.28}$$

where $\vec{n}$ denotes the outer unit normal to $\partial\Omega$. Here, the function $g_\mathrm{R}(u)$ describes the thermal transfer to the environment which we model as combination of Newton's law of cooling and the Stefan-Boltzmann law

$$g_\mathrm{R}(u) = g_c(u) + g_r(u) = \alpha(u_a - u) + \epsilon\sigma(u_a^4 - u^4) \tag{5.29}$$

with α being the heat transfer coefficient, u_a denoting the ambient temperature, σ is the Stefan-Boltzmann constant and ϵ is the emissivity of the material. As before, the coefficients α and ϵ are assumed to be constant in each subdomain $\Omega_i(t)$.

At the solid-liquid phase boundary $\Gamma_{1,2}(t)$, we expect the so-called isothermal interface condition

$$u(\cdot, t) = u_{1,2} \quad \text{on } \Gamma_{1,2}(t) \tag{5.30}$$

to hold for all times $t \in [t_0, t_f]$ with $u_{1,2}$ being the melting temperature. As a result, we have

$$u(\cdot, t) < u_{1,2} \text{ in } \Omega_1(t), \quad \text{and} \quad u(\cdot, t) > u_{1,2} \text{ in } \Omega_2(t). \tag{5.31}$$

Initially, the temperature distribution on $\Omega_1(t_0) \cup \Omega_2(t_0)$, which has to fulfill the conditions (5.30) and (5.31), is given by

$$u(\cdot, t_0) = u_0. \tag{5.32}$$

Representation the solid-liquid interface

Since there are only two regions and one interface present, only one hierarchy level is needed within the method. Thus, the hierarchical level set method equates to the conventional approach. Given the initial zero level set $\Gamma_{1,2}(t_0)$, we introduce a corresponding signed distance function $\varphi_1(\boldsymbol{x}, t_0)$ whose evolution in time can be described by the problem

$$\begin{aligned} \frac{\partial \varphi_1}{\partial t} + \vec{V}_{1,2} \cdot \nabla \varphi_1 = 0 \qquad & \text{in } \Omega \times [t_0, t_f], \\ \varphi_1(\cdot, t_0) = \varphi_{1,0} \qquad & \text{in } \Omega, \end{aligned} \tag{5.33}$$

where $\vec{V}_{1,2} = \vec{V}_{1,2}(\boldsymbol{x}, t)$ is given by the Stefan condition

$$(\lambda_1 \nabla u_1 - \lambda_2 \nabla u_2) \cdot \vec{n}_1 = \rho L \vec{V}_{1,2} \cdot \vec{n}_1 \quad \text{on } \Gamma_{1,2}, \tag{5.34}$$

coupling the heat equation with the level set problem. In equation (5.34), L corresponds to the latent heat that is released or absorbed during the phase change.

Remark 5.1. Alternatively, one could also define a function

$$\check{\varphi}_1(\boldsymbol{x}, t_0) := u(\boldsymbol{x}, t_0) - u_{1,2}. \tag{5.35}$$

which also has the zero level set $\Gamma_{1,2}(t_0)$. Unfortunately, $\check{\varphi}_1$ may not have the regularity that is required for the transport equation (5.33). Hence, we again have to introduce a signed distance function φ_1. However, this idea can be enhanced to detect topological changes, see Section 5.4.1.

Coupled problem

Altogether, the coupled two-phase Stefan problem in level set formulation is given by: For $t \in [t_0, t_f]$ find the solid-liquid interface $\Gamma_{1,2}$ given as zero level of $\varphi_1 \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$ and the temperature distribution u which is sufficiently smooth, i.e., $u \in \mathcal{C}^0(\bar{\Omega} \times [t_0, t_f])$,

$u(\cdot,t)_{|\Omega_i} \in \mathcal{C}^2(\Omega_i(t))$ and $\partial_t u(\cdot,t) \in C^0(\Omega_1(t) \cup \Omega_2(t))$ such that

$$\begin{cases} \rho c \dfrac{\partial u}{\partial t} - \nabla \cdot (\lambda \nabla u) = f & \text{in } \bigcup_{i=1}^2 \Omega_i(t),\ t \in (t_0, t_f), \quad (5.36a) \\ u = g_{\mathrm{D}} & \text{on } \Gamma_{\mathrm{D}} \times (t_0, t_f], \quad (5.36b) \\ -\lambda \nabla u \cdot \vec{n} = g_{\mathrm{N}} & \text{on } \Gamma_{\mathrm{N}} \times (t_0, t_f], \quad (5.36c) \\ -\lambda \nabla u \cdot \vec{n} = g_{\mathrm{R}}(u) & \text{on } \Gamma_{\mathrm{R}} \times (t_0, t_f], \quad (5.36d) \\ u(\cdot,t) = u_{1,2} & \text{on } \Gamma_{1,2}(t), \quad (5.36e) \\ u(\cdot,t_0) = u_0 & \text{in } \bigcup_{i=1}^2 \Omega_i(t_0), \quad (5.36f) \end{cases}$$

$$(\lambda_1 \nabla u_1 - \lambda_2 \nabla u_2) \cdot \vec{n}_1 = \rho L \vec{V}_{1,2} \cdot \vec{n}_1 \qquad \text{on } \Gamma_{1,2}(t), \qquad (5.37)$$

$$\begin{cases} \dfrac{\partial \varphi_1}{\partial t} + \vec{V}_{1,2} \cdot \nabla \varphi_1 = 0 & \text{in } \Omega \times [t_0, t_f], \quad (5.38a) \\ \varphi_1(\cdot, t_0) = \varphi_{1,0} & \text{in } \Omega, \quad (5.38b) \end{cases}$$

holds for given sufficiently smooth data.

5.3.2 Discretization and computational approach

We decouple the problem given by equation (5.36) to (5.38) into the subproblems (5.36) and (5.38) and discretize them individually. Therefore, we first consider the geometry problem given by (5.38) and use the old temperature data for computing the velocity $\vec{V}_{1,2}(t) \cdot \vec{n}_{1,2}$ on $\Gamma_{1,2}(t)$ by evaluating the coupling condition (5.37) with one of the methods described in Section 4.4.2. As (5.37) only provides the normal component of $\vec{V}_{1,2}(t)$ on $\Gamma_{1,2}(t)$, the computed velocity is extended to obtain a full velocity field, see Section 4.4.2.2, which is required for the solution of the transport problem.

The heat equation is discretized with the (hierarchical) eXtended finite element method as described in Section 3.3. Therefore, we first scale equation (5.36) by $\frac{1}{\rho c}$ and then use Rothe's method, see Section 3.3.3. The resulting quasi-stationary problem, which is very similar to Example 3.4, is then discretized in space whereas the interface condition is imposed using Nitsche's method. The level set problem is discretized as described in Section 4.4.1 and the velocity field $\vec{V}_{1,2}$ is computed using the approach presented in Section 4.4.2.

For computing a numerical solution, we solve the subproblems in succession. Therefore, we begin with computing the velocity field by evaluating the Stefan condition for the initial temperature $u(\cdot, t_0)$ and then solving the level set problem. While we use the same time discretization scheme for the heat equation and the level set problem, we still may need to introduce intermediate time steps for solving the level set problem due to the CFL conditions arising due to the narrow band as explained in Section 4.4.1.5. To take this into account, we use a very similar strategy

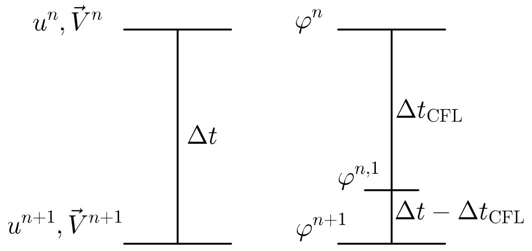


Figure 5.10 Time stepping synchronization: Intermediate time steps for the solution of the level set problem are synchronized with the time step size for the thermal problem.

Algorithm 5 Computational approach for solving the two-phase Stefan problem in level set formulation.

Input: Ω, Γ_{D}, Γ_{N}, Γ_{R}, $\mathcal{S}_h$, $\varphi(t_0)$, $u(t_0)$, $u_{1,2}$, f, κ, λ, ρ, c, L, g_{D}, g_{N}, g_{R}, t_0, t_f, Δt, β_O, β_I, γ.
Output: u_h^n, Γ_h^n, $\vec{V}_h^n$ for $n > 0$.

```
1: procedure COMPUTATIONAL APPROACH FOR THE TWO-PHASE STEFAN PROBLEM.
2:     Initialization (1): $t_n = t_0$, $u_h^n = u(t_0)$, $\varphi_h^n = \varphi(t_0)$, $\Gamma_h^n = \Gamma(t_0)$.
3:     Initialization (2): Define $V^1_{h,u_{\mathrm{D}}}(t_0)$ and initialize $\Omega_{\mathrm{INB}}$ and $\Omega_{\mathrm{ONB}}$ for $\varphi_h^0$.
4:     for $n = 0, \ldots, N_t - 1$ do
5:         Compute $\vec{V}^n_{1,2}$ by evaluating (5.34) with an approach presented in Section 4.4.2.
6:         Assign $t_\varphi = t_0 + n\Delta t$ (current simulation time) and $\varphi_h^{t_\varphi} = \varphi_h^n$.
7:         Compute new level set function $\varphi_h^{n+1}$ with Algorithm 4.
8:         Compute the new discrete interface $\Gamma_h^{n+1}$ from $\varphi_h^{n+1}$.
9:         Define $V^1_{h,g_{\mathrm{D}}}(t_{n+1})$.
10:        Compute $u_h^{n+1}$ by solving (5.60).
11:    end for
12: end procedure
```

to Section 5.1.2 and define the time step size Δt, used for discretizing the heat equation in time, as major time step size and the values $t_n = t_0 + n\Delta t$ with $n \in \{0, \ldots, N_t\}$ as so-called synchronization points. Based on this, we adjust the time step size to $\Delta t_{\mathrm{CFL}} < \Delta t_\varphi$ used for solving the level set problem if necessary, so that the CFL condition(s) are respected. This means we may have to introduce potentially non-equidistant intermediate time steps $t_{n,i}$ in order to reach the (next) synchronization point(s). The described procedure is illustrated in Fig. 5.10. The computational approach to solve the two-phase Stefan problem in level set formulation is presented in Algorithm 5 and parts of the corresponding `UFL` file containing the discretized variational formulation are given in Figure 5.11.

Remark 5.2 (Impact of the CFL conditions on the solution and the consequences when considering complex applications). Please note that a lot of intermediate time steps of size Δt_{CFL} are necessary for solving the level set problem, if we define a rather large major time step size Δt while using the narrow band approach. This can have a huge impact on the solution's accuracy (and hence the convergence behavior of approximation errors) as we neither update the temperature nor the velocity field for these intermediate steps. For considering applications

```
[...]
a = v * u_new * dx + dt * dot( k * grad(v), grad(u_new)) * dx \
  + dt('+') *nitsche('+') * u_new('+') * v('+') * dc(0) \
  + dt('+') *nitsche('+') * u_new('-') * v('-') * dc(0) \
  - dt('+') * k('-') * dot(grad(u_new('-')), n(phi0('-'))) * v('-') * dc(0) \
  + dt('+') * k('+') * dot(grad(u_new('+')), n(phi0('+'))) * v('+') * dc(0) \
  - dt('+') * k('-') * dot(grad(v('-')), n(phi0('-'))) * u_new('-') * dc(0) \
  + dt('+') * k('+') * dot(grad(v('+')), n(phi0('+'))) * u_new('+') * dc(0)

L = v * u_old * dx - dt('+') * gN * v* ds(0)    \
  + dt('+') *nitsche('+') * u_m('+') * v('+') * dc(0) \
  + dt('+') *nitsche('+') * u_m('+') * v('-') * dc(0) \
  - dt('+') * k('-') * dot(grad(v('-')), n(phi0('-'))) * u_m('+') * dc(0) \
  + dt('+') * k('+') * dot(grad(v('+')), n(phi0('+'))) * u_m('+') * dc(0)
[...]
```

Figure 5.11 Part of the UFL implementation of the discretized variational formulation of the two-phase Stefan problem in miXFEM.

as we will do in Sections 5.4 to 5.6, we therefore do not specify a fixed time step size Δt but an interval $[\Delta t_{\min}, \Delta t_{\max}]$ and choose $\Delta t \in [\Delta t_{\min}, \Delta t_{\max}]$ so that the CFl conditions are satisfied.

Remark 5.3 (Fix point iteration scheme)*.* Instead of solving the subproblems in succession, we could also use a fixed point scheme to improve accuracy. Such a scheme is based on iterating the solution of the heat equation and the solution of the level set problem against each other. An implementation approach for such a procedure is presented in [29, Sec. 9].

5.3.3 Validation of the implementation and examples

Similar to the previous section, we validate our method and its implementation using several examples. As the computational approach for solving the coupled two-phase Stefan problem involves different methods, such as the level set toolbox and the schemes to derive a velocity field by evaluating the Stefan condition, there are many parameters that can have an influence on the accuracy of the solution and convergence behavior of the approximation. As a results, several aspects have to be considered:

(SP 1) Since we already validated the level set toolbox, cf. Section 5.1, and the capabilities of miXFEM to solve problems involving stationary discontinuities, see Section 5.2, we now need to validate the XFEM solver for time-dependent problems. Therefore, we first prescribe the interface evolution analytically and reduce the two-phase Stefan problem to an (time-dependent) heat equation with moving interface.

(SP 2) In regards to the computation of the velocity field based on evaluating the Stefan condition, both approaches (DGE and DSCE) presented in Section 4.4.2 need to be tested. Additionally, we have to study the impact of the parameter choice on the solution accuracy and convergence behavior, especially when using the DSCE scheme.

(SP 3) Finally, we investigate the convergence behavior by performing studies for varying time step size Δt and element size h.

For this, we consider the following examples:

5.3.3.1 Example 1: Straight interface

On $\Omega := (0,1)^d$, $d = 2, 3$, with $\Gamma_\mathrm{D} := \{\boldsymbol{x} \in \partial\Omega \,|\, x = 0 \vee x = 1\}$ and $\Gamma_\mathrm{N} := \partial\Omega \backslash \Gamma_\mathrm{D}$, we define the level set function

$$\varphi(\boldsymbol{x}, t) := x_0 + V_{\vec{n}} t - \cos\left(\frac{\theta\pi}{180^\circ}\right) x + \sin\left(\frac{\theta\pi}{180^\circ}\right) y \tag{5.39}$$

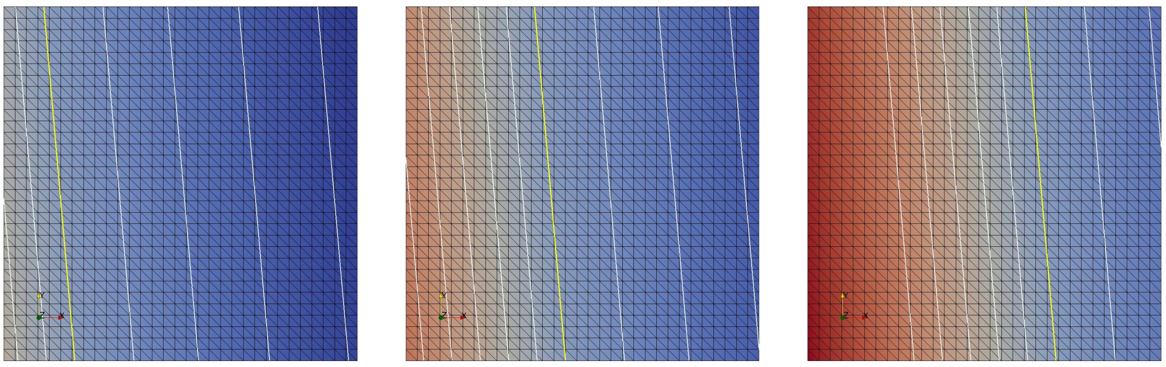
for the time interval $[0,1]$ with $\theta \in [0^\circ, 360^\circ]$. The zero level $\Gamma(t)$ separates Ω into the subdomains $\Omega_1(t) := \{\boldsymbol{x} \in \Omega : \varphi(\boldsymbol{x}, t) < 0\}$ and $\Omega_2(t) := \{\boldsymbol{x} \in \Omega : \varphi(\boldsymbol{x}, t) > 0\}$, $t \in [0,1]$. Now, we choose the data f, g_N, and g_D so that the function

$$u(\boldsymbol{x}, t) := \begin{cases} 2V_{\vec{n}}\varphi(\boldsymbol{x}, t) & \text{on } \Omega_1(t) \\ \exp\left(V_{\vec{n}}\varphi(\boldsymbol{x}, t)\right) - 1 & \text{on } \Omega_2(t) \end{cases} \tag{5.40}$$

is a solution of the Stefan problem given by equations (5.34), (5.36) and (5.38) for $\rho|_{\Omega_i} = c|_{\Omega_i} = \lambda|_{\Omega_i} = 1$, $i = 1, 2$, $u_{1,2} = 0$, and $L = 1$. The solution $u(\cdot, t)$ is visualized for $V_{\vec{n}} = 0.5$, $x_0 = 0.2$, and $\theta = 85^\circ$ in Figure 5.12.

5.3.3.2 Example 2: Straight interface

On $\Omega := (0,1)^d$, $d = 2, 3$, with $\Gamma_\mathrm{D} := \{\boldsymbol{x} \in \partial\Omega \,|\, y = 0 \vee y = 1\}$ and $\Gamma_\mathrm{N} := \partial\Omega \backslash \Gamma_\mathrm{D}$, consider the two-phase Stefan problem (5.36) to (5.38) with $\rho|_{\Omega_i} = c|_{\Omega_i} = 1$, $\lambda_1 := 1$, $\lambda_2 := 2$, $L := 2$, and



(a) u at $t = 0$ (b) u at $t = 0.5$ (c) u at $t = 0.1$

Figure 5.12 Example 1: Different level sets including the zero level set (yellow) of the analytical solution u at different time instants t.

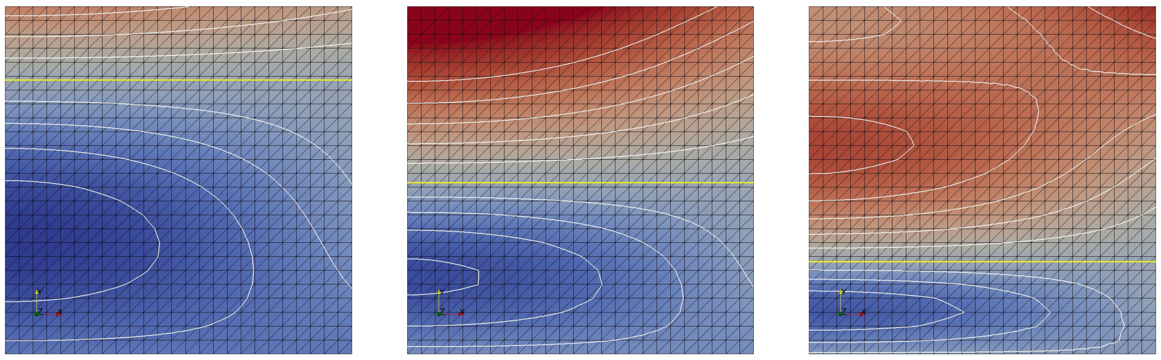

(a) u at $t = 0$ (b) u at $t = 0.15625$ (c) u at $t = 0.3125$

Figure 5.13 Example 2: Different level sets including the zero level set (yellow) of the analytical solution u at different time instants t.

$u_{1,2} := 0$ so that an analytical solution to problem is given by

$$u(\boldsymbol{x},t) = \begin{cases} \cos\left(\frac{\pi x}{2}\right)\sin\left(\frac{\pi\varphi(\boldsymbol{x},t)}{y-\varphi(\boldsymbol{x},t)}\right) + \varphi(\boldsymbol{x},t) & \text{on } \Omega_1(t) \\ \cos\left(\frac{\pi x}{2}\right)\sin\left(\frac{\pi\varphi(\boldsymbol{x},t)}{2(y-\varphi(\boldsymbol{x},t))}\right) + \frac{1}{2}\varphi(\boldsymbol{x},t) + e^y + \frac{11}{11t+5} & \text{on } \Omega_2(t) \end{cases}, \tag{5.41}$$

for $[t_0, t_f] := [0, 5 \cdot 2^{-3}]$ and

$$\varphi(\boldsymbol{x},t) := y - \ln\left(\frac{11}{11t+5}\right). \tag{5.42}$$

The corresponding interface represented by the zero level set $\Gamma(t)$, separating Ω into the subdomains $\Omega_1(t)$ ($\varphi < 0$) and $\Omega_2(t)$ ($\varphi > 0$), is a straight horizontal line ($d = 2$) or plane ($d = 3$) moving downwards. The source term f for the right-hand-side, the boundary functions g_D and g_N, and the initial conditions chosen with respect to (5.41). The analytical solution $u(\boldsymbol{x},t)$ with interface $\Gamma(t)$ is shown for different points in time in Figure 5.13.

5.3.3.3 Example 3: Circular interface

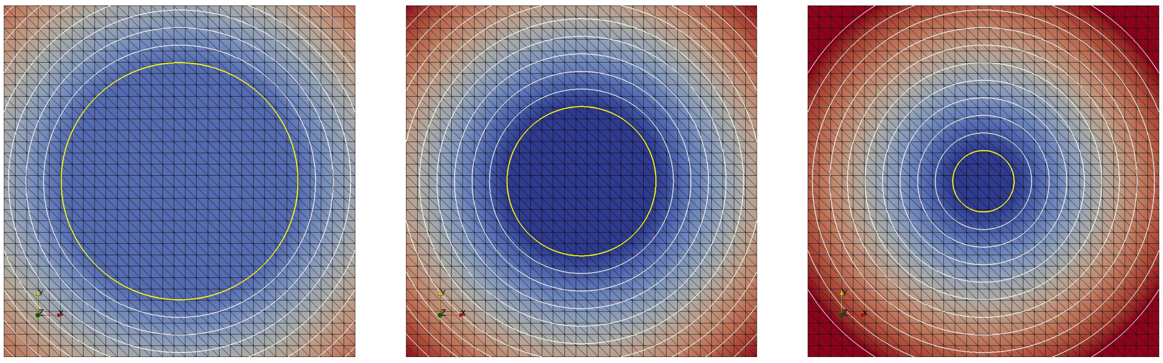
Now, let $\Omega := (-2,2)^d$, $d = 2,3$ be a domain with $\Gamma_\text{D} := \partial\Omega$. For $t \in [0,1]$, we define the level set function

$$\varphi(\boldsymbol{x},t) := \|\boldsymbol{x}\|_2 - r_0 - t \tag{5.43}$$

which separates Ω into $\Omega_1(t)$ ($\varphi < 0$) and $\Omega_2(t)$ ($\varphi > 0$) and chose the data f and g_D such that

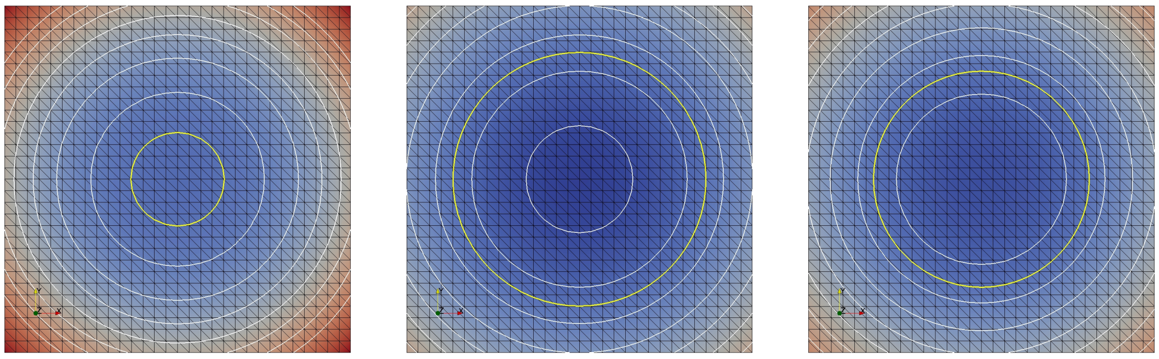
$$u(\boldsymbol{x},t) := \begin{cases} 0 & \text{on } \Omega_1(t) \\ \|\boldsymbol{x}\|_2 - r_0 - t & \text{on } \Omega_2(t) \end{cases} \tag{5.44}$$

is a solution of problem (5.36) to (5.38) for $\rho|_{\Omega_i} = c|_{\Omega_i} = \lambda|_{\Omega_i} = 1$, $i = 1,2$, $u_{1,2} = 0$, and $L = 1$. A characteristic of this example is that we have $V_{\tilde{n}} = 1$ and the solution $u(\boldsymbol{x},t)$ with $r_0 = 0.15$ is shown for different points in time in Figure 5.14



(a) u at $t = 0$ (b) u at $t = 0.5$ (c) u at $t = 1$

Figure 5.14 Example 3: Different level sets including the zero level set (yellow) of the analytical solution u at different time instants t.



(a) u at $t = 0$ (b) u at $t = 0.375$ (c) u at $t = 0.75$

Figure 5.15 Example 4: Different level sets including the zero level set (yellow) of the analytical solution u at different time instants t.

5.3.3.4 Example 4: Circular interface

Last but not least, consider on $\Omega := (-1, 1)^2$ with $\Gamma_{\mathrm{N}} := \partial\Omega$ for $t \in [0, \frac{3}{4}]$ the level set function

$$\varphi(\boldsymbol{x}, t) := R^2(t) - \|\boldsymbol{x}\|^2 \tag{5.45}$$

with $R(t) := R_0 + \frac{1}{2}\sin(\pi t)$ separating Ω into $\Omega_1(t)$ ($\varphi < 0$) and $\Omega_2(t)$ ($\varphi > 0$). The corresponding zero level set $\Gamma(t)$ is a $d-$sphere with radius $R(t)$ that is centered at the origin and expands for $t \leq 0.5$ and then shrinks again. Now we choose the right-hand-side term f, the Neumann boundary function g_{N}, and the initial conditions such that

$$u(\boldsymbol{x}, t) = \begin{cases} A\left(\|\boldsymbol{x}\|_2^2 - R(t)^2\right) & \text{on } \Omega_1(t) \\ A\left(\|\boldsymbol{x}\|_2^2 - R(t)^2\right) - R(t) - \dot{R}(t)\left(\|\boldsymbol{x}\|_2 - R(t)\right) + \frac{\dot{R}(t)\left(\dot{R}(t) + \frac{1}{R(t)}\right)}{2\left(\|\boldsymbol{x}\|_2 - R(t)\right)^2} & \text{on } \Omega_2(t) \end{cases} \tag{5.46}$$

is a solution the two-phase Stefan problem for $\rho|_{\Omega_i} = c|_{\Omega_i} = \lambda_1 = \lambda_2 = 1$, $L = 1$ and $u_{1,2} = 0$. Please note that we have to choose $A > \frac{\dot{R}(t)}{2R(t)}$ to ensure $u < 0$ on $\Omega_1(t)$ and $u > 0$ on $\Omega_2(t)$. The solution $u(\boldsymbol{x}, t)$ is visualized for $R_0 = 0.3$ and $A = 4.1$ for different points in time in Figure 5.15.

5.3.4 Results

The results of the validation tests and parameter studies listed in (SP 1) to (SP 3) are given in our publications [93, 102, 103, 105]. We give a brief overview about our findings in the following list:

(Result 1) When reducing the problem to a heat equation with prescribed interface evolution, the XFEM solver converges with the optimal rates in (almost) every example, see [102].

(Result 2) Both approaches for computing a velocity field provide satisfying results. Although, it has to be noted that the approximation quality of the DCSE method is superior to the DGE approach in most of the examples. Unfortunately, the DSCE method can be sensitive to the step size parameter δ, see [93].

(Result 3) When considering the completely coupled problem, where the velocity is derived from evaluating the Stefan condition using either the DGE or DCSE method, the convergence rates are only suboptimal in some situations, see [93, 103]. Further studies showed that this is a result of the imprecision when computing the velocity field. Therefore, the convergence rates are worse if the temperature gradients differ significantly in the vicinity of the interface.

(Result 4) However, depending on the choice of δ, we obtain satisfying convergence rates in every example when considering the coupled problem. As the approximation errors of some of the considered examples are heavily dominated by either the time or the spatial error, it is necessary to choose a sufficiently fine discretization.

(Result 5) Moreover, it has to be noted that the narrow band approach can only be used for suitable parameters that satisfy the CFL conditions. Otherwise, intermediate time steps are introduced, which has an impact on the observed convergence rates.

(Result 6) The stabilization parameter λ has to be sufficiently large. It is recommended to use a value that is at least one magnitude larger than required for the stability, see [50]. For larger values, the parameters have no significant impact on the convergence behavior.

We exemplarily show the results of the convergence study for the example Section 5.3.3.2.

5.3.5 Convergence analysis for Section 5.3.3.2

Following the theory presented in [107], we define the L^2-error

$$\|u - u_h\|_{L^\infty(L^2)} := \max_{t\in(t_0,t_f)} \|u(\cdot,t) - u_h(\cdot,t)\|_{L^2(\Omega)}, \tag{5.47}$$

which can be bounded by

$$\|u - u_h\|_{L^\infty(L^2)} \leq c_0 \left(c_1\Delta t + c_2(\Delta t)^2 + c_3 h^{m+1}\right), \tag{5.48}$$

and the (semi) H^1-error

$$\|\nabla u - \nabla u_h\|_{L^2(L^2)} := \sqrt{\int_{t_0}^{t_f} \|\nabla u(\cdot,t) - \nabla u_h(\cdot,t)\|^2_{L^2(\Omega)}\, dt}, \tag{5.49}$$

which can be bounded by

$$\|\nabla u - \nabla u_h\|_{L^2(L^2)} \leq c_0 \left(c_1\Delta t + c_2(\Delta t)^2 + c_3 h^{m+1} + c_4 h^m\right). \tag{5.50}$$

Using the implicit Euler scheme for time discretization and $m = 1$ as polynomial degree for the eXtended approximation space V_h that is based on the (conventional) function space $V^m_{\mathrm{cg},h}$, the optimal orders of convergence are

$$\|u - u_h\|_{L^\infty(L^2)} = \mathcal{O}(\Delta t + h^2) \tag{5.51}$$

and

$$\|\nabla u - \nabla u_h\|_{L^2(L^2)} = \mathcal{O}(\Delta t + h). \tag{5.52}$$

We consider the two-phase Stefan problem for the setting as specified in Section 5.3.3.2 on a regular triangulations $\mathcal{S}_h$ for several mesh sizes $h = \frac{1}{N_{\mathrm{el}}}\sqrt{2}$ and time step sizes Δt. To show that the method provides the optimal convergence orders in both the mesh size and the time, we have to choose the counterpart sufficiently small, cf. Equations (5.51) and (5.52).

Choosing the DCSE method with $\delta = 0.5$, we first present the convergence behavior in space for $N_{\mathrm{el}} = \{2 \times 10^2, 2 \times 15^2, 2 \times 20^2, 2 \times 25^2 2 \times 30^2\}$ elements while choosing the time step size $\Delta t = 2^{-8}$. The results for the $\|u - u_h\|_{L^\infty(L^2)}$ and $\|\nabla u - \nabla u_h\|_{L^\infty(L^2)}$ errors and the estimated orders of convergence are given in Table 5.3 and the corresponding values for the difference $\|\nabla u - \nabla u_h\|_{L^2(L^2)}$ are shown in Table 5.4. With respect to the time variable, the errors and convergence behavior of the $\|u - u_h\|_{L^\infty(L^2)}$ error are shown in Table 5.5. As we can see in all tables, the optimal orders can be achieved.

N_{el}	$\|u - u_h\|_{L^\infty(L^2)}$	eoc	$\|u - u_h\|_{L^\infty(H^1)}$	eoc
2×10^2	0.217337	-	2.02533	-
2×15^2	0.0769138	2.5619	1.06039	1.6036
2×20^2	0.0463953	1.7571	0.806164	0.9562
2×25^2	0.0284696	2.1886	0.61892	1.1872
2×30^2	0.0184754	2.3716	0.481172	1.3826
2×35^2	0.0138268	1.8802	0.415048	0.9602

Table 5.3 Approximation errors and estimated order of convergence for the example presented in Section 5.3.3.2 using $\Delta t = 2^{-8}$.

N_{el}	$\|\nabla u - \nabla u_h\|_{L^2(L^2)}$	eoc
2×10^2	0.496109	-
2×15^2	0.305875	1.1928
2×20^2	0.220111	1.1438
2×25^2	0.168133	1.2072
2×30^2	0.136743	1.1334
2×35^2	0.115692	1.0845

Table 5.4 Approximation errors and estimated order of convergence for the example presented in Section 5.3.3.2 using $\Delta t = 2^{-8}$.

Δt	$\|u - u_h\|_{L^\infty(L^2)}$	eoc
2^{-3}	0.9288	-
2^{-4}	0.2844	1.707
2^{-5}	0.1514	0.909
2^{-6}	0.0506	1.580
2^{-7}	0.0263	0.940

Table 5.5 Approximation errors and estimated order of convergence for the example presented in Section 5.3.3.2 using $N_{\text{el}} = 44$.

5.4 Laser welded hybrid joints

The first application with industrial background that we consider is the process to generate laser welded hybrid overlap joints, see Section 1.1.2. Recall that for the generation of hybrid overlap joints, a steel and an aluminum sheet are arranged with a small overlapping region, cf. Figure 5.16. A laser heat source is applied to the steel sheet and the energy is conducted into the aluminum sheet. Due to the significantly lower melting temperature of aluminum compared to steel, the aluminum melts and wets the steel. After switching off the laser and the completion of the solidification process, a hybrid joint is generated.

The described process can be modeled mathematically in a continuum mechanical framework. The corresponding model has to (at least) include the heat equation on all subdomains (steel, and aluminum), the two-phase Stefan problem, and the Navier-Stokes equations for incompressible fluids and a free capillary surface [13]. The Stefan problem is used to describe phase changes within the aluminum, and the Navier-Stokes equations model the fluid dynamics in the molten part. At outer and inner domain boundaries, conditions for laser heating, thermal conduction, and radiation described by the Stefan-Boltzmann law have to be included, see, e.g., [9]. In addition, we need to include wetting conditions. A full 2D mathematical cross-section model for generating laser welded hybrid overlap joints can be found in [13].

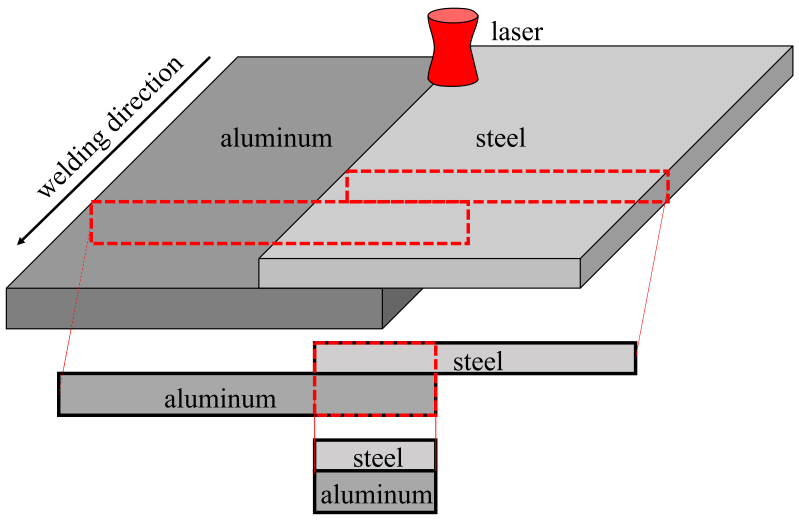


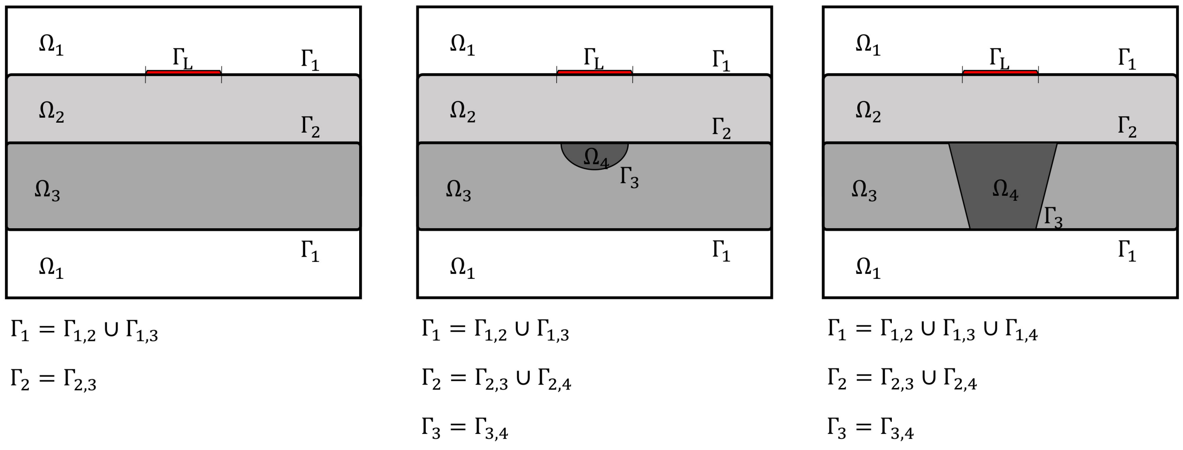
Figure 5.16 Generation of laser-welded hybrid joints.

5.4.1 Modeling the process with the hierarchical level set method

While considering the complete process model and performing simulations for different process configurations is beyond the scope of this thesis, we use this process to demonstrate the enhanced modeling possibilities when using the presented method and the capability of `miXFEM` to handle topology changes and the nucleation of new subdomains and corresponding interfaces. Moreover, we extend previous model approaches of the process by taking into account not only the steel and aluminum sheets but also the surrounding gas atmosphere. However, to simplify the problem, we neglect the fluid dynamics and geometric changes of the workpiece and only consider parts of the melting process instead.

Modeling approach

In brief, the fundamental idea of our simplified model is to use three hierarchically ordered level set functions to decompose a hold-all domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, into four subdomains. The initial configuration and different process stages at some points in time are visualized for $d = 2$ in Figure 5.17. Given a time interval $[t_0, t_f]$ and an initial geometry $\Gamma_1(t_0)$, we first define $\varphi_1 \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$ as signed distance function to $\Gamma_1(t_0)$ such that its zero level set $\Gamma_1(t_0)$ separates the surrounding atmosphere ($\varphi_1 < 0$) from the workpiece ($\varphi_1 > 0$). The workpiece is decomposed by the zero level set Γ_2 of a signed distance function $\varphi_2 \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$ of less hierarchical order into steel ($\varphi_2 < 0$) and aluminum ($\varphi_2 > 0$). On a comparatively small part $\Gamma_\mathrm{L} \subset \Gamma_1$, a (laser) heat source is applied, introducing energy into the workpiece. On the remaining part of the boundary $\Gamma_1 \setminus \Gamma_\mathrm{L}$ and across the boundary Γ_2 separating steel and aluminum, we assume continuity of the heat flux and temperature $u \in \mathcal{C}^0(\bar{\Omega} \times [t_0, t_f])$. Furthermore, since we assume that the temperature u only exceeds the melting temperature $u_{3,4}$ of aluminum but not the melting temperature of steel, we only have to



(a) Situation at time t_0. (b) Nucleation of molten aluminum. (c) Melt pool evolves and penetrates aluminum sheet.

Figure 5.17 Different process stages of the hybrid welding process.

take into account phase changes within the aluminum. Therefore, we represent the solid-liquid interface by the zero level set Γ_3 of another hierarchically subordinated signed distance function $\varphi_3 \in C^1\left(\Omega \times \left((t_0, t_f) \setminus \mathcal{T}\right)\right) \cap C^0\left(\bar{\Omega} \times \left([t_0, t_f] \setminus \mathcal{T}\right)\right)$ which separates the aluminum into a solid part ($\varphi_3 < 0$) and a liquid part ($\varphi_3 > 0$). As before, we model the evolution of $\Gamma_3(t)$ in time by the Stefan condition. However, while the (active parts of the) zero level sets Γ_1 and Γ_2 of the functions φ_1 and φ_2 are present for all $t \in [t_0, t_f]$, φ_3, or Γ_3 to be more precise, only exists if $u(\boldsymbol{x}, t) > u_{3,4}$ for some $\boldsymbol{x} \in \Omega \setminus (\Omega_1 \cup \Omega_2)$, $t \in [t_0, t_f]$. For example at the initial state we have $u(\boldsymbol{x}, t) < u_{3,4}$ for all $\boldsymbol{x} \in \Omega$; hence, we have to detect the times and places when and where topological changes, such as nucleation or dissolution of melt, occur within our model and the computational method. Unfortunately, this also means that we have to consider such points in time within our model.

The fundamental idea to detect topology changes and the times when they occur is as follows: Firstly, we introduce the function $\check{\varphi}_3(\boldsymbol{x}, t) := u(\boldsymbol{x}, t) - u_{3,4}$[¶] whose hierarchically active zero level set is

$$\check{\Gamma}_3(t) = \{\boldsymbol{x} \in \Omega \,:\, \check{\varphi}_3(\cdot, t) = 0 \wedge \varphi_i(\boldsymbol{x}, t) > 0,\, i = 1, 2\} \quad \text{for } t \in [t_0, t_f]. \tag{5.53}$$

With this, we define a set of times where the topology changes by

$$\mathcal{T}_1 := \Big\{t \in [t_0, t_f] \,:\, \check{\Gamma}_3(t) \neq \Gamma_3^-(t) \wedge \operatorname{meas}_{d-1}(\check{\Gamma}_3(t) \setminus \Gamma_3^-(t)) \geq \epsilon > 0, \tag{5.54}$$

$$\text{with } 1 \gg \epsilon \in \mathbb{R}^+\Big\}, \tag{5.55}$$

with the notation $\Gamma_3^-(t) := \lim\limits_{\tau \in [t_0, t_f] \nearrow t} \Gamma_3(\tau)$ and $\Gamma_3(t_0) = \emptyset$. To also model topological changes where small connected volumes of melt, whose volume is below a given tolerance, dissolve, we

[¶]Please note that we cannot use $\check{\varphi}_3(\boldsymbol{x}, t)$ as level set function since we may not have enough regularity, cf. Remark 5.1.

separate the domain $\Omega_4(t) := \{\boldsymbol{x} \in \Omega \,:\, \varphi_i(\boldsymbol{x}, t) > 0,\, i = 1, 2, 3\}$ into its connected subdomains $\omega(t)$ with $\bigcup_{j=1}^{N_{\mathrm{sub}}(t)} \omega_j(t) = \Omega_4(t)$, $t \in [t_0, t_f]$. Based on this, we define the index set

$$\mathcal{L}(t) := \left\{ j \in \{1, \dots, N_{\mathrm{sub}}(t)\} \,:\, \mathrm{meas}_{d-1}(\partial\omega_j(t)) \le \delta < \epsilon,\, \delta \in \mathbb{R}^+ \right\} \quad \text{for } t \in [t_0, t_f], \tag{5.56}$$

the boundaries of the dissolved domain

$$\Xi(t) := \bigcup_{l \in \mathcal{L}(t)} \partial\omega_l(t) \setminus \partial\Omega, \tag{5.57}$$

and the set of times, when the topology changes due to dissolution by

$$\mathcal{T}_2 := \left\{ t \in [t_0, t_f] \,:\, \mathcal{L}^-(t) \neq \emptyset \right\}. \tag{5.58}$$

The total set of points in time when the topology changes is then given by $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$[‖]. For all $t \in \mathcal{T}$, we define $\varphi_3(\boldsymbol{x}, t)$ as corresponding signed distance function to

$$\Gamma_3(t) := \check{\Gamma}_3^-(t) \setminus \Xi^-(t). \tag{5.59}$$

for all $\boldsymbol{x} \in \Omega \setminus \Omega_1(t)$ and extend it to Ω[**].

Remark 5.4. By modeling the workpiece geometry by the zero level set of $\varphi_1 \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$, we would smoothen the corners of the sheets when considering the complete geometry. However, doing so is actually advantageous as it supports the wetting of the steel sheet. In our previous simulations using the conventional finite element method, we therefore had to add small wetting segments, see [13].

Remark 5.5 (A comment on the process configuration). Within our modeling, we assume that the temperature never reaches the melting temperature of steel. If we want to drop this assumption, we have to include an additional level set function which decomposes the molten and the solid part of the steel. However, introducing this function would cause a very different modeling setup since it would need to be of higher hierarchical order than the current functions φ_2 and φ_3.

Coupled model

Altogether, the simplified model for the simulation of the described process is as follows: Assuming that for $t \in [t_0, t_f]$ the workpiece geometry $\Gamma_1(t)$ and workpiece material composition $\Gamma_2(t)$ as well as the corresponding signed distance functions $\varphi_1, \varphi_2 \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$ are a priori known, and that $\Gamma_i(t)$, $i = 1, 2$, is sharp and sufficiently smooth for all $t \in [t_0, t_f]$, then our

[‖]Note that the set $\mathcal{T}_1$ and $\mathcal{T}_2$ are not necessarily disjoint since nucleation and dissolution can occur at different positions at the same time.

[**]In practice, this means that we define φ_3 as signed distance function to a (reduced and smoothed) zero level set $\check{\Gamma}_3 \subseteq \tilde{\Gamma}_3$ of $\check{\varphi}_3$ (which is defined on the hold-all Ω).

reduced coupled model of the laser welding process for hybrid joints reads: Find the solid-liquid interface Γ_3 which is the zero level set of $\varphi_3 \in C^1\left(\Omega \times \left((t_0,t_f)\setminus\mathcal{T}\right)\right) \cap C^0\left(\bar{\Omega} \times \left([t_0,t_f]\setminus\mathcal{T}\right)\right)$ and the temperature distribution $u \in \mathcal{C}^0(\bar{\Omega}\times[t_0,t_f])$ which is sufficiently smooth, i.e. $u(\cdot,t)_{|\Omega_i} \in \mathcal{C}^2(\Omega_i(t))$ and $\partial_t u(\cdot,t) \in C^0\left(\bigcup_{i=1}^4 \Omega_i(t)\right)$ such that

$$\begin{cases} \rho c \dfrac{\partial u}{\partial t} - \nabla\cdot(\lambda\nabla u) = 0 & \text{in } \bigcup_{i=1}^4 \Omega_i(t),\ t\in(t_0,t_f), & (5.60a)\\ u = g_{\mathrm{D}} & \text{on } \Gamma_{\mathrm{D}}\times(t_0,t_f], & (5.60b)\\ -\lambda\nabla u\cdot\vec{n} = g_{\mathrm{N}} & \text{on } \Gamma_{\mathrm{N}}\times(t_0,t_f], & (5.60c)\\ -\lambda\nabla u\cdot\vec{n} = g_{\mathrm{R}}(u) & \text{on } \Gamma_{\mathrm{R}}\times(t_0,t_f], & (5.60d)\\ [\![u]\!] = 0 & \text{on } \Gamma_1(t), & (5.60e)\\ [\![\lambda\nabla u]\!]\cdot\vec{n}_1 = g_1 & \text{on } \Gamma_1(t), & (5.60f)\\ [\![u]\!] = 0 & \text{on } \Gamma_2(t), & (5.60g)\\ [\![\lambda\nabla u]\!]\cdot\vec{n}_2 = 0 & \text{on } \Gamma_2(t), & (5.60h)\\ u(\cdot,t) = u_{3,4} & \text{on } \Gamma_3(t), & (5.60i)\\ u(\cdot,t_0) = u_0 & \text{in } \bigcup_{i=1}^4 \Omega_i(t_0), & (5.60j) \end{cases}$$

$$\begin{cases} (\lambda_3\nabla u_3 - \lambda_4\nabla u_4)\cdot\vec{n}_3 = \rho L\vec{V}_{3,4}\cdot\vec{n}_3 & \text{on } \Gamma_3(t),\ t\in[t_0,t_f]\setminus\mathcal{T}, & (5.61a)\\ \dfrac{\partial\varphi_3}{\partial t} + \vec{V}_{3,4}\cdot\nabla\varphi_3 = 0 & \text{in } \Omega\times[t_0,t_f]\setminus\mathcal{T}, & (5.61b) \end{cases}$$

$$\begin{cases} \check{\varphi}_3(\cdot,t) := u(\cdot,t) - u_{3,4} & \text{in } \Omega\times[t_0,t_f], & (5.62a)\\ \Gamma_3(t) := \{\boldsymbol{x}\in\Omega : \check{\varphi}_3(\boldsymbol{x},t) = 0 \wedge \varphi_i(\boldsymbol{x},t) > 0, i=1,2\}\setminus\Xi^-(t) & t\in\mathcal{T}, & (5.62b)\\ \varphi_3(\cdot,t) := \begin{cases} -\min\limits_{\boldsymbol{y}\in\Gamma_3(t)} \|\boldsymbol{x}-\boldsymbol{y}\|_2, & \text{if } \boldsymbol{x}\in\Omega_3\\ \min\limits_{\boldsymbol{y}\in\Gamma_3(t)} \|\boldsymbol{x}-\boldsymbol{y}\|_2, & \text{if } \boldsymbol{x}\in\Omega_4\ ,\\ \text{smooth} & \text{else} \end{cases} & \text{in } \Omega(t),\ t\in\mathcal{T}, & (5.62c) \end{cases}$$

holds for given sufficiently smooth data. The function g_{D} is usually defined as ambient temperature, g_{R} describes cooling, cf. equation (5.29), and g_{N} is usually chosen to be zero which corresponds to isolation. The outer unit normals $\vec{n}_i(t,\boldsymbol{x})$ to $\Gamma_i(t)$ are pointing from Ω_i into Ω_l, $1 \le i < l \le 4$.

On the workpiece boundary $\Gamma_1(t)$, we define

$$g_1(\boldsymbol{x},t) = \begin{cases} g_{\mathrm{L}}(\boldsymbol{x},t) & \text{if } \boldsymbol{x}\in\Gamma_{\mathrm{L}}(t)\subset\Gamma_1(t)\\ 0 & \text{if } \boldsymbol{x}\in\Gamma_1(t)\setminus\Gamma_{\mathrm{L}}(t) \end{cases}, \tag{5.63}$$

as interface condition, where

$$g_{\mathrm{L}}(\boldsymbol{x},t) = I_0 \left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{L}}(t)\|}{r_{\mathrm{L}}(\boldsymbol{x})} \right)^2 \cdot \exp\left(- \frac{2\,\|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{L}}(t)\|^2}{r_{\mathrm{L}}(\boldsymbol{x})^2} \right) \tag{5.64}$$

includes a laser heat source located at $\boldsymbol{x}_{\mathrm{L}}(t)$ with a Gaussian-like distributed intensity profile which is applied on a small subset $\Gamma_{\mathrm{L}}(t) \subset \Gamma_1(t)$ of the workpiece boundary. Therein,

$$I_0 = \frac{2P_{\mathrm{L}}}{r_{\mathrm{f0}}^2 \pi} \tag{5.65}$$

is the intensity maximum, with P_{L} denoting the laser power and r_{L} is the laser beam radius in focus height z_0 which depends, among others, on the wave length λ_{L} of the laser beam and $r_{\mathrm{L}}(\boldsymbol{x})$ is the current laser beam width

$$r_{\mathrm{L}}(\boldsymbol{x}) = r_{\mathrm{L},0} \left(1 + \left(\frac{z - z_0}{z_{\mathrm{Ray}}} \right)^2 \right)^{\frac{1}{2}} \tag{5.66}$$

which depends on the Rayleigh length z_{Ray}.

5.4.2 Discretization and computational approach

On first glance, the problem given by (5.60) to (5.61b) is much more complicated than the problem discussed in Section 5.3 as it involves more domains and level set functions as well as different conditions and topological changes. However, we can use the same decoupling strategy to break down this problem into subproblems due to the obvious analogy of the problems. The arising subproblems are then discretized using the same approaches as presented in Section 5.3.2. However, we now have to extended the computational scheme to detect and consider topological changes.

For doing so, we extend the computational scheme along equations (5.62a) to (5.62c) in a straight-forward way: First of all, we drop the idea of introducing major and intermediate time steps, but always compute the time step size based on the CFL conditions. Moreover, instead of using the level set toolbox to capture the evolution of the solid-liquid interface, we first evaluate the temperature u^n on the whole domain for each time step t_n and define $\check{\varphi}_3^n$, compute its zero level set Γ_3^n, and introduce the corresponding signed-distance function φ_3^n. Only after doing so, we compute the velocity field $\vec{V}_{3,4}^n$ based on the Stefan condition and determine the evolution of Γ_3^n to Γ_3^{n+1} by solving the level set problem given by (5.61b). As we do not know if the topology changes a priori, we construct a new φ_3 for each time step which corresponds to a reinitialization step in some way.

Algorithm 6 Computational approach for the welding process for hybrid joints.

Input: Ω, Γ_{D}, Γ_{N}, Γ_{R}, $\mathcal{S}_h$, $\varphi_1(t_0)$, $\varphi_2(t_0)$, $u(t_0)$, $u_{1,2}$, f, κ, λ, ρ, c, L, g_{D}, g_{N}, g_{R}, t_0, t_f, $\Delta t_{\max}$, $\Delta t_{\min}$, β_O, β_I, γ.
Output: u_h^n, $\Gamma_{3,h}^n$, $\vec{V}_{3,4,h}^n$ for $n > 0$.

```
1:  procedure SIMULATION OF THE WELDING PROCESS FOR HYBRID JOINTS.
2:      Initialization: $t_n = t_0$, $u_h^n = u(t_0)$, $\varphi_{i,h}^n = \varphi_i(t_0)$, $\Gamma_{i,h}^n = \Gamma_i(t_0)$, $i = 1, 2$.
3:      while $t_n < t_f$ do
4:          Define $\check{\varphi}_{3,h}^n(\boldsymbol{x}) := u_h^n(\boldsymbol{x}) - u_{3,4}$ for $\boldsymbol{x} \in \Omega$.
5:          if $\Gamma_{3,h}^n := \left\{\boldsymbol{x} \in \Omega \,:\, \varphi_{i,h}^n(\boldsymbol{x}) \geq 0, i = 1, 2, \wedge \check{\varphi}_{3,h}^n(\boldsymbol{x}) = 0\right\} \neq \emptyset$ then
6:              Define $\varphi_{3,h}^n(\boldsymbol{x})$ as signed distance function to $\Gamma_{3,h}^n$ and extend it to $\Omega$.
7:              if the narrow band method is applied then
8:                  Initialize $\Omega_{3,\mathrm{INB}}$ and $\Omega_{3,\mathrm{ONB}}$ with $\varphi_{3,h}^n$.
9:              end if
10:             Compute $\vec{V}_{3,4}^n$ by evaluating (5.61a) with an approach presented in Section 4.4.2.
11:             Compute $\Delta t := \min\{\max\{\Delta t_{\mathrm{CFL}}, \Delta t_{\min}\}, \Delta t_{\max}, t_f - t_n\}$.
12:             $t_{n+1} := t_n + \Delta t$.
13:             Compute new level set function $\varphi_{3,h}^{n+1}$ by solving (5.61b).
14:             Compute the new discrete interface $\Gamma_{3,h}^{n+1}$ from $\varphi_{3,h}^{n+1}$.
15:         else
16:             Compute $\Delta t := \min\{\Delta t_{\max}, t_f - t_n\}$.
17:             $t_{n+1} := t_n + \Delta t$.
18:         end if
19:         Define $V_{h,g_{\mathrm{D}}}^1(t_{n+1})$.
20:         Compute $u_h^{n+1}$ by solving (5.60).
21:     end while
22: end procedure
```

In addition, we now have not only an evolving domain but also the configuration of the interfaces $\Gamma_1(t)$ and $\Gamma_2(t)$ changes. While initially we have

$$\Gamma_1(t_0) := \Gamma_{1,2}(t_0) \cup \Gamma_{1,3}(t_0) \tag{5.67}$$

and

$$\Gamma_2(t_0) := \Gamma_{2,3}(t_0), \tag{5.68}$$

cf. Figure 5.17(a), new interfaces arise due to the nucleation of the melt and the solid-liquid interface $\Gamma_3(t) = \Gamma_{3,4}(t)$, see Figure 5.17(b). Due to the evolution of the melt, we can end up with

$$\Gamma_1(t) := \Gamma_{1,2}(t) \cup \Gamma_{1,3}(t) \cup \Gamma_{1,4}(t) \tag{5.69}$$

and

$$\Gamma_2(t) := \Gamma_{2,3}(t) \cup \Gamma_{2,4}(t) \tag{5.70}$$

for some $t \in (t_0, t_f]$ as visualized in Figure 5.17(c). To take this into account and to be able to impose different boundary conditions on the each part $\Gamma_{i,l}$ of Γ_i, we have to include all possible combinations within our discrete variational formulation implemented in `UFL`. Using `miXFEM`'s

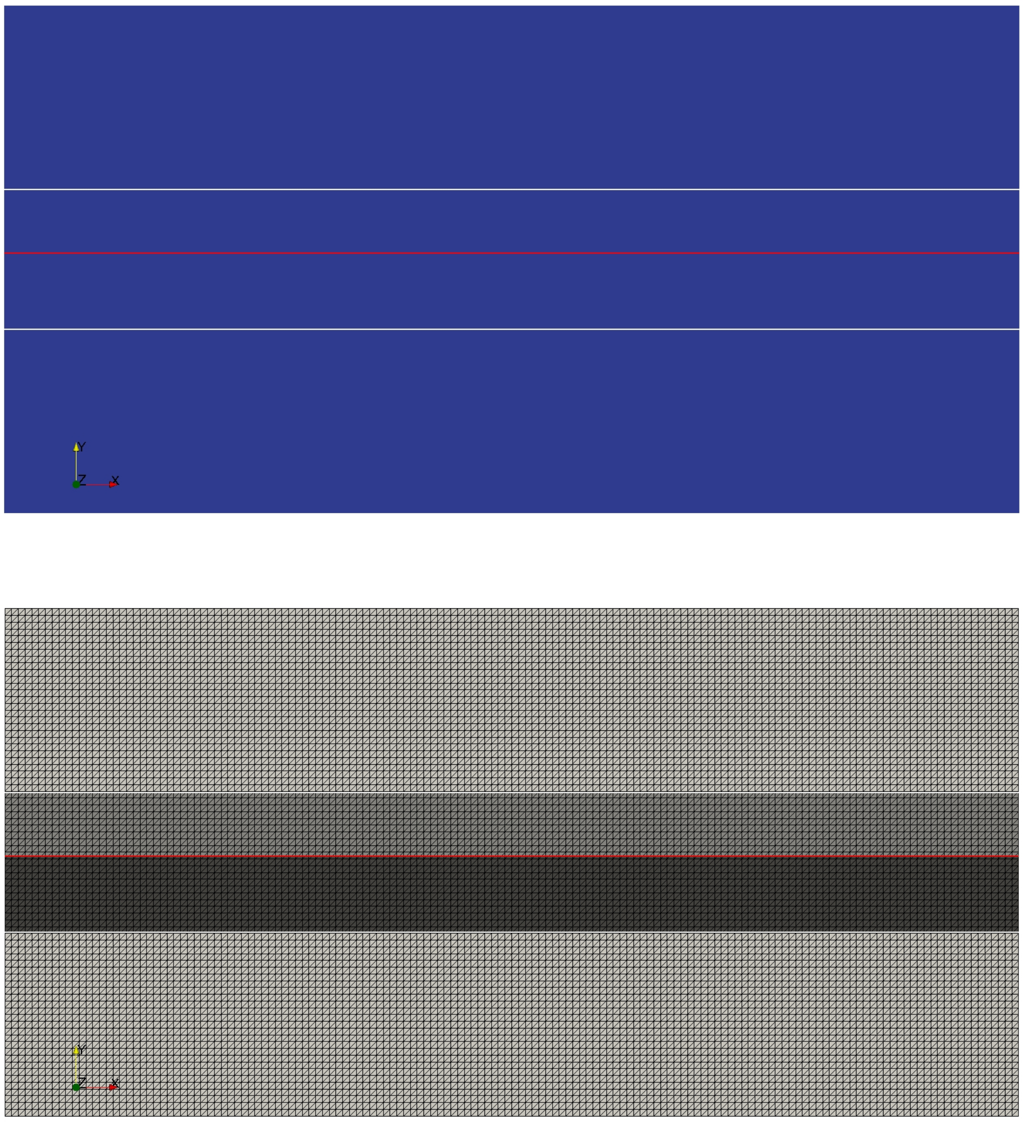
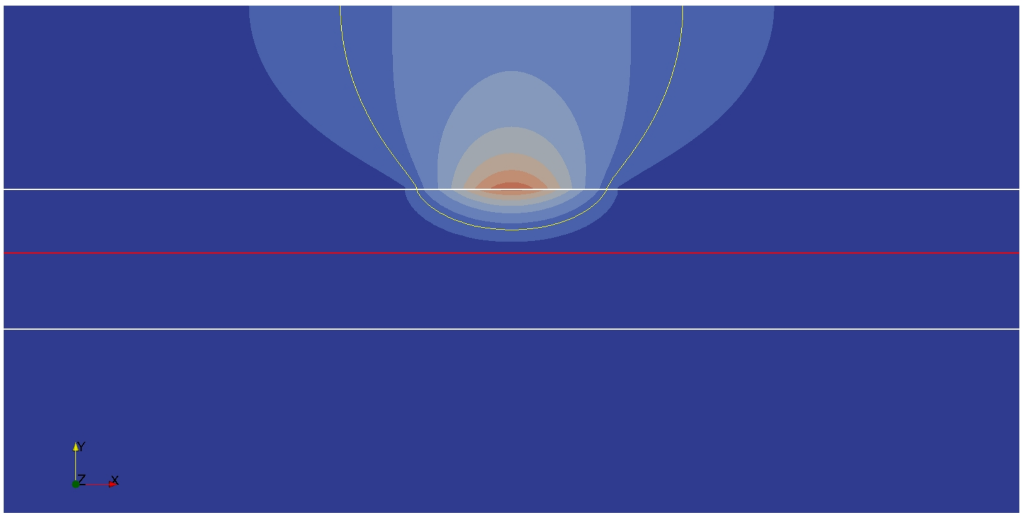


Figure 5.18 Initial state: Steel and aluminum sheet overlap. The boundary Γ_1 between air and workpiece is shown in white, materials are separated by Γ_2 visualized in red.

domain id mapping concept, we then can impose different interface conditions at each part $\Gamma_{i,l} \subseteq \Gamma_i$. The complete computational approach is presented in Algorithm 6.

5.4.3 Results

We now present simulation results on a 2D cross-section domain for the generation of a overlap hybrid joint using laser welding consisting of an aluminum 3.2315 sheet with thickness 1.2 mm and a steel 1.0330 sheet with thickness 1.0 mm without using shielding gas. The material parameters of both sheets and the surrounding atmosphere are specified in Table 5.6 and the process parameters are given in Table 5.7.



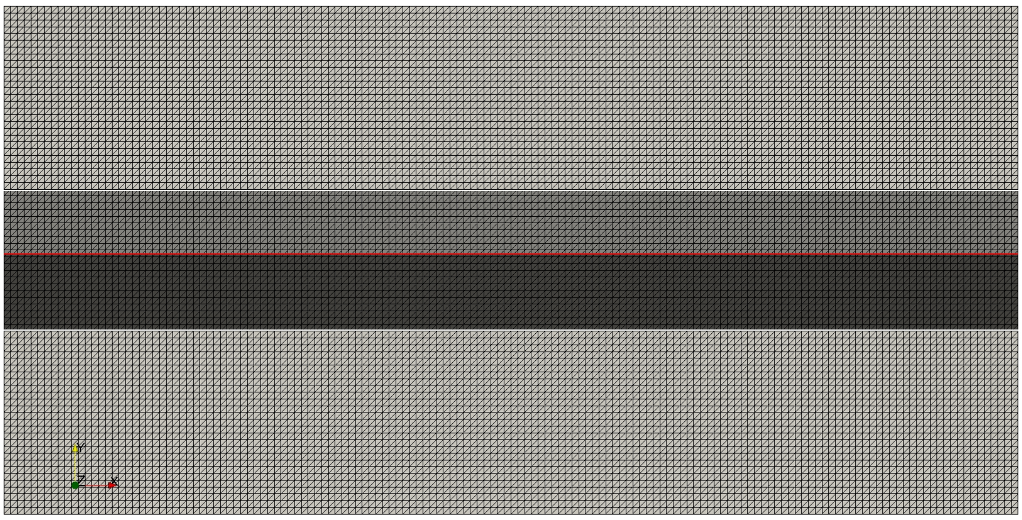


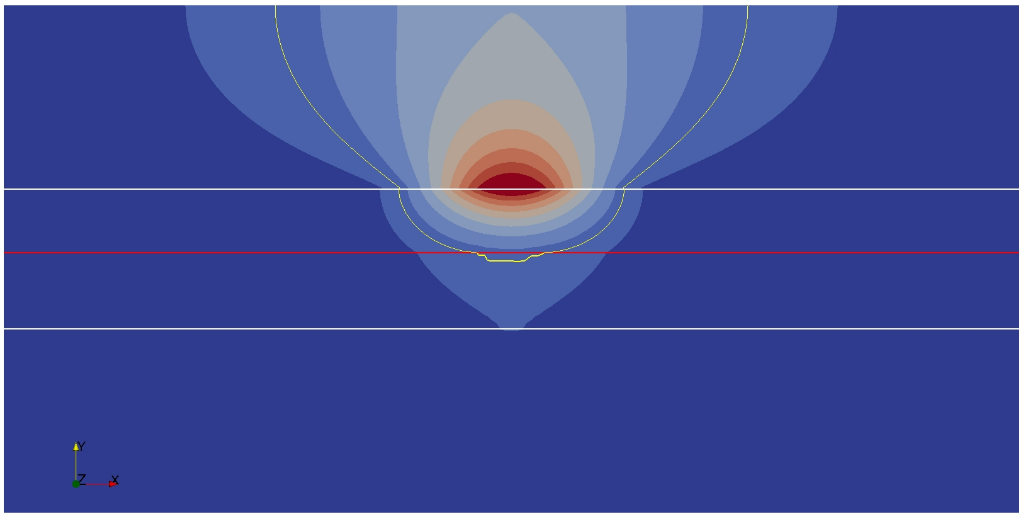
Figure 5.19 Laser is applied to the steel sheet and heat is conducted. The level set corresponding to the melting temperature of aluminum is shown in yellow. As it is well below the melting temperature of steel, it does not have an impact on the steel sheet.

Setup: Let $\Omega = (-8\,\text{mm}, 8\,\text{mm})^2$ with $\partial\Omega = \Gamma_\text{R}$ be a fixed domain and $t \in [0\,\text{s}, 0.1\,\text{s}]$. We describe the boundary $\Gamma_1(t)$ between workpiece and surrounding atmosphere by the zero level set of the function

$$\varphi_1(\boldsymbol{x}, t) := r_0 - \left\| \boldsymbol{x} - \boldsymbol{c}_\text{geo} - \vec{n}_\text{geo} \cdot (\boldsymbol{x} \cdot \vec{n}_\text{geo}) \right\|_2 \tag{5.71}$$

with $r_0 = 1.1$ mm, $\boldsymbol{c}_\text{geo} = (0, 0)^T$, and $\vec{n}_\text{geo} = (1, 0)^T$. The steel and aluminum sheet are separated by the zero level set $\Gamma_2(t)$ of the function

$$\varphi_2(\boldsymbol{x}, t) := y_0 - y \tag{5.72}$$



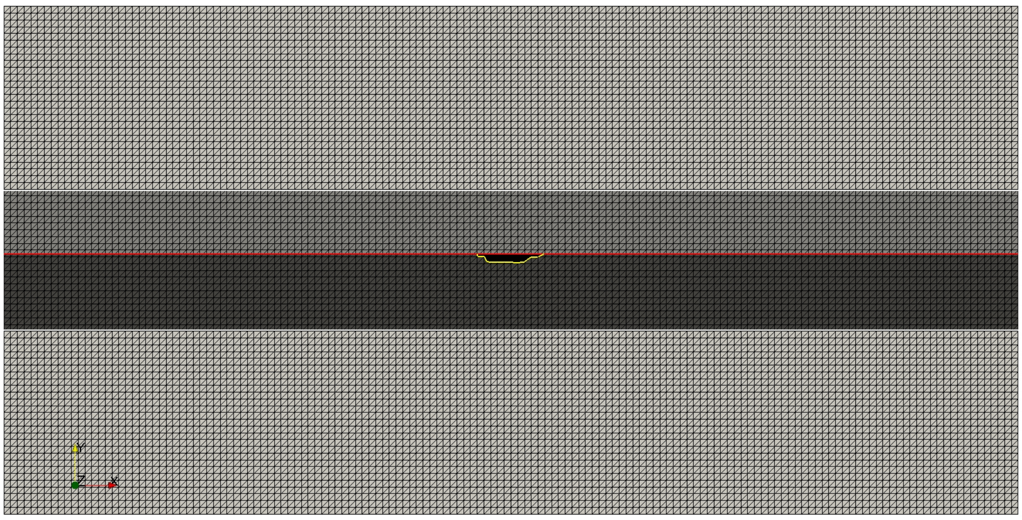


Figure 5.20 Due to the increasing amount of energy introduced by the laser, the aluminum heats up and a melt pool nucleates.

with $y_0 = 0.1$ mm. The boundary $\Gamma_{\mathrm{L}}(t)$ on which the laser heat source is active is given by

$$\Gamma_{\mathrm{L}}(t) := \{\boldsymbol{x} \in \Gamma_1(t) \,:\, \|\boldsymbol{x} - \boldsymbol{c}_{\mathrm{geo}}\|_2 \leq r_{\mathrm{L}}\}, \tag{5.73}$$

with $r_{\mathrm{L}} = 1.5$ mm.

For the numerical simulation, we used an implicit Euler scheme for time discretization and the polynomial degree $m = 1$ for the hierarchically eXtended approximation space V_h that is based on the (conventional) function space $V^m_{\mathrm{cg},h}$. The computational mesh consists of $N_{\mathrm{el}} = 2\times150\times75$ elements and the time step size is $\Delta \in [10^{-6}, 5\cdot10^{-5}]$, depending of the CFL conditions. For the velocity computation scheme, we used the DSCE method with $\delta = 0.4$. The narrow band parameters for φ_3 are $\beta_{\mathrm{I}} = 2$, $\beta_{\mathrm{O}} = 4$, and $\gamma = 6$.

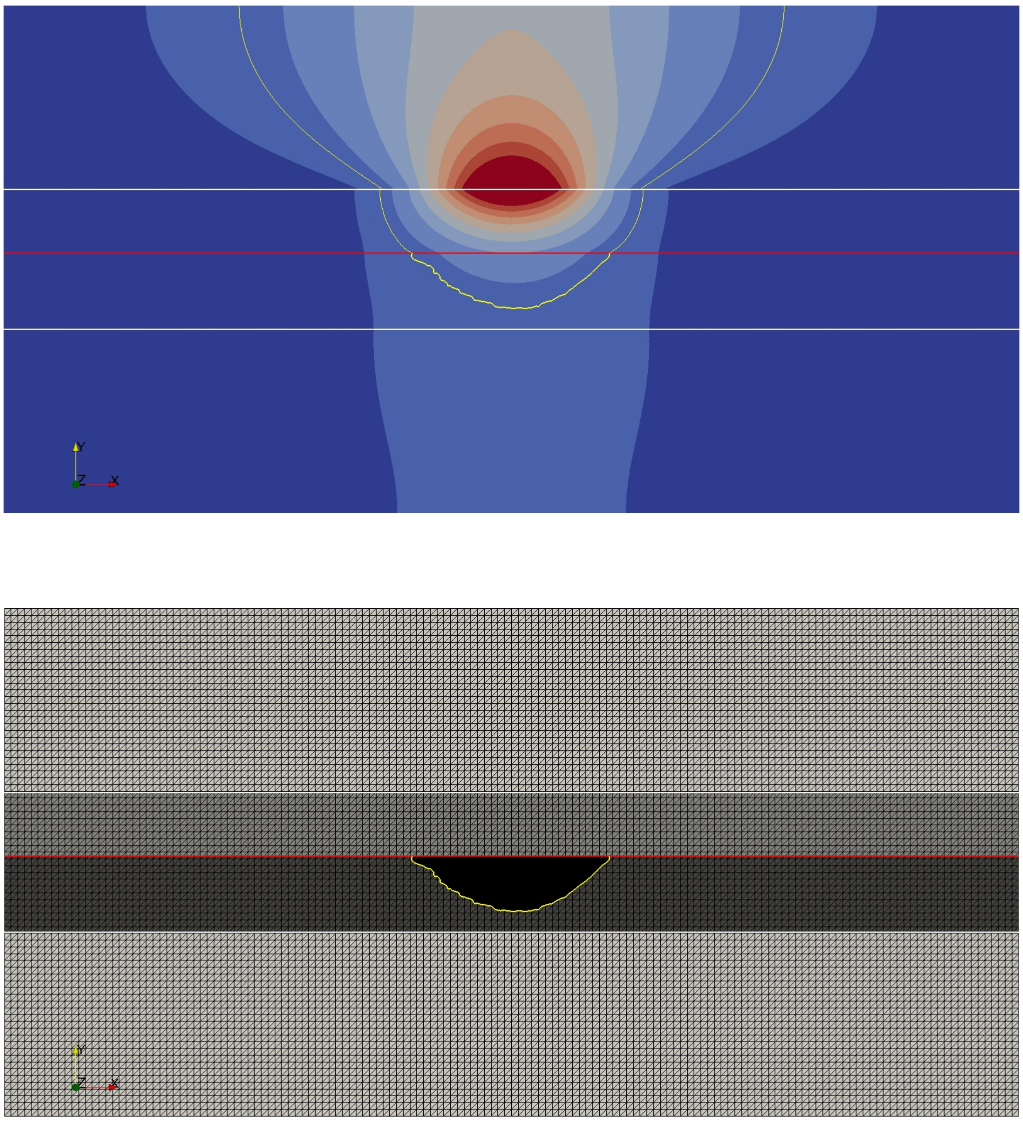
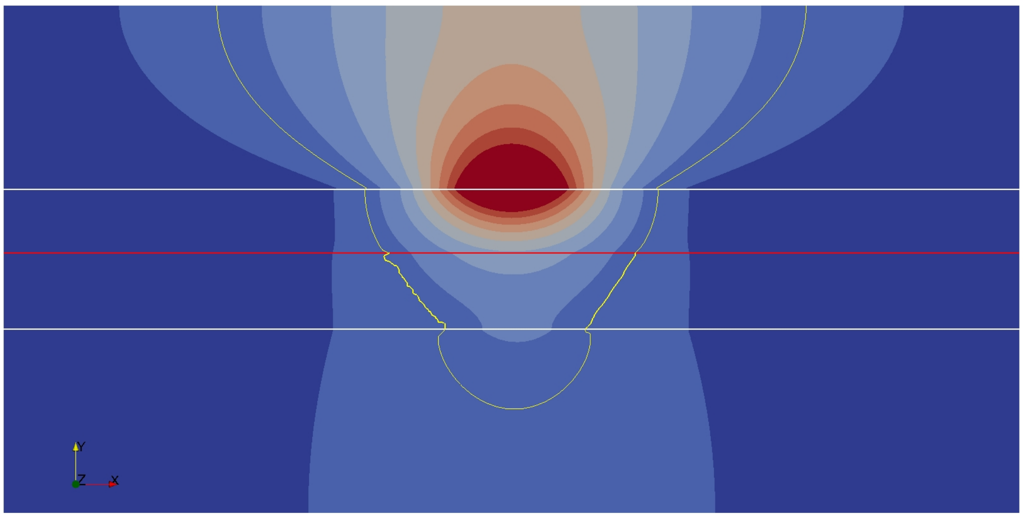


Figure 5.21 The melt pool evolves due to the applied laser energy.

Results: Visualizations of the numerical results are given in Figures 5.18 to 5.22. Therein we show the temperature distribution and the various subdomains for different points in time. Initially, we have overlapping solid steel and aluminum sheets which are surrounded by a gas atmosphere, Figure 5.18. The laser is applied on $\Gamma_{\mathrm{L}}(t) \subset \Gamma_1(t)$ and heats the steel which conducts the energy to the aluminum, Figure 5.19. After some time, when the heat is transported through the steel sheet and the temperature in the aluminum exceeds the melting temperature $u_{3,4}$, the aluminum changes it phase and the melt pool nucleates, see Figure 5.20. Figure 5.21 shows the further evolution of the melt pool which finally penetrates the aluminum sheet completely, see Figure 5.22. Now, we would have to consider the free capillary surface of the melt pool, which is not taken into account here as we consider a reduced model and neglect the fluid dynamics.



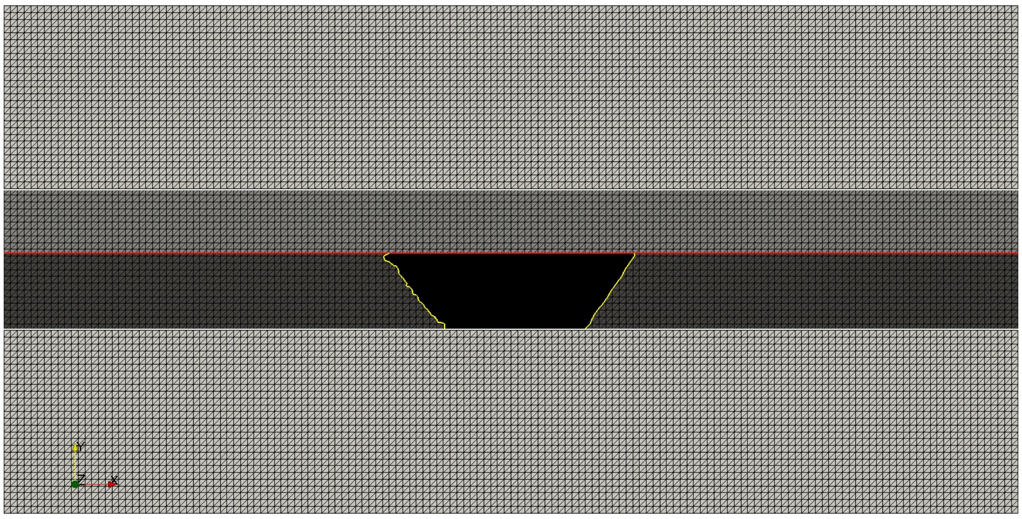


Figure 5.22 The amount of molten aluminum increases further, now the aluminum sheet is penetrated and the melt has a free surface.

Remark 5.6 (Continuity of the temperature at the triple point)*.* In fact, the conditions at the triple points, which are continuity of the temperature across all involved subdomains and a jump in the temperature gradient across Γ_3, cannot both be satisfied. This is because we have not enough degrees of freedom in non-melting subdomain. In our method, we put more weight on the continuity of the temperature (by choosing larger stabilization parameters for this condition) and, hence, introduce a hierarchical order of conditions in some way.

Remark 5.7 (Nucleation and dissolution of subdomains)*.* In practice, the detection nucleation and dissolution of subdomains is subject to the discrete representation of the subdomain and, hence, the mesh resolution and the chosen approximation space of the corresponding level set function.

Table 5.6 Material properties of air, steel 1.0330 and aluminum 3.2315 (solid/liquid)[3–5].

symbol	**air**	**1.0330**	**3.2315** (sol.)	**3.2315** (liq.)	**description**
u_0	293	293	293	-	initial temperature [K]
u_a	293	293	293	293	ambient temperature [K]
u_m	-	1700	933	-	melting temperature [K]
ρ	1.2041	7860	2700	2700	density $\left[\frac{\mathrm{kg}}{\mathrm{m}^3}\right]$
c	830	460	900	900	specific heat capacity $\left[\frac{\mathrm{J}}{\mathrm{kgK}}\right]$
λ	0.0262	60	160	110	thermal conductivity $\left[\frac{\mathrm{J}}{\mathrm{mK}}\right]$
L	-	-	386000	386000	latent heat $\left[\frac{\mathrm{J}}{\mathrm{kg}}\right]$

Table 5.7 Process parameters.

symbol	**value**	**description**
P_L	2100	laser power [W]
λ_L	$1.03 \cdot 10^{-6}$	laser wave length [m]
r_L	$1.5 \cdot 10^{-3}$	laser beam radius [m]
α	10	heat transfer coefficient $\left[\frac{\mathrm{W}}{\mathrm{m}^2\mathrm{K}}\right]$
ϵ	0.5	radiation coefficient

5.5 A laser-based thermal upsetting process

The next application that will be considered is the laser-based thermal upsetting process, see Section 1.1.1, which is the first stage of an alternative cold forming process for the generation of functional parts in micro scale that has been developed within the Collaborative Research Center (CRC) 747 “Micro Cold Forming”.

Just as the process discussed in Section 5.4, the thermal upsetting process can be modeled mathematically by considering the heat equation on all subdomains and coupling the two-phase Stefan problem with the Navier-Stokes equations including a free capillary surface. At outer and inner domain boundaries, conditions for laser heating, thermal conduction, and radiation described by the Stefan-Boltzmann law have to be included, see, e.g., [9, 13]. While we have extensively analyzed the process in various publications, see i.a. [9–11, 13, 108], several finite element approaches have been specifically developed for the consideration of different process designs. This is because all previous simulation approaches base on the conventional finite element method, where physical and computational domain are the same. Hence, all geometrical changes have to be considered by moving the computational mesh, which also prevented the consideration of the surrounding atmosphere. In contrast to this, the presented hierarchical eXtended finite element method allows for decoupling the geometry from the mesh and, thereby,

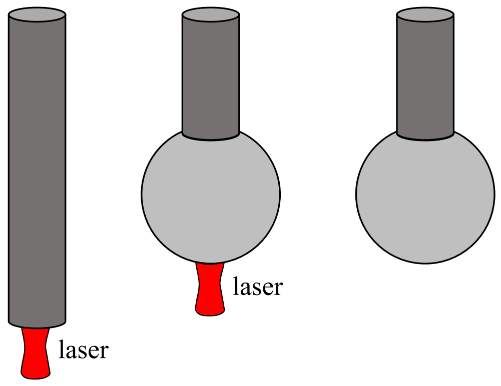


Figure 5.23 Thermal upsetting process.

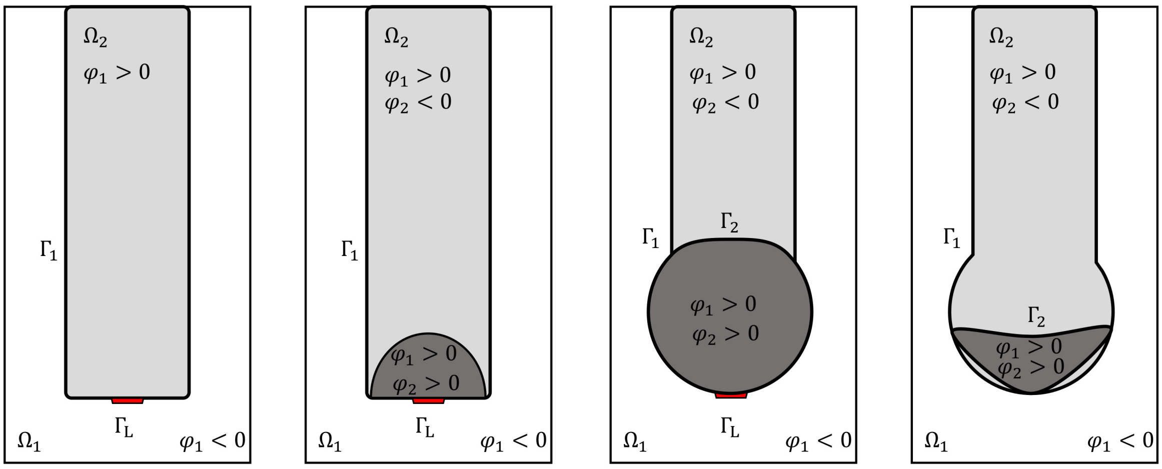
significantly enhances our modeling and simulation possibilities. Moreover, it provides a very flexible framework so that we can also apply the process model to arbitrary geometries.

5.5.1 Modeling the process with the hierarchical level set method

In the following, we show that the implemented framework can be used to simulate (a simplified version of) the laser-based thermal upsetting process. While we do not model and simulate the full process including all interdependencies, we still want to point out the advantages of using the hierarchical finite element method for the modeling and simulation of all process stages. Therefore, we define a reduced setting where we only consider the thermal aspects on the workpiece, which is chosen here to be a cylindrical rod, and the surrounding gas atmosphere but neglect the fluid dynamics. Compared to the process model for the generation of hybrid joints presented in Section 5.4.1, we now extend the setting and additionally model the evolution of the workpiece geometry. Since we do not model the fluid dynamics, the geometrical evolution of the molten domain in time is approximated by an analytical approach which, in a way, corresponds to considering the application in a zero-gravity environment.

Modeling approach

Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be a fixed domain and let $[t_0, t_f]$ denote the considered time interval. We model the thermal upsetting process applied to a rod for the given time interval $[t_0, t_f]$ using two hierarchically ordered level set functions $\varphi_1, \varphi_2 \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$, see Figure 5.24. Given an initial workpiece geometry $\Gamma_1(t_0)$, we first define $\varphi_1(\boldsymbol{x}, t_0)$ as signed distance function to the given workpiece boundary such that its zero level set $\Gamma_1(t_0)$ separates the surrounding atmosphere ($\varphi_1 < 0$) from the workpiece ($\varphi_1 > 0$). As before, we separate the workpiece boundary in a small region $\Gamma_\mathrm{L} \subset \Gamma_1$, where a heat source is applied, see Figure 5.24(a), and assume continuity of the temperature $u \in \mathcal{C}^0(\bar{\Omega} \times [t_0, t_f])$ on $\Gamma_1 \setminus \Gamma_\mathrm{L}$. Assuming that the temperature u exceeds the melting temperature $u_{2,3}$ of the workpiece at some point in time, we describe the solid-liquid interface by the zero level set Γ_2 of a hierarchically subordinated



(a) Situation at time t_0. (b) Nucleation of molten material. (c) Melt pool evolves and forms spherically. (d) Melt solidifies (after laser is switched off).

Figure 5.24 Different process stages of the laser-based thermal upsetting process.

function $\varphi_2 \in C^1\left(\Omega \times \left((t_0, t_f) \setminus \mathcal{T}\right)\right) \cap C^0\left(\bar{\Omega} \times \left([t_0, t_f] \setminus \mathcal{T}\right)\right)$ which separates the workpiece into a solid part ($\varphi_2 < 0$) and a molten part ($\varphi_2 > 0$), see Figure 5.24(b). The melt pool evolves during the irradiation time, cf. Figure 5.24, until the laser is switched off. Afterwards, the solidification begins, see Figure 5.24(d). Similarly to the description in Section 5.4.1, Γ_2, whose evolution is driven by the Stefan condition, is only non-empty if $u(\boldsymbol{x}, t) > u_{2,3}$ for some $\boldsymbol{x} \in \Omega \setminus \Omega_1(t)$ and $t \in [t_0, t_f]$. Consequently, we again have to detect the points in time $\mathcal{T}$ when the topology changes due to nucleation or dissolution of melt. Analog to the description in Section 5.4.1, we do this by introducing the function $\check{\varphi}_2(\boldsymbol{x}, t) := u(\boldsymbol{x}, t) - u_{2,3}$, the index set $\mathcal{L}$, and the dissolved domain $\Xi(t)$ to define $\mathcal{T}$ and define, for $t \in \mathcal{T}$, the zero level set $\Gamma_2(t) := \check{\Gamma}_2(t)$ and $\varphi_2(\boldsymbol{x}, t)$ as corresponding signed distance function to $\Gamma_2(t)$ for all $\boldsymbol{x} \in \Omega \setminus \Omega_1(t)$ that is in a sufficiently smooth way extended to Ω_1.

Coupled model

In brief, the reduced model for the thermal upsetting process for the time interval $[t_0, t_f])$ reads: Given the initial workpiece geometry $\Gamma_1(t_0)$ and the corresponding signed distance function $\varphi_1(\boldsymbol{x}, t_0)$, find the workpiece geometry $\Gamma_1(t)$ given by the zero level set of $\varphi_1 \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$, the solid-liquid interface $\Gamma_2(t)$ given by the zero level set of $\varphi_2 \in C^1\left(\Omega \times \left((t_0, t_f) \setminus \mathcal{T}\right)\right) \cap C^0\left(\bar{\Omega} \times \left([t_0, t_f] \setminus \mathcal{T}\right)\right)$, and the temperature distribution $u \in \mathcal{C}^0(\bar{\Omega} \times [t_0, t_f])$ which is sufficiently smooth, i.e., $u(\cdot, t)_{|\Omega_i} \in \mathcal{C}^2(\Omega_i(t))$ and $\partial_t u(\cdot, t) \in C^0(\Omega_1(t) \cup$

$\Omega_2(t) \cup \Omega_3(t))$ for $t \in [t_0, t_f]$ such that

$$\begin{cases} \rho c \dfrac{\partial u}{\partial t} - \nabla \cdot (\lambda \nabla u) = 0 & \text{in } \bigcup_{i=1}^{3} \Omega_i(t),\ t \in (t_0, t_f), \quad (5.74a) \\ u = g_\mathrm{D} & \text{on } \Gamma_\mathrm{D} \times (t_0, t_f], \quad (5.74b) \\ -\lambda \nabla u \cdot \vec{n} = g_\mathrm{N} & \text{on } \Gamma_\mathrm{N} \times (t_0, t_f], \quad (5.74c) \\ -\lambda \nabla u \cdot \vec{n} = g_\mathrm{R}(u) & \text{on } \Gamma_\mathrm{R} \times (t_0, t_f], \quad (5.74d) \\ [\![u]\!] = 0 & \text{on } \Gamma_1(t), \quad (5.74e) \\ [\![\lambda \nabla u]\!] \cdot \vec{n}_1 = g_1 & \text{on } \Gamma_1(t), \quad (5.74f) \\ u(\cdot, t) = u_{1,2} & \text{on } \Gamma_2(t), \quad (5.74g) \\ u(\cdot, t_0) = u_0 & \text{in } \bigcup_{i=1}^{3} \Omega_i(t_0), \quad (5.74h) \end{cases}$$

$$\begin{cases} (\lambda_2 \nabla u_2 - \lambda_3 \nabla u_3) \cdot \vec{n}_2 = \rho L \vec{V}_{2,3} \cdot \vec{n}_2 & \text{on } \Gamma_2(t),\ t \in [t_0, t_f] \setminus \mathcal{T} \quad (5.75a) \\ \dfrac{\partial \varphi_2}{\partial t} + \vec{V}_{2,3} \cdot \nabla \varphi_2 = 0 & \text{in } \Omega \times [t_0, t_f] \setminus \mathcal{T}, \quad (5.75b) \end{cases}$$

$$\begin{cases} \check{\varphi}_2(\cdot, t) := u(\cdot, t) - u_{2,3} & \text{in } \Omega \times [t_0, t_f], \quad (5.76a) \\ \Gamma_2(t) := \{\boldsymbol{x} \in \Omega \,:\, \check{\varphi}_2(\cdot, t) = 0 \wedge \varphi_1(\boldsymbol{x}, t) > 0\} \setminus \Xi^-(t) & t \in \mathcal{T}, \quad (5.76b) \\ \varphi_2(\cdot, t) := \begin{cases} -\min\limits_{\boldsymbol{y} \in \Gamma_2(t)} \|\boldsymbol{x} - \boldsymbol{y}\|_2, & \text{if } \boldsymbol{x} \in \Omega_2 \\ \min\limits_{\boldsymbol{y} \in \Gamma_2(t)} \|\boldsymbol{x} - \boldsymbol{y}\|_2, & \text{if } \boldsymbol{x} \in \Omega_3\ , \\ \text{smooth} & \text{else} \end{cases} & \text{in } \Omega(t),\ t \in \mathcal{T}, \quad (5.76c) \end{cases}$$

$$s_{\varphi_1}(\varphi_1, \varphi_2, t) = 0 \qquad \text{in } \Omega \times [t_0, t_f], \qquad (5.77)$$

for given sufficiently smooth data. As before, we introduce a small subset $\Gamma_\mathrm{L}(t) \subset \Gamma_1(t)$ and define the function $g_1(\boldsymbol{x}, t)$ as in equation (5.64) so that it models the laser heat source applied on $\Gamma_\mathrm{L}(t)$ and enforces continuity of the heat fluxes across $\Gamma_1(t) \setminus \Gamma_\mathrm{L}(t)$. The condition g_D is defined as ambient temperature, g_N is usually chosen to be zero which corresponds to isolation, and g_R again describes cooling, cf. equation (5.29). The outer unit normals $\vec{n}_i(t, \boldsymbol{x})$ to $\Gamma_i(t)$ are pointing from Ω_i into Ω_l, $1 \leq i < l \leq 3$. Moreover, s_{φ_1} is an analytical model providing us with the workpiece geometry and we have a hierarchical order among the level set functions φ_1 and φ_2.

5.5.2 Discretization and computational approach

Because of the obvious analogy, the problem given by the equations (5.74) to (5.77) is considered in the same way as the problem modeled and discussed in Section 5.4.2. Therefore, we omit a detailed presentation and only address the issue of representing the evolving workpiece geometry described by $\Gamma_1(t)$ and its impact on the computational approach.

As we do not model the fluid dynamics and capillary surface of the melt, we consider the evolution of the geometry due to changes of the melt pool volume by an analytical model. This model bases on the observation that the melt forms spherically during the melting process and preserves its shape during the solidification process. Therefore, we only have to model the evolution of φ_1 during the melting process and preserve the computed geometry as soon as the solidification process starts. For simplicity, we restrict our presentation to $d = 2$. The full analytical approach for the computation of $\Gamma_1(t_{n+1})$ is visualized in Figure 5.25 and works as follows:

Given the workpiece geometry $\Gamma_{1,h}(t_n)$ and positions of the interface $\Gamma_{2,h}(t_n)$ and $\tilde{\Gamma}_{2,h}(t_{n+1})$ for $t_0 \le t_n < t_{n+1} \le t_f$ as well as the corresponding level set functions $\varphi_{1,h}(t_n)$, $\varphi_{2,h}(t_n)$, and $\varphi_{2,h}(t_{n+1})$, we firstly compute the difference in the volume of the solid part

$$\begin{aligned}\Delta V_2(t_{n+1}) &= V_2(t_n) - V_2(t_{n+1}) \\ &= \int_{\{\boldsymbol{x}\in\Omega\,:\,\varphi_{1,h}(\boldsymbol{x},t_n)>0\wedge\varphi_{2,h}(\boldsymbol{x},t_n)<0\}} 1\,\mathrm{d}x - \int_{\{\boldsymbol{x}\in\Omega\,:\,\varphi_{1,h}(\boldsymbol{x},t_n)>0\wedge\varphi_{2,h}(\boldsymbol{x},t_{n+1})<0\}} 1\,\mathrm{d}x,\end{aligned} \tag{5.78}$$

cf. Figure 5.25(a).

If $\Delta V_2(t_{n+1}) \le 0$, the amount of solid material has not changed or even increased due to solidification and we do not change the geometry due to our assumption that the spherical shape is preserved. Hence we define $\varphi_{1,h}(t_{n+1}) := \varphi_{1,h}(t_n)$ and $\Gamma_{1,h}(t_{n+1}) := \Gamma_{1,h}(t_n)$. However, if $\Delta V_2(t_{n+1}) > 0$, the amount of solid material has decreased and we have to compute a new geometry $\Gamma_{1,h}(t_{n+1})$ and a new corresponding signed distance function $\varphi_{1,h}(t_{n+1})$. The new geometry $\Gamma_{1,h}(t_{n+1})$ is constructed by performing the following steps:

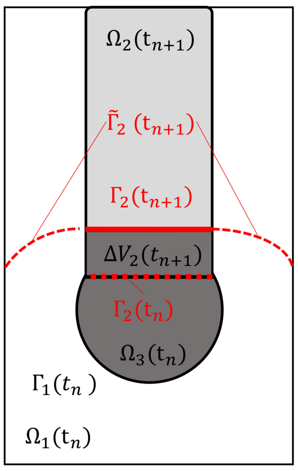
(Step 1) Compute the intersection points $p_l(t_{n+1})$, $l = 1, 2$, of $\tilde{\Gamma}_2(t_{n+1})$ with $\Gamma_{1,h}(t_n)$ and the new volume of the melt pool at t_{n+1} by

$$V_3(t_{n+1}) = V_3(t_n) + \Delta V_2(t_{n+1}), \tag{5.79}$$

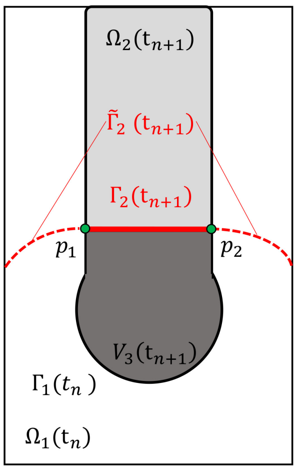
cf. Figure 5.25(b), with

$$V_3(t_n) = \int_{\{\boldsymbol{x}\in\Omega\,:\,\varphi_{1,h}(\boldsymbol{x},t_n)>0\wedge\varphi_{2,h}(\boldsymbol{x},t_n)>0\}} 1\,\mathrm{d}x. \tag{5.80}$$

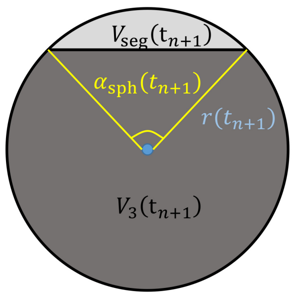
(Step 2) Since we assume that the melt pool forms spherically, it can be described by large spherical segment that is actually a sphere which is missing a small spherical segment, see Figure 5.25(c). As a result, we get the radius $r(t_{n+1})$ of the new (large) spherical



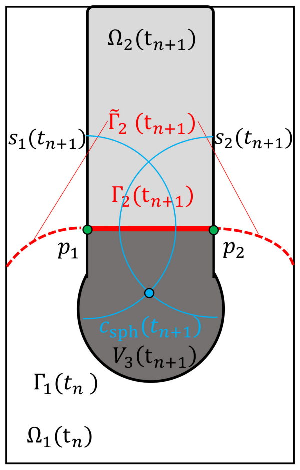
(a) Situation at time t_n.



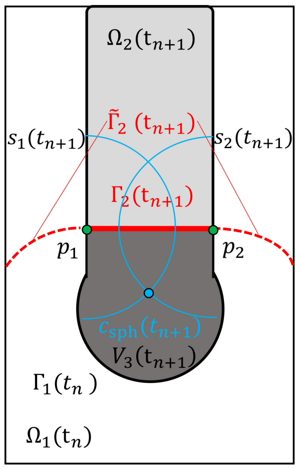
(b) Situation after moving $\Gamma_{2,h}(t_n)$ with $V_{1,2}^{n+1}$.



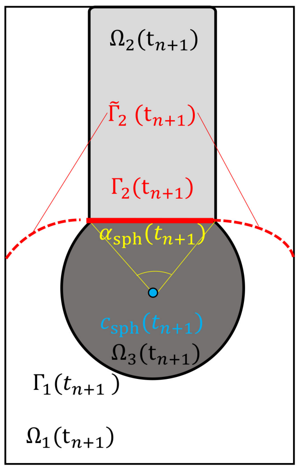
(c) Computation of new radius $r(t_{n+1})$.



(d) Construction of spheres s_1, s_2 with radius $r(t_{n+1})$.



(e) Construction of the center of gravity $c_{\text{sph}}(t_{n+1})$.



(f) Resulting geometry $\Gamma_1(t_{n+1})$.

Figure 5.25 Analytical approach for approximating the evolution of the workpiece geometry Γ_1.

segment by computing the root of

$$\begin{aligned} 0 &= V_3(t_{n+1}) - \left(V_{\text{sph}}(t_{n+1}) - V_{\text{seg}}(t_{n+1})\right) \\ &= V_3(t_{n+1}) - \left(r(t_{n+1})^2\pi - \frac{1}{2}r(t_{n+1})^2\left(\alpha_{\text{seg}}(t_{n+1})\right) - \sin\left(\alpha_{\text{seg}}(t_{n+1})\right)\right) \end{aligned} \tag{5.81}$$

with

$$\alpha_{\text{seg}}(t_{n+1}) = 2\arcsin\left(\frac{\text{dist}(p_1(t_{n+1}), p_2(t_{n+1}))}{2r(t_{n+1})}\right). \tag{5.82}$$

(Step 3) After that, we define spheres $s_1(t_{n+1})$ and $s_2(t_{n+1})$ with radius $r(t_{n+1})$ using the intersection points $p_1(t_{n+1})$ and $p_2(t_{n+1})$ as center of gravities by

$$s_l(t_{n+1}) := \{\boldsymbol{x} \in \Omega\ :\ \|\boldsymbol{x} - p_l(t_{n+1})\|_2 = r(t_{n+1})\}\,, \quad l = 1, 2, \tag{5.83}$$

cf. Figure 5.25(d), and compute the intersection points $\tilde{p}_1$ and $\tilde{p}_2$ of the spheres. As only one of these points is in $\Omega_3(t_n)$, the new center of gravity $c_{\text{sph}}(t_{n+1})$ of the melt is given by

$$c_{\text{sph}}(t_{n+1}) := \begin{cases} \tilde{p}_1, & \text{if } \varphi_{2,h}(\tilde{p}_1, t_n) > 0 \\ \tilde{p}_2, & \text{if } \varphi_{2,h}(\tilde{p}_2, t_n) > 0 \end{cases}, \tag{5.84}$$

cf. Figure 5.25(e).

(Step 4) Then, the new workpiece geometry $\Gamma_{1,h}(t_{n+1})$ is given by

$$\begin{aligned} \Gamma_{1,h}(t_{n+1}) &:= \{\boldsymbol{x} \in \Gamma_{1,h}(t_n) \,:\, \varphi_{2,h}(t_{n+1}) < 0\} \\ &\quad \cup \{\boldsymbol{x} \in \Omega \,:\, \varphi_{2,h}(t_{n+1}) > 0 \wedge \|\boldsymbol{x} - c_{\text{sph}}(t_{n+1})\|_2 = r(t_{n+1})\}, \end{aligned} \tag{5.85}$$

cf. Figure 5.25(f), and we define $\varphi_{1,h}(t_{n+1})$ as corresponding signed distance function that is positive within the material and negative in the surrounding atmosphere.

Please note that we only use the presented approach if $V_3(t_{n+1}) > V_{\text{crit}}(d_0)$, which means that the volume of the melt $V_3(t_{n+1})$ exceeds a lower boundary $V_{\text{crit}}(d_0)$ that depends on the diameter d_0 of the rod. For example, we can choose $V_{\text{crit}}(d_0) = \frac{2}{3}V_{\text{sph}}$. This analytical approach is presented as pseudo code in Algorithm 7 and the complete computational approach is presented in Algorithm 8.

Algorithm 7 Computational approach for updating the workpiece geometry.

Input: Ω, $\mathcal{S}_h$, $\varphi_{1,h}(t_n)$, $\varphi_{2,h}(t_n)$, $\varphi_{2,h}(t_{n+1})$, $\Gamma_{1,h}(t_n)$, $\Gamma_{2,h}(t_n)$, $\tilde{\Gamma}_{2,h}(t_{n+1})$, V_{crit}.
Output: $\varphi_{1,h}(t_{n+1})$, $\Gamma_{1,h}(t_{n+1})$.

```
 1: procedure COMPUTATIONAL APPROACH FOR UPDATING THE WORKPIECE GEOMETRY.
 2:     Compute $\Delta V_2^{n+1}$ by (5.78).
 3:     if $\Delta V_2^{n+1} \leq 0$. then
 4:         $\Gamma_{1,h}^{n+1} := \Gamma_{1,h}^{n}$.
 5:         $\varphi_{1,h}^{n+1} := \varphi_{1,h}^{n}$.
 6:     else
 7:         Compute intersection points $p_l^{n+1}$ of $\tilde{\Gamma}_{2,h}^{n+1}$ and $\Gamma_{1,h}^{n}$, cf. (Step 1).
 8:         Compute $V_3^{n+1}$, cf. (5.79).
 9:         if $V_3^{n+1} \leq V_{\text{crit}}$. then
10:             $\Gamma_{1,h}^{n+1} := \Gamma_{1,h}^{n}$.
11:             $\varphi_{1,h}^{n+1} := \varphi_{1,h}^{n}$.
12:         else
13:             Compute $r^{n+1}$ by solving (5.81).
14:             Compute center of sphere $c_{\text{sph}}$ as in (Step 3).
15:             Update $\Gamma_{1,h}^{n+1}$ and $\varphi_{1,h}^{n+1}$ as described in (Step 4).
16:         end if
17:     end if
18: end procedure
```

Algorithm 8 Computational approach for the laser-based thermal upsetting process.

Input: Ω, Γ_{D}, Γ_{N}, Γ_{R}, $\mathcal{S}_h$, $\varphi_1(t_0)$, $u(t_0)$, $u_{1,2}$, f, κ, λ, ρ, c, L, g_{D}, g_{N}, g_{R}, t_0, t_f, $\Delta t_{\max}$, $\Delta t_{\min}$, β_O, β_I, γ.
Output: u_h^n, $\Gamma_{1,h}^n$, $\Gamma_{2,h}^n$, $\vec{V}_{1,2,h}^n$ for $n > 0$.

```
1: procedure SIMULATION OF THE LASER-BASED THERMAL UPSETTING PROCESS.
2:     Initialization: $t_n = t_0$, $u_h^n = u(t_0)$, $\varphi_{1,h}^n = \varphi_1(t_0)$, $\Gamma_{1,h}^n = \Gamma_1(t_0)$.
3:     while $t_n < t_f$ do
4:         Define $\check{\varphi}_{2,h}^n(\boldsymbol{x}) := u_h^n(\boldsymbol{x}) - u_{1,2}$ for $\boldsymbol{x} \in \Omega$.
5:         if $\Gamma_{2,h}^n := \left\{\boldsymbol{x} \in \Omega \,:\, \varphi_{1,h}^n(\boldsymbol{x}) \geq 0 \wedge \check{\varphi}_{2,h}^n(\boldsymbol{x}) = 0\right\} \neq \emptyset$ then
6:             Define $\varphi_{2,h}^n(\boldsymbol{x})$ as signed distance function to $\Gamma_{2,h}^n$ and extend it to $\Omega$.
7:             if the narrow band method is applied then
8:                 Initialize $\Omega_{2,\mathrm{INB}}$ and $\Omega_{2,\mathrm{ONB}}$ with $\varphi_{2,h}^n$.
9:             end if
10:            Compute $\vec{V}_{1,2}^n$ by evaluating (5.75a) with an approach presented in Section 4.4.2.
11:            Compute $\Delta t := \min\{\max\{\Delta t_{\mathrm{CFL}}, \Delta t_{\min}\}, \Delta t_{\max}, t_f - t_n\}$.
12:            $t_{n+1} := t_n + \Delta t$.
13:            Compute new level set function $\varphi_{2,h}^{n+1}$ by solving (5.75b).
14:            Compute the new discrete interface $\Gamma_{2,h}^{n+1}$ from $\varphi_{2,h}^{n+1}$.
15:            Compute the new discrete interface $\Gamma_{1,h}^{n+1}$ with Algorithm 7.
16:        else
17:            Compute $\Delta t := \min\{\Delta t_{\max}, t_f - t_n\}$.
18:            $t_{n+1} := t_n + \Delta t$.
19:        end if
20:        Define $V_{h,g_{\mathrm{D}}}^1(t_{n+1})$.
21:        Compute $u_h^{n+1}$ by solving (5.74).
22:    end while
23: end procedure
```

5.5.3 Results

We now present simulation results for the thermal upsetting process for a rod of diameter $d_0 = 0.2$ mm consisting of steel 1.4301 using a coaxial process design, which means that the laser heat source is applied to the head surface of the rod, in a shielding gas atmosphere. The

Table 5.8 Material properties of Argon and steel 1.4301 (solid/liquid)[3–5].

symbol	**Argon**	**1.4301 (sol.)**	**1.4301** (liq.)	**description**
u_0	293	293	-	initial temperature [K]
u_a	293	293	293	ambient temperature [K]
u_m	-	1673	-	melting temperature [K]
ρ	1.784	7900	7900	density $\left[\frac{\mathrm{kg}}{\mathrm{m}^3}\right]$
c	520	830	830	specific heat capacity $\left[\frac{\mathrm{J}}{\mathrm{kgK}}\right]$
λ	0.0177	15	35	thermal conductivity $\left[\frac{\mathrm{J}}{\mathrm{mK}}\right]$
L	-	276000	276000	latent heat $\left[\frac{\mathrm{J}}{\mathrm{kg}}\right]$

Table 5.9 Process parameters.

symbol	**value**	**description**
t_{L}	0.07	irradiation duration [s]
P_{L}	80	laser power [W]
λ_{L}	$1.03 \cdot 10^{-6}$	laser wave length [m]
r_{L}	$0.025 \cdot 10^{-3}$	laser beam radius [m]
α	100	heat transfer coefficient $\left[\frac{\mathrm{W}}{\mathrm{m}^2\mathrm{K}}\right]$
ϵ	0.5	radiation coefficient

material parameters of the workpiece material and the shielding gas Argon are specified in Table 5.8 and the process parameters are given in Table 5.9.

Setup: Let $\Omega = (-1\,\mathrm{mm}, 1\,\mathrm{mm})^2$ be a fixed domain with $\partial\Omega = \Gamma_{\mathrm{R}}$ and $t \in [0\,\mathrm{s}, 0.2\,\mathrm{s}]$. We describe the initial workpiece boundary by the zero level set $\Gamma_1(t_0)$ of the level set function

$$\varphi_1(\boldsymbol{x}, t_0) := \min\left\{ r_0 - \left\| \boldsymbol{x} - \boldsymbol{c}_{\mathrm{geo}} - \vec{n}_{\mathrm{geo}} \cdot (\boldsymbol{x} \cdot \vec{n}_{\mathrm{geo}}) \right\|_2, y_0 - y \right\} \tag{5.86}$$

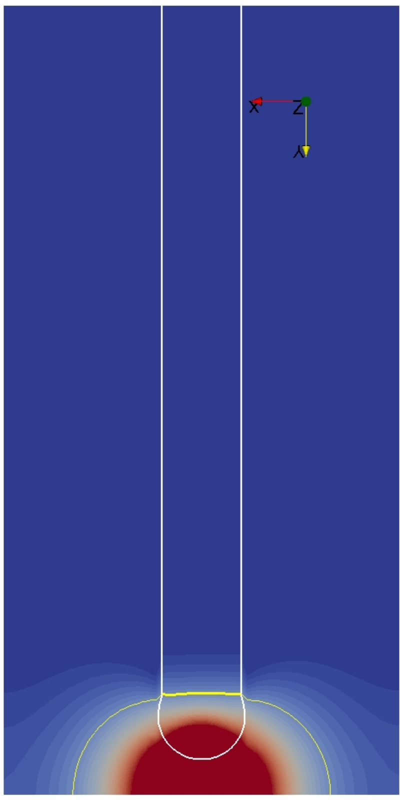
with $r_0 = 0.1$ mm $\boldsymbol{c}_{\mathrm{geo}} = (0,0)^T$, $\vec{n}_{\mathrm{geo}} = (0,1)^T$, and $y_0 = -0.1$ mm. The boundary $\Gamma_{\mathrm{L}}(t)$ on which the laser heat source is active is given by

$$\Gamma_{\mathrm{L}}(t) := \{\boldsymbol{x} \in \Gamma_1(t) \,:\, \|\boldsymbol{x} - \boldsymbol{c}_{\mathrm{geo}}\|_2 \leq r_{\mathrm{L}}\}, \tag{5.87}$$

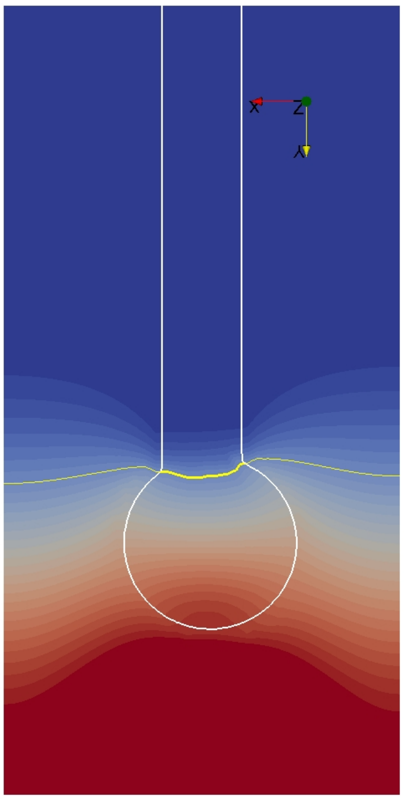
with $r_{\mathrm{L}} = 0.025$ mm.

For the numerical simulation, we used an implicit Euler scheme for time discretization and the polynomial degree $m = 1$ for the hierarchically enriched approximation space V_h that bases on the (conventional) function space $V_{\mathrm{cg},h}^m$. The computational mesh consists of $N_{\mathrm{el}} = 2 \times 39 \times 78$ elements and the time step size is $\Delta \in [10^{-6}, 10^{-4}]$, depending of the CFL conditions. For the velocity computation scheme, we used the DSCE method with $\delta = 0.3$. The narrow band parameters for φ_2 are $\beta_{\mathrm{I}} = 2$, $\beta_{\mathrm{O}} = 4$, and $\gamma = 6$.

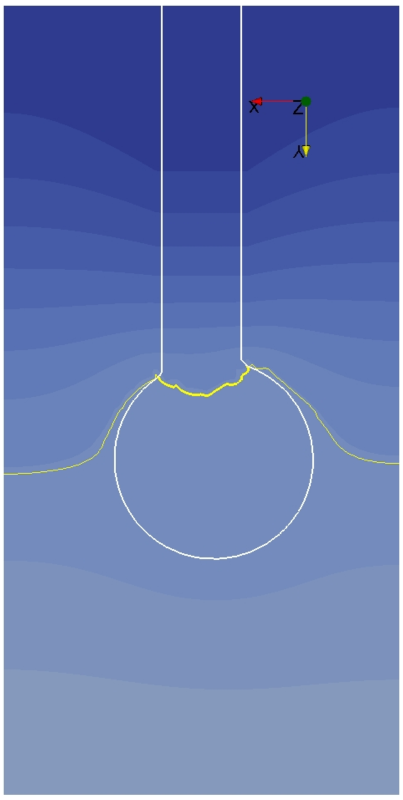
Results: Visualizations of the numerical results for the laser-based thermal upsetting process are given in Figures 5.26 and 5.27. Therein we show the temperature distribution, the subdomains, and the evolution of the geometry for different points in time. The laser is applied on $\Gamma_{\mathrm{L}}(t) \subset \Gamma_1(t)$ and heats the steel which melts after some time. When the volume of the melt $V_{\mathrm{melt}}(t)$ exceeds the value of $\frac{1}{2}V_{\mathrm{sphere}}(r_0)$, a new geometry is computed using the analytical approach, cf. Figures 5.26(a) and 5.27(a). Due to energy introduced by the laser heat source, the melt pool increases during the irradiation time, see Figures 5.26(b) and 5.27(b). The melting process continues for a short time period even after the laser is switched off because of the superheated melt until, finally, the solidification process starts, which is depicted



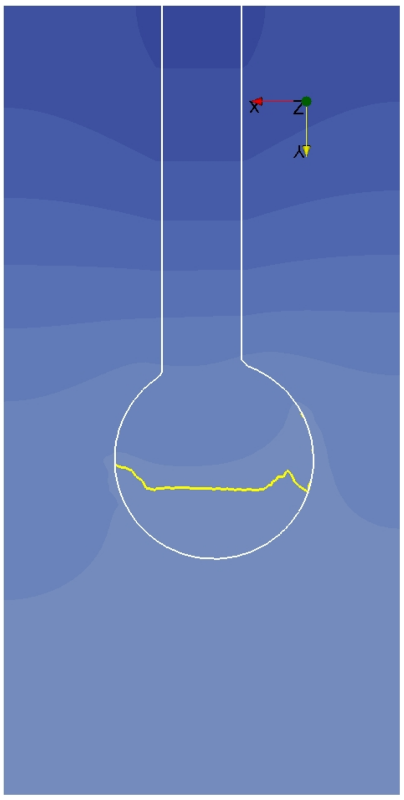
(a) Initial state (temperature).



(b) Melt pool increases (temperature).

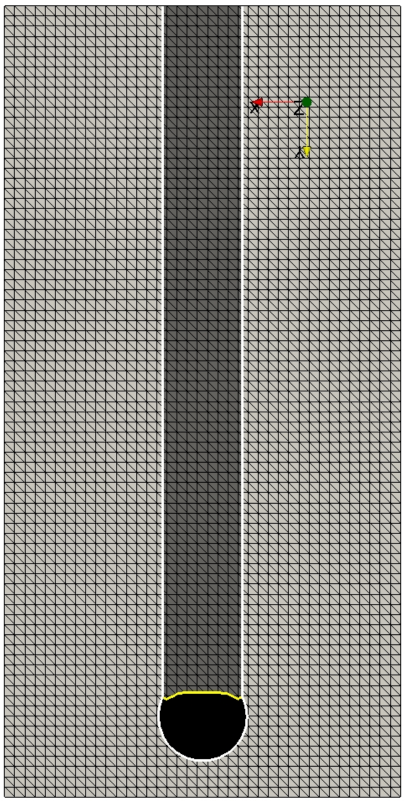


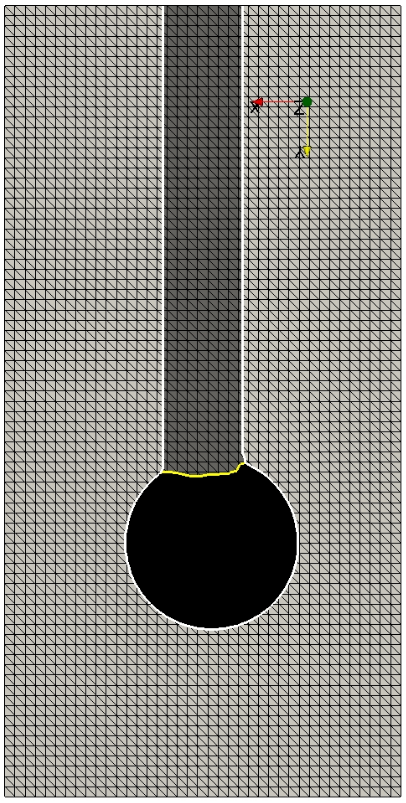
(c) Laser is switched off and solidification starts (temperature).

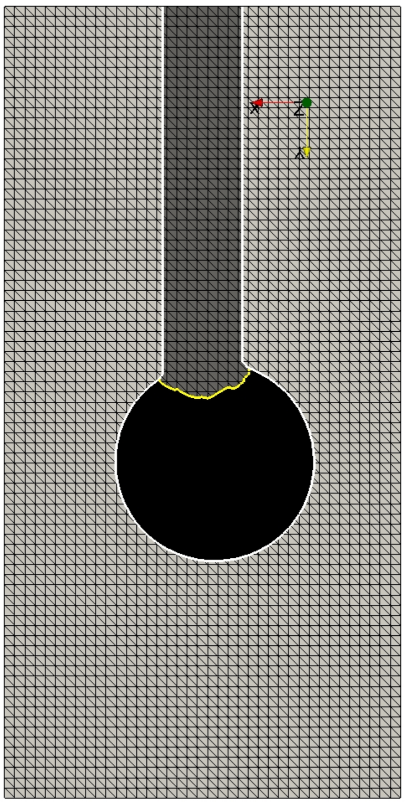


(d) Solidification process (temperature).

Figure 5.26 Temperature distribution for different process stages of the thermal upsetting process: Workpiece boundary Γ_1 is shown in white, solid-liquid interface Γ_1 is shown in yellow.


(a) Initial state (domains).


(b) Melt pool increases (domains).


(c) Laser is switched off and solidification starts (domains).

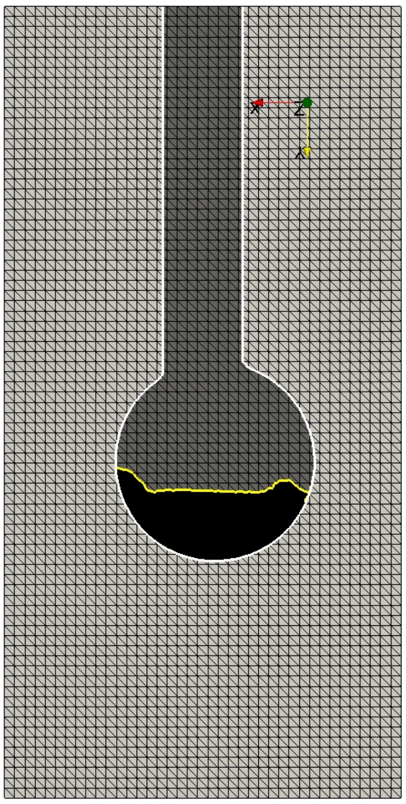

(d) Solidification process (domains).

Figure 5.27 Visualization of the domains for different process stages of the thermal upsetting process: Workpiece boundary Γ_1 is shown in white, solid-liquid interface Γ_1 is shown in yellow.

in Figures 5.26(c), 5.26(d), 5.27(c) and 5.27(d). The observable partial loss of the symmetry of the simulation results during the process time is a result of the the mesh orientation and discretization errors.

5.6 Keyhole-based laser welding

The last application motivating this thesis is the keyhole-based laser welding process which is briefly described in Section 1.1.3. In contrast to heat conduction welding, which is, for example, used to produce hybrid joints, the characteristic of the keyhole welding is that the material is processed with very high laser beam intensities so that the material vaporizes and a narrow but deep vapor channel, the keyhole, is formed. This process again can be modeled by considering the heat equation including the two-phase Stefan problem and the Navier-Stokes equations with a free capillary surface. In addition, the generation and evolution of the keyhole has to be modeled, which is a challenging task as keyhole welding processes are highly dynamical and sometimes unstable in practice [109]. Moreover, such models are very complex since aspects such as (re)vaporization effects and plasma creation have to be considered.

Remark 5.8 (Using the keyhole effect in other applications)*.* The keyhole effect can of course also be used within other applications. For example, it is of interest for the production of preforms generated by the thermal upsetting process, see Sections 1.1.1 and 5.5. The reason for this is that the keyhole effect allows for higher process speeds and, hence, can be used to reduce the process duration. This is of significant importance as the industrial production of preforms aims for high output rates such as 300 preforms per minute.

5.6.1 Modeling the process with the hierarchical level set method

For modeling the keyhole-based laser welding process with the hierarchical level set method, we neglect the issue of modeling a dynamic keyhole shape but assume that the keyhole geometry is constant and stable. Moreover, we further simplify the problem and assume that the (constant) keyhole shape can be computed a priori by an analytical model, see Appendix A, instead of

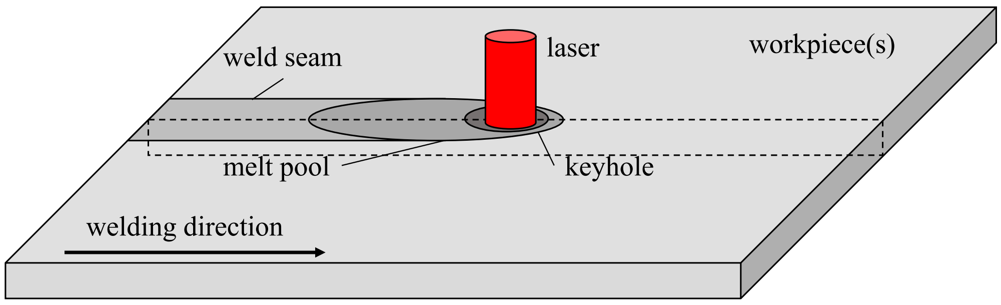


Figure 5.28 Keyhole-based laser welding.

considering the location and shape of the keyhole as part of the solution. By doing so, we can focus on the numerical applicability of `miXFEM` and model the process by the heat equation and the two-phase Stefan problem in level set formulation as in the previous applications. Therefore, the analytically derived keyhole geometry is considered as internal interface at which we impose evaporating temperature $u_{1,2}$ as (Dirichlet) interface condition. While the keyhole geometry is then translated with the laser motion velocity $\vec{V}_{\mathrm{L}}$ along the welding line, the evolution of the solid-liquid interface is still part of the solution.

Modeling approach

Let $\Omega \subset \mathbb{R}^3$ be a polygonally bounded domain (the workpiece). We model the process for $t \in [t_0, t_f]$ by introducing two level set function $\varphi, \varphi_2 \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$ whose zero level sets $\Gamma_1(t)$ (the keyhole wall) and $\Gamma_2(t)$ (the solid-liquid interface) separate Ω into disjoint regions $\Omega_1(t)$ (the vaporized material), $\Omega_2(t)$ (the molten material), and $\Omega_3(t)$ (the solid material). The keyhole model used in the remainder of this section is adapted from the approach presented in [110] and described in detail in Appendix A. The general idea of the model is to approximate the keyhole shape for a quasi-stationary situation by considering the energy balance at the keyhole wall, see Appendix A.4. For given material and laser parameters, the heat conduction is therefore approximated by a moving line source [111, 112] and the local inclination angle yielding the keyhole geometry is computed using a point-by-point scheme that compares absorbed power and conduction losses. Then, $\varphi_1(\cdot, t)$ is defined as corresponding signed distance function to the computed geometry, cf. Appendix A.6, and we assume that the temperature $u \in \mathcal{C}^0(\bar{\Omega} \times [t_0, t_f])$ at $\Gamma_1(t)$ corresponds to vaporization temperature $u_{1,2}$. In regards to the solid-liquid interface $\Gamma_2(t)$, we compute its initial position $\Gamma_2(t_0)$ as zero level set of $\varphi_2(\check{\boldsymbol{x}}, t_0) = u(\boldsymbol{x}, t_0) - u_{2,3}$, where $u(t_0)$ is a given initial temperature distribution and $u_{2,3}$ denotes the melting temperature of the material. The function $\varphi_2(\boldsymbol{x}, t_0)$ is then defined as corresponding signed distance function. The evolution of $\varphi_2(\cdot, t)$ (and hence the interface $\Gamma_2(t)$) in time is then modeled by the Stefan problem as in the previous applications. Since we have $\Gamma_1(t) \cap \Gamma_2(t) = \emptyset$ and $u = u_{1,2}$ at Γ_1 and $u = u_{2,3}$ at Γ_2, the level set functions φ_1 and φ_2 are hierarchically ordered in a natural way.

Remark 5.9 (Use of other keyhole models). Please note that since the only request for the thermal problem is that we have an a priori given keyhole geometry on which a Dirichlet condition for the temperature can be applied, any such keyhole model can be integrated into this approach[††].

[††]In fact, it is also possible to include an approach where the geometry is part of the solution. However, this would significantly increase the complexity of the model and, usually, would require adaptive mesh refinement, see [109, 113].

Coupled model

Assuming that the level set function $\varphi_1 \in C^1(\Omega\times(t_0,t_f))\cap C^0(\bar{\Omega}\times[t_0,t_f])$, and hence, the keyhole geometry $\Gamma_1(t)$, is a priori known or can be determined by some approach independently from the thermal problem for $t \in [t_0,t_f]$, our coupled model for a keyhole-based laser welding process reads: Find the solid-liquid interface $\varphi_2 \in C^1(\Omega\times(t_0,t_f))\cap C^0(\bar{\Omega}\times[t_0,t_f])$ and the temperature distribution u which is sufficiently smooth, i.e., $u \in \mathcal{C}^0(\bar{\Omega}\times[t_0,t_f])$, $u(\cdot,t)_{|\Omega_i} \in \mathcal{C}^2(\Omega_i(t))$ and $\partial_t u(\cdot,t) \in C^0(\Omega_1(t)\cup\Omega_2(t)\cup\Omega_3(t))$ for $t \in (t_0,t_f)$, such that

$$\begin{cases} \rho c\dfrac{\partial u}{\partial t} - \nabla\cdot(\lambda\nabla u) = 0 & \text{in } \bigcup_{i=1}^{3}\Omega_i(t),\ t\in(t_0,t_f), & (5.88a)\\ u = g_{\mathrm{D}} & \text{on } \Gamma_{\mathrm{D}}\times(t_0,t_f], & (5.88b)\\ \lambda\nabla u\cdot\vec{n} = g_{\mathrm{N}} & \text{on } \Gamma_{\mathrm{N}}\times(t_0,t_f], & (5.88c)\\ \lambda\nabla u\cdot\vec{n} = g_{\mathrm{R}}(u) & \text{on } \Gamma_{\mathrm{R}}\times(t_0,t_f], & (5.88d)\\ u(\cdot,t) = u_{1,2} & \text{on } \Gamma_1(t), & (5.88e)\\ u(\cdot,t) = u_{2,3} & \text{on } \Gamma_2(t), & (5.88f)\\ u(\cdot,t_0) = u_0 & \text{in } \bigcup_{i=1}^{3}\Omega_i(t_0), & (5.88g) \end{cases}$$

$$(\lambda_3\nabla u_3 - \lambda_2\nabla u_2)\cdot\vec{n}_2 = \rho L\vec{V}_{2,3}\cdot\vec{n}_2 \qquad \text{on } \Gamma_2(t), \qquad (5.89)$$

$$\begin{cases} \dfrac{\partial\varphi_2}{\partial t} + \vec{V}_{\Gamma_2}\cdot\nabla\varphi_2 = 0 & \text{in } \Omega\times[t_0,t_f], & (5.90a)\\ \varphi_2(\cdot,t_0) = \varphi_{2,0} & \text{in } \Omega, & (5.90b) \end{cases}$$

holds for given sufficiently smooth data g_{D}, g_{N}, g_{R} u_0, u_{Γ_1}, u_{Γ_2}, and $\varphi_{2,0}$. The unit normal vector $\vec{n}_i(t,\boldsymbol{x})$ to $\Gamma_i(t)$ is defined to point from Ω_i into Ω_{i+1}.

5.6.2 Discretization and computational approach

Due to the fact that the keyhole geometry $\Gamma_1(t)$ is known for all $t \in [t_0,t_f]$, the model of the keyhole process can be decoupled into the familiar subproblems, similar to Sections 5.3 to 5.5, which are discretized individually. As we do not allow for topological changes during the run time, the problem comes down to solving a two-phase Stefan problem on a domain consisting of three subdomains. The computational approach is summarized in Algorithm 9.

5.6.3 Results

The presented model is used to simulate keyhole-based laser welding for the materials aluminum 3.2315, steel 1.0330 and steel 1.4301 whose material properties are given in Table 5.10[‡‡]. While

[‡‡]When no value is known, we use for the steel the corresponding value of iron and for aluminum 3.2315 the value of pure aluminum.

Algorithm 9 Computational approach for the keyhole-based laser welding process.

Input: Ω, Γ_{D}, Γ_{N}, Γ_{R}, $\mathcal{S}_h$, $\varphi_1(t)$, $\varphi_2(t_0)$, $u(t_0)$, $u_{1,2,}$, $u_{3,4}$, f, κ, λ, ρ, c, L, g_{D}, g_{N}, g_{R}, t_0, t_f, $\Delta t_{\max}$, $\Delta t_{\min}$, β_O, β_I, γ.
Output: u_h^n, $\Gamma_{2,h}^n$, $\vec{V}_{2,3,h}^n$ for $n > 0$.

```
1: procedure SIMULATION OF THE KEYHOLE-BASED LASER WELDING PROCESS.
2:     Initialization: $t_n = t_0$, $u_h^n = u(t_0)$, $\varphi_{i,h}^n = \varphi_i(t_0)$, $\Gamma_{i,h}^n = \Gamma_i(t_0)$, $i = 1, 2$.
3:     while $t_n < t_f$ do
4:         if the narrow band method is applied then
5:             Initialize $\Omega_{2,\mathrm{INB}}$ and $\Omega_{2,\mathrm{ONB}}$ with $\varphi_{2,h}^n$.
6:         end if
7:         Compute $\vec{V}_{2,3}^n$ by evaluating (5.89) with an approach presented in Section 4.4.2.
8:         Compute $\Delta t := \min\{\max\{\Delta t_{\mathrm{CFL}}, \Delta t_{\min}\}, \Delta t_{\max}, t_f - t_n\}$.
9:         $t_{n+1} := t_n + \Delta t$.
10:        Compute $\Gamma_{1,h}^{n+1}$ by evaluation level set function $\varphi_{1,h}$ for $t_{n+1}$.
11:        Compute $\varphi_{2,h}^{n+1}$ by solving (5.90) with Algorithm 4.
12:        Compute $\Gamma_{2,h}^{n+1}$ from (reinitialized and volume corrected) $\varphi_{2,h}^{n+1}$.
13:        Define $V_{h,g_{\mathrm{D}}}^1(t_{n+1})$.
14:        Compute $u_h^{n+1}$ by solving (5.88).
15:    end while
16: end procedure
```

Table 5.10 Material properties of steel 1.4301, steel 1.0330 and aluminum 3.2315 [3–5].

symbol	1.4301	1.0330	3.2315	description
u_0	300 K	300 K	300 K	initial temperature
u_a	300 K	300 K	300 K	ambient temperature
u_m	1673 K	1700 K	933 K	melting temperature
u_v	3000 K	3000 K	2800 K	evaporation temperature
ρ	7900 kg/m^3	7860 kg/m^3	2700 kg/m^3	density
c	830 J/kg K	460 J/kg K	900 J/kg K	specific heat capacity
λ_s	15 J/mK	60 J/mK	160 J/mK	thermal conductivity in solid
λ_l	35 J/mK	40 J/mK	110 J/mK	thermal conductivity in melt
L	276000 J/kg	276000 J/kg	386000 J/kg	latent heat
α_{fr}	0.38	0.36	0.20	absorption coefficient

we first comment on the effect of multi-reflections on the keyhole shape, the main aspect of this section is the comparison of experimental results and the simulation findings. For this purpose, our cooperation partner, the Bremer Institut für angewandte Strahltechnik (BIAS), has performed experimental studies with varying parameters for the welding speed V_{L} and laser power P_{L} for each material so that we can compare the simulation outputs with real experimental results.

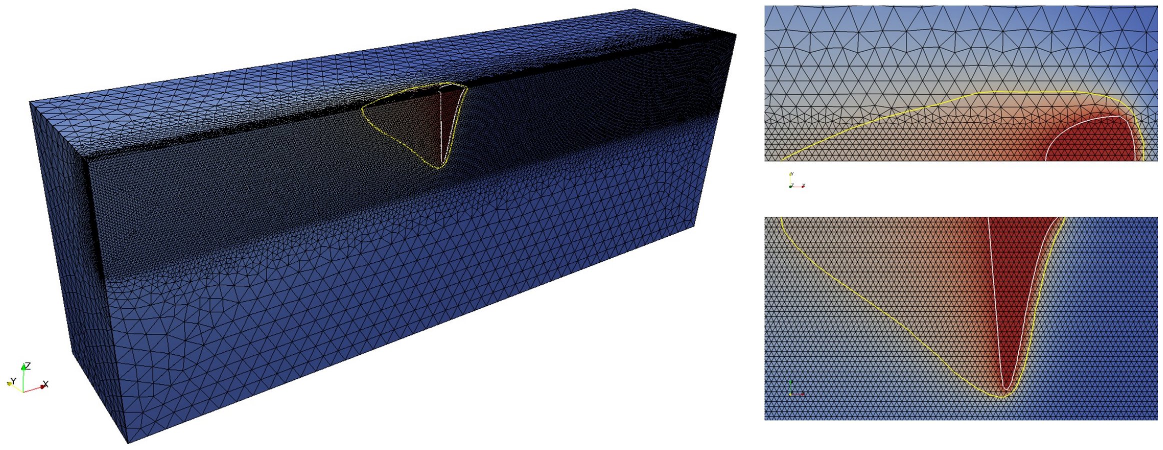


Figure 5.29 Visualization of simulation results for steel 1.0330 with $V_{\mathrm{L}} = 4$ m/min and $P_{\mathrm{L}} = 2000$ W. Keyhole is indicated by the white line, solid-liquid interface is visualized in yellow. The XFEM approach allows for considering mesh independent discontinuities.

Simulation setup: For the numerical simulation, we use an implicit Euler scheme for the time discretization and the polynomial degree $m = 1$ for the eXtended approximation space V_h that is based on the (conventional) function space $V^m_{\mathrm{cg},h}$ so that the temperature is a linear function. The time step size $\Delta t \in [5 \cdot 10^{-6}\, s,\ 5 \cdot 10^{-4}\, s]$ is computed during the run time for each step by the evaluating the CFL condition (4.23). Due to the symmetry of the problem with respect to the welding direction, we only consider half of the workpiece. The corresponding mesh consists of regions with different element sizes where $h_{\min} = 4 \cdot 10^{-5}$ mm and $h_{\max} = 3 \cdot 10^{-3}$ mm to allow for a sufficiently efficient computation with respect to all scenarios but is, of course, not *fitted* to the internal boundaries $\Gamma_{1,h}$ and $\Gamma_{2,h}$ at all. For the boundaries we choose $\partial\Omega = \Gamma_{\mathrm{N}} \cup \Gamma_R$ with Γ_{N} denoting the symmetry plane, on which we impose $g_{\mathrm{N}} = 0$, and $\Gamma_{\mathrm{R}} = \partial\Omega \setminus \Gamma_{\mathrm{N}}$ is the workpiece surface on which we assume cooling by $g_{\mathrm{R}}(u)$, cf. equation (5.29). A section of the simulation output during the process is exemplary shown in Figure 5.29. In lack of better data, we assume that the initial radii of the keyhole at the front, back and side are given by the laser spot radius for all simulation runs.

Effect of multi-reflection on the keyhole geometry:

As described in Section A.5, only the portion α_{fr} of the intensity is absorbed when a ray impinge at the material while the remaining part $1 - \alpha_{\mathrm{fr}}$ is reflected. Hence, the important characteristic in keyhole-based laser welding is the partial absorption of a multiple reflected laser ray at each wall contact. To point out the impact of this effect, Figure 5.30 shows an example of the keyhole shape without and with multi-reflections for the material steel 1.4301 with $\alpha_{\mathrm{fr}} = 0.38$, $V_{\mathrm{L}} = 4$ m/min, and $P_{\mathrm{L}} = 2000$ W. In this example, it can be observed how the inclusion of multi reflections generates a keyhole keeping the same width but getting at least twice the penetration length of the keyhole into the material. Please note that the computed keyhole geometries are not closed for two reasons, see Appendix A.4, firstly, the discretization of the workpiece in z-layers and, secondly, because we stop the computational approach if $\left\| \boldsymbol{x}^f - \boldsymbol{x}^r \right\|_2 \leq \tau$ or

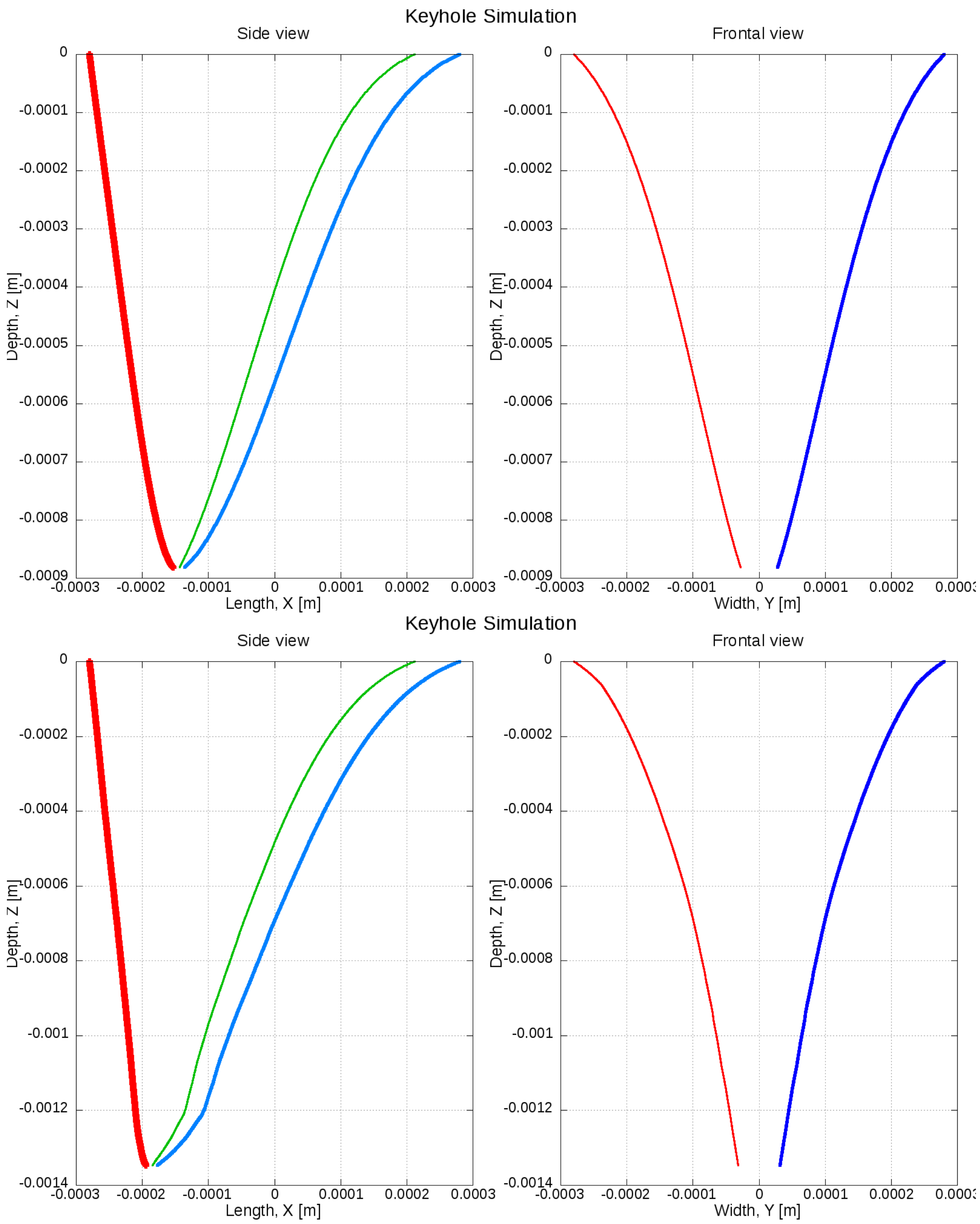
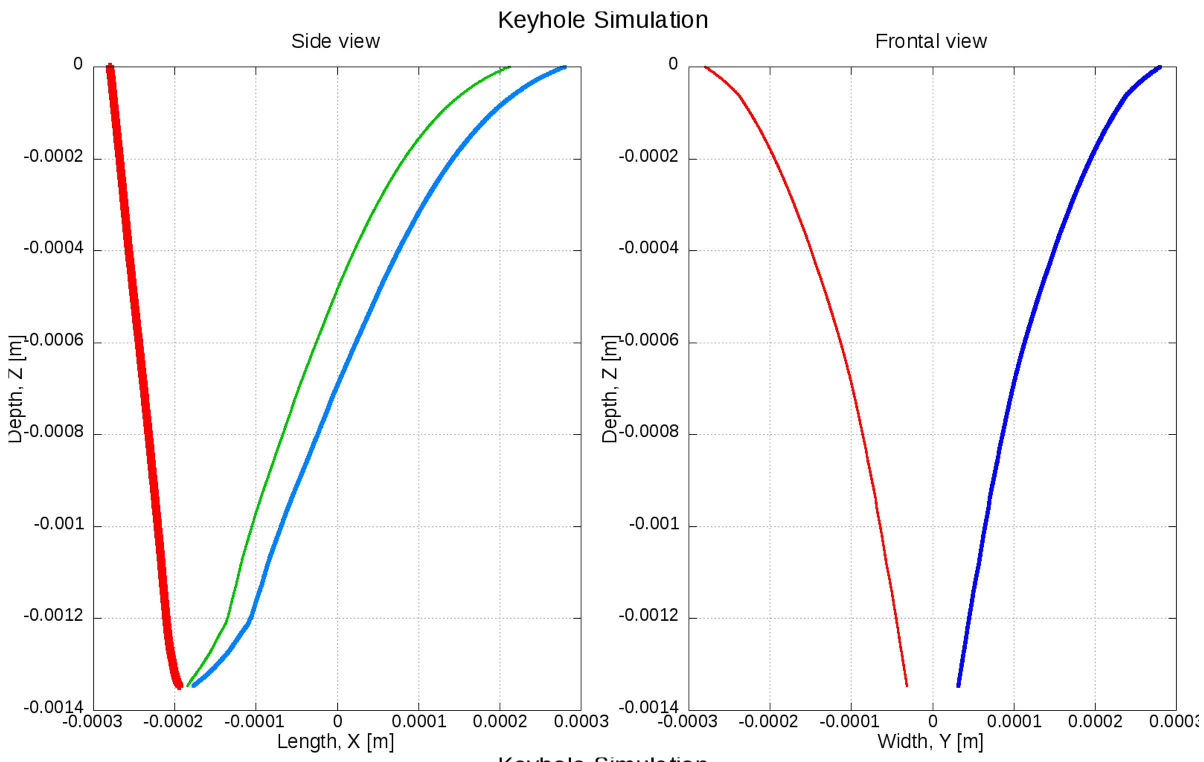


Figure 5.30 Keyhole geometries without multi-reflections (top) and with multi-reflections (bottom) for the material steel 1.4301 and the parameters $\alpha_{\mathrm{fr}} = 0.38$, $V_{\mathrm{L}} = 4$ m/min, and $P_{\mathrm{L}} = 2000$ W. The red and blue lines correspond to the keyhole wall and the green line indicates the position of the heat source. On the left hand side, the side view in the x-z plane is shown and on the right hand side, the frontal view in the x-z plane is visualized.



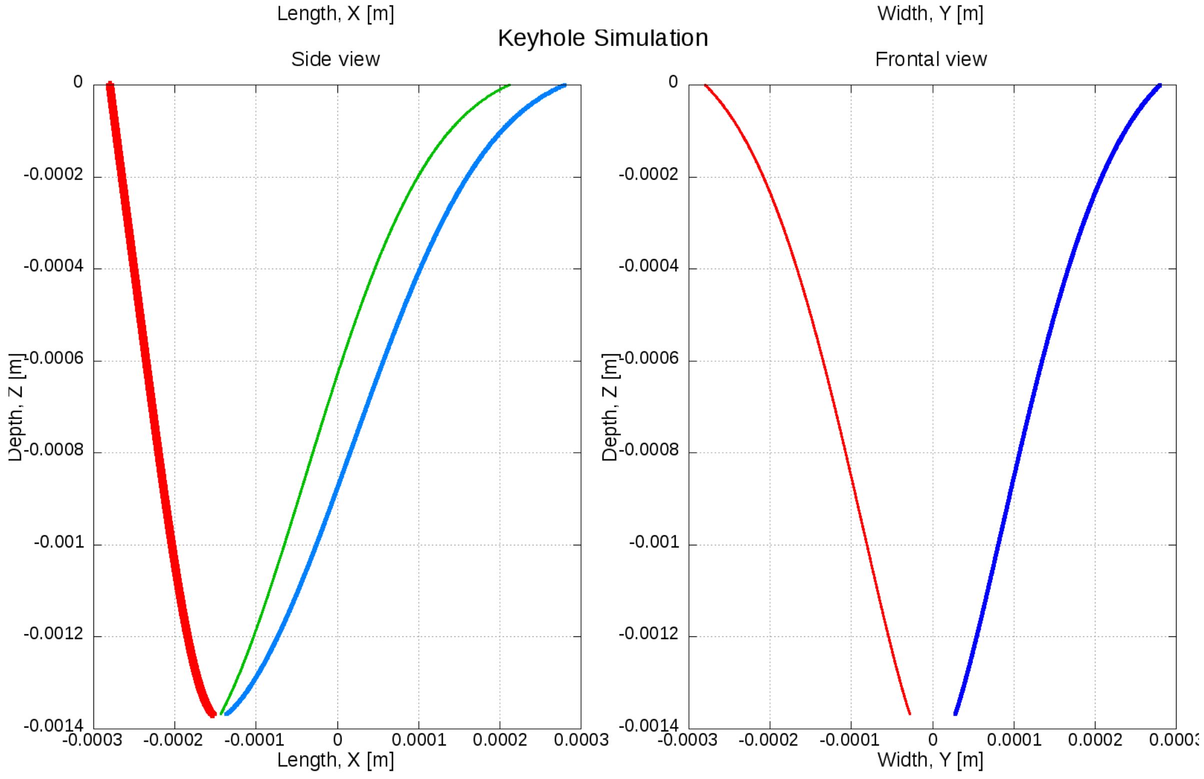


Figure 5.31 Keyhole geometries with multi-reflections for $\alpha_{\mathrm{fr}} = 0.38$ (top) and without multi-reflections but $\alpha_{\mathrm{fr}} \approx 0.60$ (bottom) for steel 1.4301 using the parameters $V_{\mathrm{L}} = 4$ m/min and $P_{\mathrm{L}} = 2000$ W. The red and blue lines correspond to the keyhole wall and the green line indicates the position of the heat source. On the left hand side, the side view in the x-z plane is shown and on the right hand side, the frontal view in the x-z plane is visualized.

Table 5.11 Experimental data and numerical results for aluminum 3.2315.

P_L	V_L	z_exp	$w_\mathrm{exp,0.5}$	z_sim	w_sim
3000 W	5 m/min	1.31 mm	2.22 mm	1.32 mm	1.75 mm
3000 W	6 m/min	1.25 mm	1.66 mm	1.25 mm	1.67 mm

Table 5.12 Experimental data and numerical results for steel 1.0330.

P_L	V_L	z_exp	$w_\mathrm{exp,0.5}$	z_sim	w_sim
2000 W	4 m/min	1.45 mm	1.20 mm	1.48 mm	1.00 mm
2000 W	5 m/min	1.10 mm	1.11 mm	1.27 mm	0.99 mm

Table 5.13 Experimental data and numerical results for steel 1.4301.

P_L	V_L	z_exp	w_exp	z_sim	w_sim
2000 W	4 m/min	1.53 mm	1.24 mm	1.45 mm	1.02 mm
2000 W	5 m/min	1.21 mm	1.13 mm	1.26 mm	1.00 mm

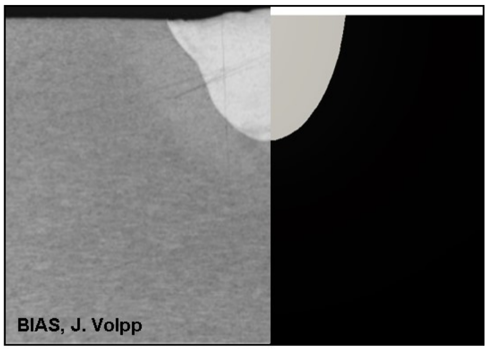
$\|\boldsymbol{x}^s-\|_2 \leq \tau$, where $\boldsymbol{x}^f$ denotes the keyhole front wall, $\boldsymbol{x}^r$ the keyhole rear wall, and $\boldsymbol{x}^s$ the keyhole side wall. The value $0 < \tau \ll 1$ is the tolerance of the stopping criterion.

For simplicity, many publications that address welding processes neglect multi reflections. Instead, the Fresnel absorption rate is chosen significantly higher in order to compensate for their impact. Using the same scenario as before, Figure 5.31 shows the keyhole shape with multi-reflections compared to a situation with $\alpha_\mathrm{fr} \approx 0.6$ where no reflections are considered. This can be done, but there is a high risk of choosing a wrong scaling factor. Notice in the example of Figure 5.30 that the same penetration depth was obtained, but the geometric shapes differ, in particular in the lowest part of the keyhole. Furthermore, the scaling factor taken here to get the same penetration depth was around 1.58, which is not straightforward to know and might only be useful for this specific combination of material, laser configuration and process speed.

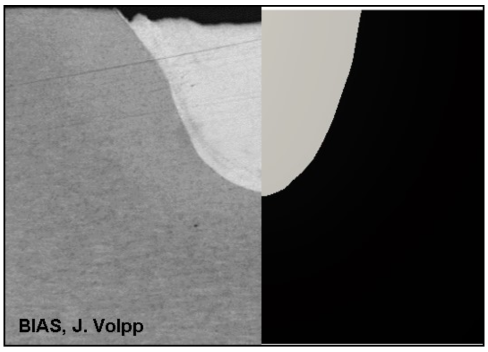
Keyhole-based laser welding for different materials:

Now, we consider keyhole-based laser welding for the materials steel 1.4301, steel 1.0330 and aluminum 3.2315. As mentioned, the BIAS has performed several experiments for each material using different parameters for the welding speed V_L and laser power P_L. The experimental design is as visualized in the introduction. In more detail, specimens of size 100 mm $\times$ 40 mm $\times$ 6 mm are welded by a single-mode fiber laser IPG YLR-1000SM with Gaussian beam profile that is applied to the $x-y$ plane of the metal sheets. In all experiments, no shielding gas is present.

The results of the experiments and simulation runs for each material and process parameter configuration are given in the Table 5.11 for aluminum 3.3215, in Table 5.12 for steel 1.0330, and in Table 5.13 for steel 1.4301. In these tables, it can be seen that the simulation always

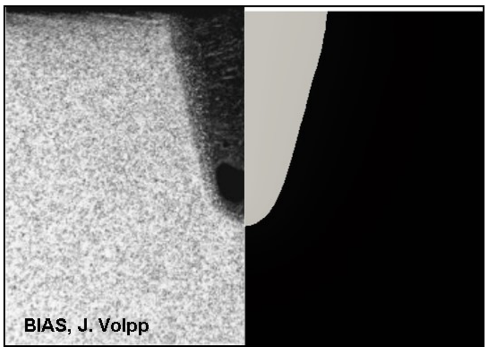


(a) aluminum 3.2315, $P_{\mathrm{L}} = 3000$ W, $V_{\mathrm{L}} = 5$ m/min.

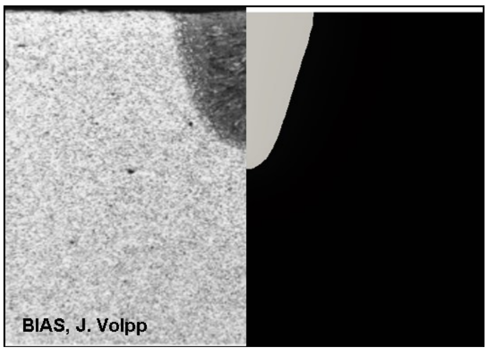


(b) aluminum 3.2315, $P_{\mathrm{L}} = 3000$ W, $V_{\mathrm{L}} = 6$ m/min.

Figure 5.32 Melt pool geometry: Experimental and numerical results for aluminum 3.2315.



(a) steel 1.0330, $P_{\mathrm{L}} = 2000$ W, $V_{\mathrm{L}} = 4$ m/min.



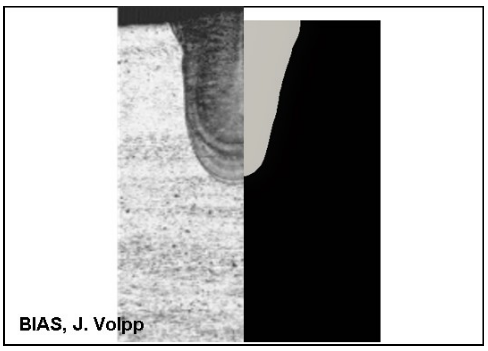
(b) steel 1.0330, $P_{\mathrm{L}} = 2000$ W, $V_{\mathrm{L}} = 5$ m/min.

Figure 5.33 Melt pool geometry: Experimental and numerical results for steel 1.0330.

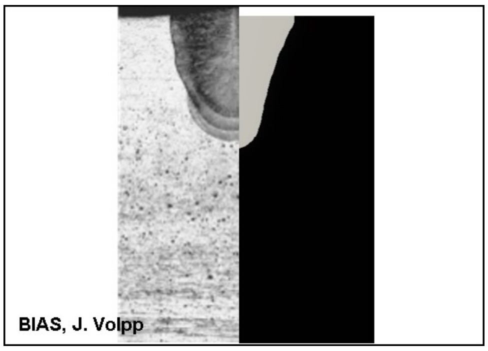
underestimates the melt pool width while the measured and simulated melt pool depth coincides very well. Comparing the experimental data to the numerical results, cf. Figure 5.32 for aluminum, Figure 5.33 for steel 1.0330, and Figure 5.34 for steel 1.4301, the smaller melt pool width in all simulation runs is obviously a consequence of neglecting the melt dynamics in the model which would have a significant influence in the upper part of the melt pool due to buoyancy forces. This assumption is supported by the fact that the difference in the melt pool width decreases for increasing welding speed.

5.7 Multiphase flow

The last example considered in this thesis is a multiphase flow problem. Motivated by the benchmarks considered in [114, 115], we model the dynamics of two rising droplets in a fluid and show that the presented method and its implementation can also be used to model and simulate problems involving fluid dynamics. Therefore, we assume that all fluids are immiscible and incompressible. To simplify the setting, we do not allow the droplets to touch each other.



(a) steel 1.4301, $P_{\mathrm{L}} = 2000$ W, $V_{\mathrm{L}} = 4$ m/min.



(b) steel 1.4301, $P_{\mathrm{L}} = 2000$ W, $V_{\mathrm{L}} = 5$ m/min.

Figure 5.34 Melt pool geometry: Experimental and numerical results for steel 1.4301.

While we reduce our setting to three fluids, the model and the implementation can be extended to model n-phase flow in a straightforward way. As this problem is chosen as an example for applications that can be considered in the future rather than as an real-world application which has to be thoroughly studied, we keep its presentation and the discussion of results very brief.

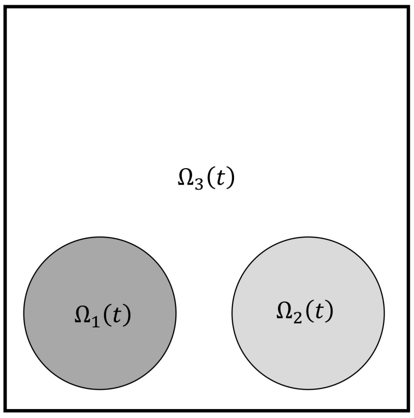
5.7.1 Modeling approach

Adapting the presentation of a two-phase flow problem in [115], let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be a fixed domain with $\partial\Omega$ polygonal and let $[t_0, t_f]$ denote the considered time interval. We assume that the domain Ω consists of different immiscible incompressible fluids that are described by the disjoint subdomains $\Omega_i(t)$, $i = 1, 2, 3$. Therein, the fluids represented by Ω_1 and Ω_2 are completely surrounded by Ω_3, cf. Figure 5.35(a). The interfaces separating the fluids are denoted by Γ_1 and Γ_2. The dynamics of all fluids are modeled by the Navier-Stokes equations [116, 117] which describes the evolution of the velocity $\boldsymbol{u}\colon \Omega \times [t_0, t_f] \to \mathbb{R}^d$ and the pressure $p\colon \Omega \times [t_0, t_f] \to \mathbb{R}$ by

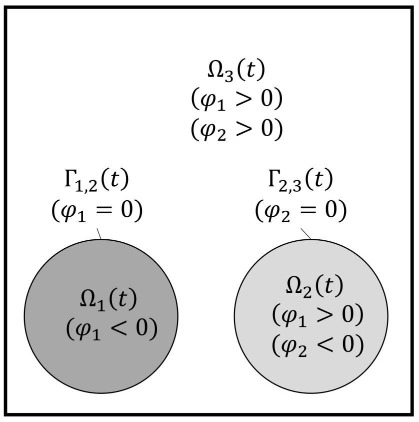
$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right) + \nabla p - \nabla\cdot\boldsymbol{\sigma} = \rho\boldsymbol{g} \qquad \text{in } \bigcup_{i=1}^{3}\Omega_i(t),\ t \in (t_0, t_f), \tag{5.91a}$$

$$\nabla\cdot\boldsymbol{u} = 0 \qquad \text{in } \bigcup_{i=1}^{3}\Omega_i(t),\ t \in (t_0, t_f), \tag{5.91b}$$

where $\boldsymbol{\sigma} = -p\boldsymbol{I}_d + \mu\boldsymbol{D}(\boldsymbol{u})$ denotes the stress tensor, ρ the density with $\rho|_{\Omega_i} = \rho_i$, μ the dynamic viscosity with $\mu|_{\Omega_i} = \mu_i$, and $\boldsymbol{g}$ is the gravitational force.



(a) Problem setting.



(b) Domain decomposition using the hierarchical level set method.

Figure 5.35 Problem setting for the multiphase flow example and decomposition of the hold-all domain Ω into subdomains Ω_1, Ω_2, Ω_3 by the zero level sets Γ_1, Γ_2 of hierarchically ordered level set functions φ_1, φ_2.

The different fluids are coupled by the interface conditions

$$[\![\boldsymbol{\sigma} \cdot \vec{n}_1]\!] = \tau_1 \mathcal{C}_1 \cdot \vec{n}_1 \qquad \text{on } \Gamma_1(t),\ t \in [t_0, t_f], \tag{5.92a}$$

$$[\![\boldsymbol{u}]\!] = 0 \qquad \text{on } \Gamma_1(t),\ t \in [t_0, t_f], \tag{5.92b}$$

$$[\![\boldsymbol{\sigma} \cdot \vec{n}_2]\!] = \tau_2 \mathcal{C}_2 \cdot \vec{n}_2 \qquad \text{on } \Gamma_2(t),\ t \in [t_0, t_f], \tag{5.92c}$$

$$[\![\boldsymbol{u}]\!] = 0 \qquad \text{on } \Gamma_2(t),\ t \in [t_0, t_f], \tag{5.92d}$$

where

$$\mathcal{C}_1 = \nabla \cdot \vec{n}_1, \tag{5.93}$$

and

$$\mathcal{C}_2 = \nabla \cdot \vec{n}_2, \tag{5.94}$$

are the curvatures of the interfaces, τ_1, τ_2 denote the surface tensions of the fluids, and $\vec{n}_i$ are the outwards pointing unit normal vectors to Ω_i.

Usually, $\Gamma_1(t)$ and $\Gamma_2(t)$ are only known for t_0 while for $t > t_0$, their positions are part of the solution. Therefore, we couple the problem given by (5.91a) to (5.92d) with the level set method, see Section 2.1, and describe the evolution of $\Gamma_1(t)$ and $\Gamma_2(t)$ for $t \in (t_0, t_f]$ by the transport problem (2.7).

5.7.2 Coupled model using the hierarchical level set method

The described setting can be modeled using the hierarchical level set method as follows: Given the hold-all domain Ω and a time interval $[t_0, t_f]$, let $\Omega_1(t)$ and $\Omega_2(t)$ represent the initial positions of the droplets whose boundaries are defined by the interfaces $\Gamma_{1,2}(t_0) = \Gamma_1(t_0)$ and $\Gamma_{2,3}(t_0) = \Gamma_2(t_0)$. Then, we first introduce the level set functions $\varphi_i \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times$

$[t_0, t_f])$ as signed distance function to $\Gamma_i(t_0)$ with $\varphi_1(\boldsymbol{x}, t_0) < 0$ for $\boldsymbol{x} \in \Omega_1(t_0)$, $\varphi_2(\boldsymbol{x}, t_0) < 0$ for $\boldsymbol{x} \in \Omega_2(t_0)$, and $\varphi_i(\boldsymbol{x}, t_0) > 0$ for $\boldsymbol{x} \in \Omega_3(t_0)$, $i = 1, 2$, cf. Figure 5.35(b). As there is no natural hierarchy given the physics for the droplets, we can make an arbitrary choice and, thus, assume that φ_1 is of higher hierarchy than φ_2. Then, the coupled model describing the evolution of the droplets in another fluid for $t \in [t_0, t_f]$ is given by: Given the interface positions $\Gamma_{1,2}(t_0)$ and $\Gamma_{2,3}(t_0)$ and sufficiently smooth data, find the locations of the interfaces $\Gamma_{1,2}(t)$ and $\Gamma_{2,3}(t)$ that are the zero level sest of $\varphi_i \in C^1(\Omega \times (t_0, t_f)) \cap C^0(\bar{\Omega} \times [t_0, t_f])$ and the velocity $\boldsymbol{u}\colon \Omega \times [t_0, t_f] \to \mathbb{R}^d$ as well as the pressure $p\colon \Omega \times [t_0, t_f] \to \mathbb{R}$ such that

$$\left\{\begin{aligned}
\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right) + \nabla p - \nabla\cdot\boldsymbol{\sigma} &= \rho\boldsymbol{g} && \text{in } \bigcup_{i=1}^{3}\Omega_i(t),\ t \in (t_0, t_f) && (5.95a)\\
\boldsymbol{u} &= g_{\mathrm{D}} && \text{on } \Gamma_{\mathrm{D}} \times (t_0, t_f], && (5.95b)\\
\nabla\cdot\boldsymbol{u} &= 0 && \text{in } \bigcup_{i=1}^{3}\Omega_i(t),\ t \in [t_0, t_f], && (5.95c)\\
[\![\boldsymbol{\sigma}\cdot\vec{n}_{1,2}]\!] &= \tau_1\mathcal{C}_1\cdot\vec{n}_{1,2} && \text{on } \Gamma_{1,2}(t),\ t \in [t_0, t_f], && (5.95d)\\
[\![\boldsymbol{u}]\!] &= 0 && \text{on } \Gamma_{1,2}(t),\ t \in (t_0, t_f), && (5.95e)\\
[\![\boldsymbol{\sigma}\cdot\vec{n}_{2,3}]\!] &= \tau_2\mathcal{C}_2\cdot\vec{n}_{2,3} && \text{on } \Gamma_{2,3}(t),\ t \in [t_0, t_f], && (5.95f)\\
[\![\boldsymbol{u}]\!] &= 0 && \text{on } \Gamma_{2,3}(t),\ t \in [t_0, t_f], && (5.95g)\\
\boldsymbol{u}(\cdot, t_0) &= \boldsymbol{u}_0 && \text{in } \textstyle\bigcup_{i=1}^{3}\Omega_i(t_0), && (5.95h)
\end{aligned}\right.$$

$$\left\{\begin{aligned}
\frac{\partial\varphi_1}{\partial t} + \boldsymbol{u}\cdot\nabla\varphi_1 &= 0 && \text{in } \Omega \times [t_0, t_f], && (5.96a)\\
\frac{\partial\varphi_2}{\partial t} + \boldsymbol{u}\cdot\nabla\varphi_2 &= 0 && \text{in } \Omega \times [t_0, t_f], && (5.96b)
\end{aligned}\right.$$

hold.

5.7.3 Discretization and computational approach

We decouple the problem given by (5.95a) to (5.96b) in a similar way as the applications presented in the previous sections by using an explicit approach. As a result, we can consider the subproblems (5.95) and (5.96) separately. While the level set problems for $\varphi_i(\cdot, t)$ are considered using the same discretization approach as before, we approximate the Navier-Stokes equations, which model the fluid dynamics, by replacing the non-linearity $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u})$ with $(\boldsymbol{w}\cdot\nabla)\boldsymbol{u})$. In the numerical scheme, $\boldsymbol{w}$ will correspond to the velocity of the previous time step. The resulting problem is scaled to have enough regularity on the time derivative and then discretized using Rothe's method as before. For approximating the time derivative, we use the implicit Euler scheme so that the problem (5.95) reduces to a series of Oseen problems, see e.g. [118]. With respect to spatial dimension, we use the Taylor-Hood $\mathcal{P}^2$-$\mathcal{P}^1$ as conventional finite element

Table 5.14 Material properties of the considered fluids represented by Ω_i.

symbol	Ω_1	Ω_2	Ω_3	**description**
μ	1.0	0.1	10	dynamic viscosity [Pa s]
ρ	100	1.0	1000	density $\left[\frac{\text{kg}}{\text{m}^3}\right]$
τ	24.5	1.96	-	surface tension $\left[\frac{\text{N}}{\text{m}}\right]$

approximation space that is hierarchically enriched with respect to the interface positions $\Gamma_i(t)$. The interface conditions are imposed using Nitsche's method.

Contrary to the usual characterization of the curvature of the interfaces by using the Laplace-Beltrami description, we use a more explicit approach by modeling the curvature as described in (5.93) and (5.94). For this purpose, we introduce the quadratic approximations $\varphi_{i,2h}$, cf. Section 4.3.2.1, and compute the curvature for these functions which in turn are then used as approximations for computing a solution to problem (5.95).

5.7.4 Results

We now present simulation results of the rising droplets example for $N_{\text{dom}} = 3$ and $d = 2$.

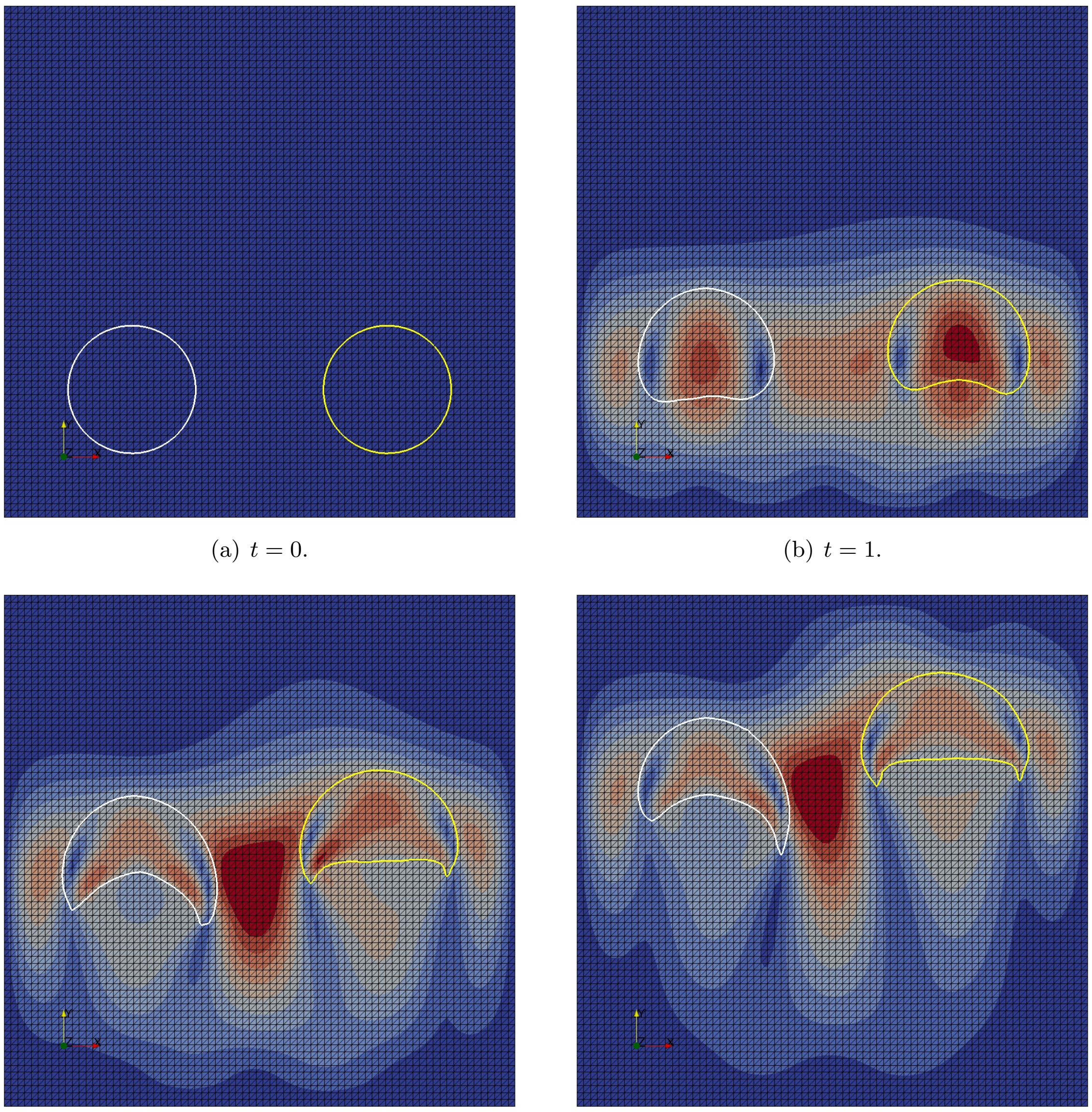
Setup: On $\Omega = [-1, 1] \times [-0.5, 1.5]$ consider for $t \in [0, 5]$ the evolution of the droplets represented by $\Omega_1(t)$ and $\Omega_2(t)$ in the surrounding fluid $\Omega_3(t)$. The configuration is chosen such that we have $\Omega_1(t) \cap \Omega_2(t) = \emptyset$ and $\partial\Omega_i(t) \cap \partial\Omega = \emptyset$, $i = 1, 2$. On the boundary $\partial\Omega = \Gamma_{\text{D}}$, we assume non-slip condition, i.e., $g_{\text{D}} = 0$ in condition (5.95b). Initially, we have $u(\cdot, t_0) = 0$ and the droplets $\Gamma_1(t_0)$ and $\Gamma_2(t_0)$ are given by the zero level sets of the functions

$$\varphi_i(\boldsymbol{x}, t) := \|\boldsymbol{x} - \boldsymbol{c}_i\|_2 - r_i, \qquad i = 1, 2, \tag{5.97}$$

with $\boldsymbol{c}_1 = (-0.5, 0)^T$, $r_1 = 0.25$, $\boldsymbol{c}_2 = (0.5, 0)^T$, and $r_2 = 0.25$. The (fictional) material parameters used for in this example are given in Table 5.14 [114, 115]. Due to the buoyancy effects caused by the different material parameters, both droplets will rise and change their shape during the considered time interval.

For the numerical simulation, we used an implicit Euler scheme for the time discretization and approximate the solution of the problem in the eXtended approximation space V_h that is constructed by hierarchically enriching the conventional finite element approximation space given by the $\mathcal{P}^2$-$\mathcal{P}^1$ Taylor-Hood element. The computational mesh consists of $N_{\text{el}} = 2 \times 75^2$ elements and the time step size $\Delta t \in [10^{-5}, 5 \cdot 10^{-3}]$ is computed by considering the CFL conditions (4.24). The narrow band parameters for φ_i, $i = 1, 2$, are $\beta_{\text{I}} = 2$, $\beta_{\text{O}} = 4$, and $\gamma = 6$.

Results: Visualizations of the numerical results of the computed velocity for different points in time are given in Figure 5.36. Therein, we can see that while both droplets rise similarly at



(a) $t = 0$. (b) $t = 1$.

(c) $t = 2.5$. (d) $t = 4$.

Figure 5.36 Rising droplets: Results of the computed velocity for different time steps. The droplets Γ_1 and Γ_2 are shown in white and yellow.

the beginning, cf. Figure 5.36(b), the droplet represented by Ω_2 is subject to a larger buoyancy force. Thus, it rises faster than the droplet represented by Ω_1 and the resulting velocity field impacts the shape and evolution of Ω_1, cf. Figure 5.36(c). This effects increases during the evolution of the droplets in time, cf. Figure 5.36(d).

Chapter 6

Summary and outlook

We conclude the thesis with a short summary and give an outlook about open questions and future work.

6.1 Summary

Motivated by several multiphysics problems arising from the modeling of industrial applications and shortcomings of conventional finite element methods in regards to their numerical simulation, a hierarchical eXtended finite element method for the solution of multiphysics problems with an arbitrary number of possible evolving and dissolving discontinuities has been presented. Belonging to the class of unfitted methods, the presented approach is based of the following ideas:

- The physical domain was embedded in a hold-all domain and decomposed in a geometrically consistent way into subdomains by introduced hierarchical ordered level set functions. The (active parts of the) zero levels of these functions represented the domain boundaries. By design, the approach guarantees that no overlapping regions or voids can arise.
- Based on the zero level sets, enriched approximation spaces have been defined using the Heaviside function as the enrichment function. Due to the hierarchical order of the discontinuities, the enriched quantities have been easily defined.
- For the imposition of boundary and interface conditions, Nitsche's method has been used. The weighted average operators similar to [49].
- For the discretization of time-dependent problems, Rothe's method has been used and the time derivative has been approximated using the implicit Euler scheme. Thereby, a series of quasi-stationary problems was derived, where only a right-hand-side term depends on the old time step. Problems with not enough regularity have been rescaled.

While the derived method can be implemented as stand-alone library, it is especially suited to be implemented into a framework that makes use of automated code generation. The advantage of automated code generation is its convenience for user and that it allows for the rapid implementation and solution of different kinds of problems. The open source project `FEniCS` offers such a framework for conventional finite element method and, in addition, is widely used. Therefore, the presented method has been implemented into a toolbox called `miXFEM` for the `FEniCS` framework. As a result, it can be used for the automated solution of PDE problems involving any kind and number of possible evolving discontinuities. In order for this to work, `miXFEM` enhanced several core components of `FEniCS`:

- For the definition of discretized variational formulations in the high-level unified domain language `UFL`, several concepts have been exploited which have already been implemented.
- To understand and reinterpret these expressions, a new form compiler called `miXFFC` has been developed. This compiler generates the corresponding low-level `C++` code for the numerical solution of problems involving discontinuities.
- To make use of the generated code, a new library called `miXDOLFIN` has been implemented which provides all methods related to the hierarchical eXtended finite element method.
- As the solution of multiphysics problems often requires additional methods, a full-featured level set toolbox has been added to `miXFEM`. The toolbox provides maintaining methods such as reinitialization via fast marching and volume correction methods and allows for a efficient solution of the level set problem by restricting the problem to a narrow band region.

The implemented framework `miXFEM` and its components have been validated using several examples, partly with known analytical solution. Then the toolbox was used for the numerical simulation of several applications motivating this thesis:

- Using the laser welding process for the production of hybrid joints as motivation, it has been shown that the hierarchical eXtended finite element method enhances and simplifies the modeling of the process. Moreover, it as shown that topology changes and multiple junctions can be considered naturally.
- A reduced model of a laser-based thermal upsetting process has been considered. While the evolution of the geometry was defined by an analytical model instead of considering the completely coupled problem including the fluid dynamics and the capillary boundary, it was demonstrated that these type of problems can be simulated with `miXFEM`.
- A highlight of the thesis is the application of the approach for the simulation of keyhole-based laser welding processes, where the numerical results matched the experimental data well.

- While the presentation of the capabilities of the toolbox has been demonstrated in detail for the aforementioned applications, it was also shown that the developed approach can be used to solve problems from other research areas such as multiphase flow. Due to the generality of the method, it can also be used to solve problems resulting from other fields of research such as structural mechanics or fluid-structure interaction.

All in all, it has been shown that the hierarchical eXtended finite element method and its implementation `miXFEM` considerably enhanced the modeling and simulation capabilities and, by using automated code generation, provide a very flexible framework for the solution of multiphysics problems in both 2D and 3D.

6.2 Outlook and future work

The presentation of the hierarchical eXtended finite element method in this thesis is primarily focused on introducing the essential ideas of the method and its implementation into the automated code generation framework. As a result, we neither included extensive analyses of proposed method and the considered problems, nor did we present the technical details of the toolbox in too much depth. Both are topics for upcoming publications. In addition to this, there are several aspects concerning the hierarchical eXtended finite element method, its implementation `miXFEM`, and the considered examples which can be addressed in future work:

- First of all, the implementation of higher order approximations of the interface should be addressed. This is of particular interest when considering fluid dynamics, which are often discretized using Taylor-Hood elements.
- For the discretization of time-dependent problems, we used Rothe's method and rescaled the considered problems to gain enough regularity, if necessary. Doing so restricts us to first order convergence in the time. In contrast to this, space-time elements allow for higher-order approximations and convergence rates [50]. Hence, it would be interesting to combine our method with these methods.
- When considering more complex applications that may involve different scales, it is desirable to add adaptive mesh refinement algorithms to automatically adapt the mesh. While implementing mesh refinement approaches merely based on the geometry and the zero level sets, the idea of developing, for example., goal-orientated adaptivity algorithms to automatically refine the mesh is very interesting.
- At the moment, `miXFEM` only makes use of shared memory parallelization, i.a., for the assembling of forms. To significantly decrease the computational time, the implemented methods have to be parallelized using distributed memory. In fact, the design of the toolbox should allow for using the conventional domain decomposition approaches.

- In regards to the linear algebra solvers, we currently use the standard solvers provided by PETSc or similar libraries. We neither have performed studies with respect to the efficiency nor did we test more sophisticated methods such as multigrid algorithms.

Furthermore, all examples and applications considered in this thesis have been used to demonstrate the benefit and convenience of the hierarchical eXtended finite element method and, in particular, its implementation into a framework which makes use of automated code generation. Therefore, most problems have been considered in terms of reduced models to illustrate different features of miXFEM. While, we are currently enhancing the models of the applications to also consider the fluid dynamics and capillary surfaces, which can be described using the Laplace-Beltrami operator, future work could address the following aspects:

- In regards to the presented reduced models, we emphasized the capabilities of the method and its implementation instead of focusing on each example in detail. As a result we dropped i.a. a thorough analysis with methods from the functional analysis of each problem. This is something we should catch on in the future.
- Moreover, the current solution strategy for couped models is to decouple the problem into subproblems in an explicit way and solve them in succession. Instead of doing that, we could also use a fix point scheme to iterate the subproblems against each other, similar to [29, Chap. 9]. A comparison of the results would be interesting.
- Another important research topic is the solution of PDE constrained optimization problems. All mentioned applications depend on an appropriate choice of process parameters. For the industrial application, the identification of optimal process windows are of great interest.
- Also, the modeling and application of the method to consider more complex fluid dynamics problems, such as multiphase flow with mass transport, could be addressed.

Last but not least, we want to mention that one of the few drawbacks of being engaged in different research fields and multidisciplinary projects has been that there was not enough time to follow all developments and updates of FEniCS. As a result, miXFEM is currently not compatible with up-to-date versions of FEniCS but relies on an older code base making it less attractive for users. To make the hierarchical eXtended finite element method and its implementation in miXFEM more useful, the framework should be refactored and implemented as toolbox which is compatible to the new API of FEniCS.

Appendix A

Keyhole model

The used keyhole model is adapted from the approach presented in [110]. Basically, it uses an analytical model of the energy balance at the keyhole wall to approximate the keyhole shape for a quasi-stationary situation. For given material and laser parameters, the heat conduction is then approximated by a moving line source [111, 112] and the local inclination angle yielding the keyhole geometry is computed using a point-by-point scheme that compares absorbed power and conduction losses.

A.1 Laser model

In contrast to the previous applications considered in Sections 5.4 and 5.5, the laser beam is now moved along the welding line. To take this into account, a Cartesian coordinate system $(\tilde{x}, y, z)$ is introduced, where $\tilde{x}$ is the welding direction and -z is the laser beam direction. After substituting the $\tilde{x}$-coordinate by $x = \tilde{x} - \vec{V}_\mathrm{L}(t - t_0)$ for a constant velocity $\vec{V}_\mathrm{L}$ in x-direction, we end up with a *quasi-stationary* situation in a coordinate system related to the laser position. In addition, we introduce the corresponding cylindrical coordinate system (r, ϕ, z), since we later want to define the keyhole geometry using these coordinates. As before, we only consider a Gaussian-like distributed intensity profile whose intensity maximum is given by

$$I_0 = \frac{2P_\mathrm{L}}{r_\mathrm{f0}^2 \pi} \tag{A.1}$$

with P_L denoting the laser power and r_f0 is the laser beam radius in focus height z_0 so that a Gaussian intensity profile is given by

$$I_\mathrm{L}(r, \phi, z) = I_0 \left(\frac{r_\mathrm{f0}}{r_\mathrm{f}(z)} \right)^2 \cdot \exp\left(-\frac{2r^2}{r_\mathrm{f}(z)^2} \right) \tag{A.2}$$

where $r_\mathrm{f}(z)$ is the current laser beam width

$$r_\mathrm{f}(z) = r_\mathrm{f0}\left(1+\left(\frac{z-z_0}{z_\mathrm{Ray}}\right)^2\right)^{\frac{1}{2}} \tag{A.3}$$

that depends on the Rayleigh length z_Ray.

A.2 Heat conduction at the keyhole wall

To approximate the heat conduction at the keyhole wall, an analytic approach based on a moving line source [111, 112] is used, whose power per unit depth P' and location $\boldsymbol{x}^\mathrm{HS}(z)$ depend on $z \in \Omega$. Due to the latter, we have a different coordinate system $(\hat{r}_{\boldsymbol{x}^\mathrm{HS}(z)}, \hat{\phi}_{\boldsymbol{x}^\mathrm{HS}(z)}, z)$ for every $z \in \Omega$ with origin $\boldsymbol{x}^\mathrm{HS}(z)$. To simplify the notation in this section, we define $(\hat{r}, \hat{\phi}, z) := (\hat{r}_{\boldsymbol{x}^\mathrm{HS}(z)}, \hat{\phi}_{\boldsymbol{x}^\mathrm{HS}(z)}, z)$ and keep in mind that all $\hat{\cdot}$ coordinates depend on $\boldsymbol{x}^\mathrm{HS}(z)$ and, especially, on z.

Using the moving line heat source model and the introduced notation, the temperature field $u(\hat{\boldsymbol{x}}, t)$ for a *quasi-stationary* situation with $\partial_t u = 0$ in a semi-infinite work-piece is given by

$$u(\hat{r}, \hat{\phi}, z) = u_a + \frac{P'(\hat{r}, \hat{\phi}, z)}{2\pi\lambda} \cdot K_0(\mathrm{Pe}'\hat{r}) \exp\left(-\mathrm{Pe}'\hat{r}\cos(\hat{\phi})\right) \tag{A.4}$$

with $K_0(\hat{\boldsymbol{x}})$ being the modified Bessel function of second kind and zeroth order, and Pe' is the modified Peclet number

$$\mathrm{Pe}' = \frac{\vec{V}_\mathrm{L}\rho c}{2\lambda} = \frac{\vec{V}_\mathrm{L}}{2\kappa}, \tag{A.5}$$

cf. [110, 112]. Since evaporating temperature u_{Γ_1} has to be reached for all points $(\hat{r}, \hat{\phi}, z)$ at the keyhole wall, we can transform (A.4) to get a formula to compute the value of the heat source $P'(\hat{r}, \hat{\phi}, z)$ which is then given by

$$P'(\hat{r}, \hat{\phi}, z) = (u_{\Gamma_1} - u_a) \cdot 2\pi\lambda \frac{1}{K_0(\mathrm{Pe}'\hat{r})} \exp\left(\mathrm{Pe}'\hat{r}\cos(\hat{\phi})\right). \tag{A.6}$$

In general, thermal conduction can be described by Fourier's law. Assuming that heat propagates isotropic which means that the isothermal surfaces are concentric spheres, we neglect heat flux in z-direction and consider the heat flow only in radial direction, i.e.

$$\dot{q}(\hat{r}, \hat{\phi}, z) = -\lambda\nabla u \approx -\lambda\partial_{\hat{r}} u. \tag{A.7}$$

Substituting u with (A.4) and differentiating leads to

$$\dot{q}(\hat{r}, \hat{\phi}, z) = -\lambda\frac{\partial u}{\partial \hat{r}} = \frac{P'(\hat{r}, \hat{\phi}, z)}{2\pi\exp\left(\mathrm{Pe}'\hat{r}\cos\hat{\phi}\right)}\mathrm{Pe}'\left[K_0(\mathrm{Pe}'\hat{r})\cos\hat{\phi} + K_1'(\mathrm{Pe}'\hat{r})\right], \tag{A.8}$$

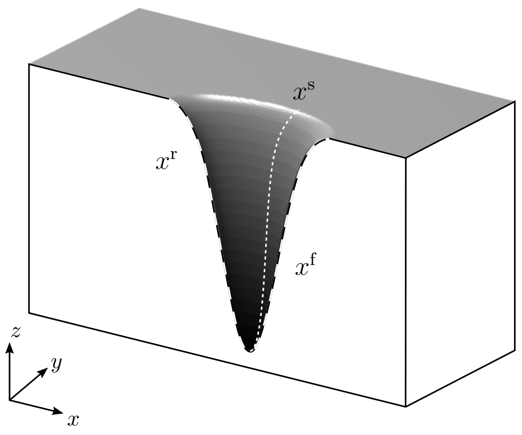


Figure A.1 Keyhole half-geometry with the lines defining its front-back form (long dashes) and its side form (short dashes).

with $K_1(\hat{\boldsymbol{x}})$ as modified Bessel function of second kind and first order. With (A.6) as value for $P'(\hat{r}, \hat{\phi}, z)$ to reach evaporating temperature at the keyhole wall, the heat flow at the keyhole wall is

$$q_v(\hat{r}, \hat{\phi}, z) = (u_{\Gamma_1} - u_a)\lambda \mathrm{Pe}' \left(\cos\hat{\phi} + \frac{K_1(\mathrm{Pe}'\hat{r})}{K_0(\mathrm{Pe}'\hat{r})} \right). \tag{A.9}$$

Remark A.1. To further simplify the notation, we introduce the transformation function $\psi(\boldsymbol{x}) := \psi(\boldsymbol{x}; \boldsymbol{x}^{\mathrm{HS}}(z)) = \psi(r, \phi, z; \boldsymbol{x}^{\mathrm{HS}}(z))$ which converts coordinates (r, ϕ, z) in the standard coordinate system to coordinates $(\hat{r}, \hat{\phi}, z)$ in the heat source's coordinate system with origin in $\boldsymbol{x}^{\mathrm{HS}}(z)$, $z \in \Omega$.

A.3 Location of the heat source

As pointed out before, equation (A.9) is derived assuming that for every $z \in \Omega$ there is a moving line source with energy P' by which evaporating temperature can be obtained at the keyhole wall in this z-layer. It is important to note that the location of the heat source $\boldsymbol{x}^{\mathrm{HS}}(z)$ depends on $z \in \Omega$ and, especially, does not align with the (fixed) laser position.

Given the coordinates $\boldsymbol{x}_i^{\mathrm{f}} = (x_i^{\mathrm{f}}, 0, z_i) = (r_i^{\mathrm{f}}, \phi_i^{\mathrm{f}}, z_i)$ and $\boldsymbol{x}_i^{\mathrm{r}} = (x_i^{\mathrm{r}}, 0, z_i) = (r_i^{\mathrm{r}}, \phi_i^{\mathrm{r}}, z_i)$ in the laser coordinate system, we can approximate the position $\boldsymbol{x}_i^{\mathrm{HS}}$ using an explicit scheme that is based on the distance between front and rear keyhole wall at layer z_i. These coordinates correspond to the lines depicted in the front, back and side of the keyhole geometry from Figure A.1. More precisely, we have to compute $\boldsymbol{x}_i^{\mathrm{HS}} = (x_i^{\mathrm{HS}}, 0, z_i) = (r_i^{\mathrm{HS}}, \phi_i^{\mathrm{HS}}, z_i)$ s.t. the following equation holds:

$$0 = u_{\Gamma_1} - u_{\Gamma_1} = u(\hat{\boldsymbol{x}}_i^{\mathrm{f}}) - u(\hat{\boldsymbol{x}}_i^{\mathrm{r}}) = u(\boldsymbol{x}_i^{\mathrm{f}} - \boldsymbol{x}_i^{\mathrm{HS}}) - u(\boldsymbol{x}_i^{\mathrm{r}} - \boldsymbol{x}_i^{\mathrm{HS}}) \tag{A.10}$$

which, with (A.4), can be simplified to

$$0 = K_0(\mathrm{Pe}'(r_i^{\mathrm{f}} - r_{i+1}^{\mathrm{HS}}) \exp(-\mathrm{Pe}'(r_i^{\mathrm{f}} - r_i^{\mathrm{HS}})) \quad - K_0(\mathrm{Pe}'(r_i^{\mathrm{r}} - r_i^{\mathrm{HS}})) \exp(\mathrm{Pe}'(r_i^{\mathrm{r}} - r_i^{\mathrm{HS}})). \tag{A.11}$$

The function in equation (A.11) has a singularity only at the origin and can be solved to find r_{i+1}^{HS}. This can be numerically well approximated by a bisection method and, once the heat source's location is found, we can define the transformation function $\psi(\boldsymbol{x})$ and evaluate (A.9) not in (r, ϕ, z), but in $(\hat{r}, \hat{\phi}, z)$ for the next z-layer.

A.4 Computation scheme for the keyhole geometry

For computing the keyhole geometry, the local heat losses at the keyhole wall are compared to the locally absorbed intensity yielding the relation

$$\tan\left(\theta(r,\phi,z)\right) = \frac{q_v(\psi(r,\phi,z))}{\alpha_{\mathrm{fr}} I(r,\phi,z)} = \frac{q_v(\psi(r,\phi,z))}{I_a(r,\phi,z)} = \frac{q_v(\hat{r},\hat{\phi},z))}{I_a(r,\phi,z)} \tag{A.12}$$

for the local inclination angle θ, with α_{fr} denoting the Fresnel absorption. Please note that we neglect the dependency of the absorption rate on the angle of incidence but use a constant (mean) Fresnel absorption coefficient.

While (A.12) is an implicit equation and, in principle, allows for a point-wise computation of the entire keyhole geometry, we only use it to determine the front, rear and side wall. Moreover, we consider the equation explicitly by approximating the values on the right-hand-side using the precisely computed value of θ. With this, the explicit point-by-point scheme illustrated in Figure A.2 is as follows:

1. Given the keyhole wall points $\boldsymbol{x}_i^{\mathrm{f}}$, $\boldsymbol{x}_i^{\mathrm{r}}$ and $\boldsymbol{x}_i^{\mathrm{s}}$ at the keyhole front, rear and side wall, we first compute the location of the heat source $\boldsymbol{x}_i^{\mathrm{HS}}$ by solving (A.10).

2. With this, we evaluate $q_v(\psi(\boldsymbol{x}_i))$ and $I(\boldsymbol{x}_i)$ to compute $\theta(\boldsymbol{x}_i)$ via relation (A.12) and determine the next keyhole wall points at the front and rear, $\boldsymbol{x}_{i+1}^{\mathrm{f}}$, $\boldsymbol{x}_{i+1}^{\mathrm{r}}$ (in the x-z-plane of the laser's coordinate system) by

$$\begin{aligned} r_{i+1} &= r_i - \Delta z \tan(\theta(\boldsymbol{x}_i)), \\ \phi_{i+1} &= \phi_i = \begin{cases} 0, & \text{at the front wall,} \\ \pi, & \text{at the rear wall,} \end{cases} \\ z_{i+1} &= z_i - \Delta z, \end{aligned} \qquad \text{resp.} \qquad \begin{aligned} x_{i+1} &= x_i - \Delta z \tan(\theta(\boldsymbol{x}_i)), \\ y_{i+1} &= y_i = 0, \\ z_{i+1} &= z_i - \Delta z. \end{aligned} \tag{A.13}$$

3. Finally, the next point at the keyhole side wall, which is symmetrical to the x-z-plane, is computed by

$$\begin{aligned} x_{i+1} &= x_i^{\mathrm{HS}}, \\ y_{i+1} &= y_i - \Delta z \tan(\theta(\boldsymbol{x}_i)), \\ z_{i+1} &= z_i - \Delta z. \end{aligned} \tag{A.14}$$

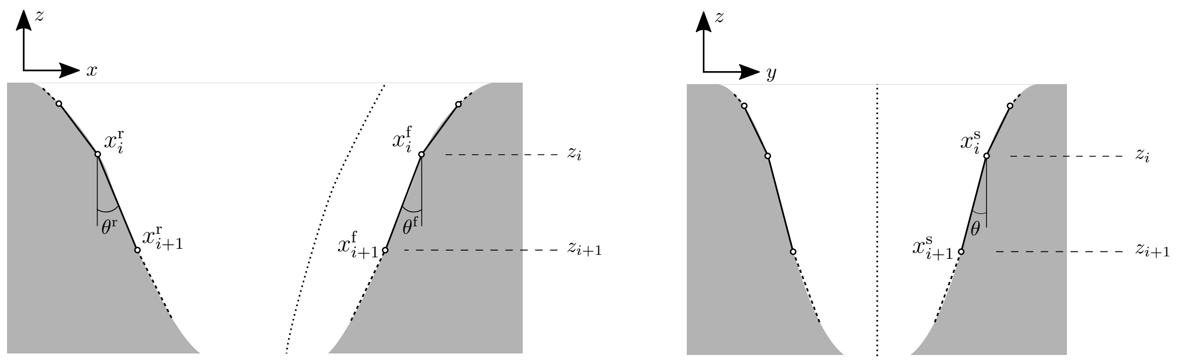


Figure A.2 Discrete iteration to compute the Keyhole shape. *Left:* Side view for computing the front and rear keyhole wall. *Right:* Front view for computing the side view.

here, it is important to note that the x-coordinate of the side wall always aligns with the x-coordinate of the heat source.

The full keyhole geometry is then approximated by defining ellipses for every z_i, using the front and rear wall points as semi-major axes and the side wall points as semi-minor axes in a secondary step. We will elaborate this approach further in Section A.6.

A.5 Multiple reflections

The existence of multi reflections is an essential phenomenon of the laser welding process, as it is responsible for a large amount of energy absorbed into the material pieces. For metallic components, it is known that these reflections can take an absorption factor of around 30% and increase it to values around 80% of the total laser beam energy.

In our formulations, we denoted the intensity in (A.12) by I rather than I_{L} because we have to consider that only a portion of the incident laser beam's intensity I_{L} gets directly absorbed into the material. The remaining intensity $(1-\alpha_{\mathrm{fr}})I_{\mathrm{L}}$ is reflected and (partly) absorbed multiple times within the keyhole causing an overall absorption of up to 80%.

Our computation of the keyhole geometry uses the front, rear and side keyhole walls by a (downwards orientated) point-by-point scheme. Due to this descending computation scheme, we only consider the influence of multi reflections in the negative z-direction, meaning that the impact of a reflection on a previously computed point at the keyhole wall is neglected. Therefore, we can assume that there is no incoming reflection that has to be considered in the computation method for the first $m > 0$ points.

Our approach to consider multi reflections within the computation method for a given point $\boldsymbol{x}_i = (r_i, \phi_i, z_i)$, with $\phi_i \in \{0, \frac{\pi}{2}, \pi\}$ at a keyhole wall, $i \leq m$, consists on performing the following steps:

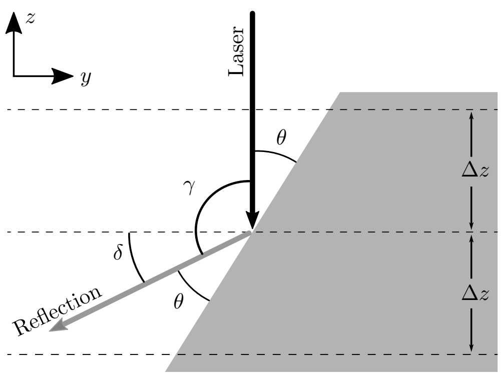


Figure A.3 Scheme for computing the reflection angle for reflected rays.

1. Compute the inclination angle $\theta(\boldsymbol{x}_{i+1}) = \theta(r_i, \phi_i, z_i)$ by taking (only) $I_{\mathrm{L}}(r_i, \phi_i, z_i)$ into account and then define $\boldsymbol{x}_{i+1}$ according to the scheme presented in Section A.4.
2. Determine the reflected intensity $I_r(\boldsymbol{x}_i) = (1 - \alpha_{\mathrm{fr}}) I_{\mathrm{L}}(\boldsymbol{x}_i)$ as well as the reflection angle $\delta(\boldsymbol{x}_i) = \frac{\pi}{2} - 2\theta(\boldsymbol{x}_i)$ relative to the horizontal axis, cf. Figure A.3.
3. Introduce a linear function $f_i(r, z) = z_i - \tan(\delta_i)(r + r_i)$ to compute the point of impact of the reflected intensity at the keyhole wall.

For a point $\boldsymbol{x}_j$ with $j > m$ the previously described method is adapted to consider multi reflections starting with the following steps:

1. Compute a tentative inclination angle $\tilde{\theta}(\boldsymbol{x}_{j+1}) = \tilde{\theta}(r_j, \phi_j, z_j)$ by taking (only) $I_{\mathrm{L}}(r_j, \phi_j, z_j)$ into account and then define a tentative point $\tilde{\boldsymbol{x}}_{j+1}$.
2. Evaluate all functions f_i, $i = 0, \ldots, j-1$, at $(\tilde{r}_j, z_j)$ to find the closest points of impact (r_{i_0}, z_{i_0}) and (r_{i_1}, z_{i_1}) forming an interval which contains $(\tilde{r}_j, z_j)$.
3. Compute the reflected intensity $I_r(\tilde{r}_j, z_j)$ (acting in $-z$-direction) by interpolating the "cosinus parts" of the corresponding intensities I_{i_0} and I_{i_1}
4. Define the total intensity at $(\tilde{r}_j, z_j)$ by $I(r_j, \phi_j, z_j) = I_{\mathrm{L}}(r_j, \phi_j, z_j) + I_r(\tilde{r}_j, \phi_j, z_j)$ and continue with step (1) of the previous scheme using this intensity.

Note that the functions f_i are independent of the angle ϕ_i since a reflection occurs always in normal direction. In our case that means that if an intensity is reflected at the front keyhole wall, it has only an impact of at the rear keyhole wall and vice versa. This is also true for the keyhole side walls. As we only consider these four points, we can model multi reflections without implementing a numerically expensive ray tracing algorithm.

A.6 Level set representation of the keyhole geometry

The discrete points at the front, rear and side keyhole wall computed by (A.12) can be used to construct a level set function whose zero level set represents the whole keyhole geometry. Let $\boldsymbol{x}_i^{\mathrm{f}} = (x_i^{\mathrm{f}}, 0, z_i)$, $\boldsymbol{x}_i^{\mathrm{r}} = (x_i^{\mathrm{r}}, 0, z_i)$ and $\boldsymbol{x}_i^{\mathrm{s}_1} = (x_i^{\mathrm{s}}, y_i^{\mathrm{s}}, z_i) = (x_i^{\mathrm{HS}}, y_i^{\mathrm{s}}, z_i)$ resp. $\boldsymbol{x}_i^{\mathrm{s}_2} = (x_i^{\mathrm{s}}, -y_i^{\mathrm{s}}, z_i) = (x_i^{\mathrm{HS}}, -y_i^{\mathrm{s}}, z_i)$, $i = 1, \dots, M$, be the coordinates of the points at the front, rear and side keyhole wall. For every depth layer z_i, we approximate the keyhole shape by two half-ellipses using the front and rear keyhole wall points as semi-major axes and the side keyhole wall points as semi-minor axes, see Figure A.4.

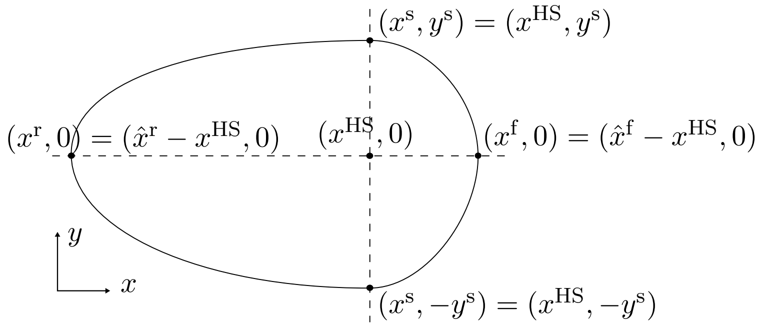

$(x^{\mathrm{s}}, y^{\mathrm{s}}) = (x^{\mathrm{HS}}, y^{\mathrm{s}})$
$(x^{\mathrm{r}}, 0) = (\hat{x}^{\mathrm{r}} - x^{\mathrm{HS}}, 0)$
$(x^{\mathrm{HS}}, 0)$
$(x^{\mathrm{f}}, 0) = (\hat{x}^{\mathrm{f}} - x^{\mathrm{HS}}, 0)$
y
x
$(x^{\mathrm{s}}, -y^{\mathrm{s}}) = (x^{\mathrm{HS}}, -y^{\mathrm{s}})$

Figure A.4 Keyhole wall approximated at a fixed depth and with a heat source location $(x^{\mathrm{HS}}, 0)$ in both coordinate systems.

By doing this, the half-ellipse connecting $\boldsymbol{x}_i^{\mathrm{f}}$, $\boldsymbol{x}_i^{\mathrm{s}_1}$ and $\boldsymbol{x}_i^{\mathrm{s}_2}$ is given by

$$y^2 = (y_i^{\mathrm{s}})^2 - \frac{(y_i^{\mathrm{s}})^2}{(x_i^{\mathrm{f}})^2}(x - x_i^{\mathrm{s}})^2. \tag{A.15}$$

and the half-ellipse connecting $\boldsymbol{x}_i^{\mathrm{r}}$, $\boldsymbol{x}_i^{\mathrm{s}_1}$ and $\boldsymbol{x}_i^{\mathrm{s}_2}$ is given by

$$y^2 = (y_i^{\mathrm{s}})^2 - \frac{(y_i^{\mathrm{s}})^2}{(x_i^{\mathrm{r}})^2}(x - x_i^{\mathrm{s}})^2. \tag{A.16}$$

Based on this equations, we introduce the level set functions

$$\begin{aligned}\varphi_{1|_{z_i}}(\boldsymbol{x}) &= \varphi(x, y, z_i) \\ &= \begin{cases} \sqrt{(y^2 - (y_i^{\mathrm{s}})^2 - \frac{(y_i^{\mathrm{s}})^2}{(x_i^{\mathrm{f}})^2}(x - x_i^{\mathrm{s}})^2)}, & \text{for } x \geq x^{\mathrm{HS}} \\ \sqrt{(y^2 - (y_i^{\mathrm{s}})^2 - \frac{(y_i^{\mathrm{s}})^2}{(x_i^{\mathrm{r}})^2}(x - x_i^{\mathrm{s}})^2)}, & \text{for } x < x^{\mathrm{HS}} \end{cases},\end{aligned} \tag{A.17}$$

which are signed distance functions whose zero level sets $\Gamma_{1|_{z_i}}$ represents the (2D) keyhole wall for each depth layer z_i, $i = 1, \dots, M$ in the x-y plane. In the vertical direction, the different ellipses (or equivalently the level set functions) are linearly interpolated between the z-layers to obtain a continuous function φ_1 whose zero level set Γ_1 defines the 3D keyhole geometry. Thereby, φ_1 has to be extended for $z < z_M$ in an arbitrarily but continuous way. For problem (5.88), the keyhole geometry is considered as constant and moves along the welding line at the given welding speed $\vec{V}_{\mathrm{L}}$. Hence, our sharp interface between keyhole and molten area is given by

$$\Gamma_1(t) = \{\boldsymbol{x} \in \Omega \, : \, \Big(\boldsymbol{x} - (t - t_0)\vec{V}_{\mathrm{L}}\Big) \in \Gamma_1(t_0), \; t \in [t_0, t_f]\}. \tag{A.18}$$

Remark A.2. While in a first step, the discrete keyhole Γ_h is computed using a very small step size Δz for a high precision approximation of the keyhole shape and depth, the level set function φ_1 and the corresponding zero level set Γ_1 are constructed using only a set of keyhole wall coordinates which contains only about 10% of the previously computed points including the first and the last one. This does not represent any remarkable reduction in precision, as the interfaces are in the end linearly interpolated during the XFEM enrichment of elements.

Bibliography

[1] A. Logg, K.-A. Mardal, and G. N. Wells, editors. *Automated solution of differential equations by the finite element method*, volume 84 of *Lecture Notes in Computational Science and Engineering*. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

[2] M. Jahn and T. Klock. A level set toolbox including reinitialization and mass correction algorithms for FEniCS. Technical Report 16-01, ZeTeM, Bremen, 2016.

[3] Lamineries MATTHEY SA. Stahl 1.4301. Technical Report 2013/01, Lamineries MATTHEY SA, Lamineries MATTHEY SA, CH-2520 La Neuveville, 2013.

[4] W.F. Gale and T.C. Totemeier. *Smithells Metals Reference Book*. Elsevier Science, 2003. ISBN 9780080480961.

[5] H. R. Shanks, A. H. Klein, and G. C. Danielson. Thermal properties of armco iron. 38: 2885 – 2892, 07 1967.

[6] F. Vollertsen, D. Biermann, H.N. Hansen, I.S. Jawahir, and K. Kuzman. Size effects in manufacturing of metallic components. *CIRP Annals - Manufacturing Technology*, 58(2): 566 – 587, 2009. ISSN 0007-8506. doi: 10.1016/j.cirp.2009.09.002.

[7] F. Vollertsen and R. Walther. Energy balance in laser based free form heading. *CIRP Annals*, 57:291–294, 2008.

[8] A. Stephen and F. Vollertsen. Influence of the rod diameter on the upset ratio in laser-based free form heading. *Steel Research Int., Special Edition: 10th Int. Conf. on Technology of Plasticity (ICTP)*, pages 220–223, 2011.

[9] E. Bänsch, J. Paul, and A. Schmidt. An ALE finite element method for a coupled Stefan problem and Navier–Stokes equations with free capillary surface. *International Journal for Numerical Methods in Fluids*, 71(10):1282–1296, 2013. ISSN 1097-0363. doi: 10.1002/fld.3711.

[10] M. Jahn, A. Luttmann, and A. Schmidt. A FEM simulation for solid-liquid-solid phase transitions during the production of micro-components. In *Proceedings of 11th International Scientific Conference MMA - Advanced Production Technologies*, 2012.

[11] M. Jahn, H. Brüning, A. Schmidt, and F. Vollertsen. Energy dissipation in laser-based free form heading: a numerical approach. *Production Engineering*, 8(1-2):51–61, 2014. ISSN 0944-6524. doi: 10.1007/s11740-013-0509-8.

[12] H. Brüning, M. Teepe, and F. Vollertsen. Surface roughness and size effect in dendrite arm spacing at preforms of aisi 304 (1.4301) generated by laser rod end melting. *Procedia Engineering*, 81(0):1589 – 1594, 2014. ISSN 1877-7058. doi: http://dx.doi.org/10.1016/j.proeng.2014.10.195. 11th International Conference on Technology of Plasticity, ICTP 2014, 19-24 October 2014, Nagoya Congress Center, Nagoya, Japan.

[13] A. Luttmann. *Modellierung und Simulation von Prozessen mit fest-flüssig Phasenübergang und freiem Kapillarrand.* Dissertation, Universität Bremen, 2018.

[14] M. Jahn A. Barr A. Schmidt A. von Hehl F. Vollertsen M. Kowalschuk, A. Luttmann. Challenges in simulation of welded hybrid joints. In F. Grün W. Eichlseder, editor, *Proceedings of the 3rd Fatigue Symposium Leoben*, pages 85–112, 2012.

[15] L. I. Rubinšteĭn. *The Stefan Problem*, volume 27 of *Translations of Mathematical Monographs.* American Mathematical Society, Rhode Island, 1971.

[16] A. Visintin. *Models of phase transition.* Birkhäuser Boston Inc., 1996.

[17] E. Bänsch. Finite element discretization of the Navier-Stokes equations with a free capillary surface. *Numerische Mathematik*, 88(2):203–235, 2001.

[18] C. Grossmann, H.G. Roos, and M. Stynes. *Numerical Treatment of Partial Differential Equations.* Universitext. Springer Berlin Heidelberg, 2007. ISBN 9783540715849.

[19] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002. doi: 10.1017/CBO9780511791253.

[20] M. Jahn, A. Luttmann, A. Schmidt, and J. Paul. Finite element methods for problems with solid-liquid-solid phase transitions and free melt surface. *PAMM*, 12(1):403–404, 2012. ISSN 1617-7061. doi: 10.1002/pamm.201210190.

[21] M. Jahn and A. Schmidt. Finite element simulation of a material accumulation process including phase transitions and a capillary surface. Technical Report 12-03, ZeTeM, Bremen, 2012.

[22] M. Jahn, A. Luttmann, and A. Schmidt. Finite element simulation for material accumulation and welding processes including a free melt surface. *PAMM*, 13(1):235–236, 2013. ISSN 1617-7061. doi: 10.1002/pamm.201310113.

[23] L. Tan and N. Zabaras. A level set simulation of dendritic solidification of multi-component alloys. *Journal of Computational Physics*, 221(1):9 – 40, 2007. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2006.06.003.

[24] B. Schott. *Stabilized Cut Finite Element Methods for Complex Interface Coupled Flow Problems.* Dissertation, TU München, München, 2017.

[25] S. Zlotnik and P. Díez. Hierarchical x-fem for n-phase flow (n¿2). *Computer Methods in Applied Mechanics and Engineering*, 198(30):2329 – 2338, 2009. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2009.02.025.

[26] T.-P. Fries and T. Belytschko. The extended/generalized finite element method: An overview of the method and its applications. *International Journal for Numerical Methods in Engineering*, 84(3):253–304, 2010. ISSN 1097-0207. doi: 10.1002/nme.2914.

[27] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36(1):9–15, 1971. ISSN 1865-8784. doi: 10.1007/BF02995904.

[28] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, November 1988. ISSN 0021-9991. doi: 10.1016/0021-9991(88)90002-2.

[29] S. Gross and A. Reusken. *Numerical Methods for two-phase Incompressible Flows.* Springer Series in Computational Mathematics. Springer, 2011. ISBN 9783642196867.

[30] E. Oñate and M. Manzán. A general procedure for deriving stabilized space–time finite element methods for advective–diffusive problems. *International Journal for Numerical Methods in Fluids*, 31(1):203–221, 1999. ISSN 1097-0363. doi: 10.1002/(SICI)1097-0363(19990915)31:1⟨203::AID-FLD964⟩3.0.CO;2-Z.

[31] W.E. Schiesser. *The Numerical Method of Lines: Integration of Partial Differential Equations.* Academic Press, 1991. ISBN 9780126241303.

[32] W.E. Schiesser. *Computational Mathematics in Engineering and Applied Science: ODEs, DAEs, and PDEs.* Symbolic & Numeric Computation. Taylor & Francis, 1993. ISBN 9780849373732.

[33] T. E. Tezduyar. Interface-tracking and interface-capturing techniques for finite element computation of moving boundaries and interfaces. *Computer Methods in Applied Mechanics and Engineering*, 195(23):2983 – 3000, 2006. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2004.09.018. Incompressible CFD.

[34] C.W Hirt, A.A Amsden, and J.L Cook. An arbitrary lagrangian-eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14(3):227 – 253, 1974. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(74)90051-5.

[35] R. H. Nochetto, K. G. Siebert, and A. Veeser. Theory of adaptive finite element methods: An introduction. In R. DeVore and A. Kunoth, editors, *Multiscale, Nonlinear and Adaptive Approximation*, pages 409–542, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[36] I. Babuška. The finite element method with penalty. *Mathematics of Computation*, 27 (122):221–228, 1973. ISSN 00255718, 10886842.

[37] I. Babuška and M. Zlámal. Nonconforming elements in the finite element method with penalty. *SIAM Journal on Numerical Analysis*, 10(5):863–875, 1973. doi: 10.1137/0710071.

[38] J. W. Barrett and C. M. Elliott. Finite element approximation of the dirichlet problem using the boundary penalty method. *Numerische Mathematik*, 49(4):343–366, Jul 1986.

[39] J. M. Melenk and I. Babuška. The partition of unity finite element method: Basic theory and applications. *Computer Methods in Applied Mechanics and Engineering*, 139(1-4): 289–314, 12 1996. ISSN 0374-2830.

[40] I. Babuška and J. M. Melenk. The partition of unity method. *International Journal for Numerical Methods in Engineering*, 40(4):727–758, 1997. ISSN 1097-0207. doi: 10.1002/(SICI)1097-0207(19970228)40:4⟨727::AID-NME86⟩3.0.CO;2-N.

[41] T. Strouboulis, K. Copps, and I. Babuška. The generalized finite element method: an example of its implementation and illustration of its performance. *International Journal for Numerical Methods in Engineering*, 47(8):1401–1417, 2000. ISSN 1097-0207. doi: 10.1002/(SICI)1097-0207(20000320)47:8⟨1401::AID-NME835⟩3.0.CO;2-8.

[42] C. A. Duarte, I. Babuška, and J. T. Oden. Generalized finite element methods for three-dimensional structural mechanics problems. *Computers & Structures*, 77(2):215 – 232, 2000. ISSN 0045-7949.

[43] J. E. Dolbow. *An extended finite element method with discontinuous enrichment for applied mechanics*. Northwestern University, 1999.

[44] N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46(1):131–150, 1999. ISSN 1097-0207. doi: 10.1002/(SICI)1097-0207(19990910)46:1⟨131::AID-NME726⟩3.0.CO;2-J.

[45] T. Belytschko, R. Gracie, and G. Ventura. A review of extended/generalized finite element methods for material modeling. *Modelling and Simulation in Materials Science and Engineering*, 17(4):043001, 2009.

[46] J. Chessa, P. Smolinski, and T. Belytschko. The extended finite element method (XFEM) for solidification problems. *International Journal for Numerical Methods in Engineering*, 53(8):1959–1977, 2002. ISSN 1097-0207. doi: 10.1002/nme.386.

[47] R. Merle and J. Dolbow. Solving thermal and phase change problems with the extended finite element method. *Computational Mechanics*, 28(5):339–350, 2002. ISSN 1432-0924. doi: 10.1007/s00466-002-0298-y.

[48] Y. Abdelaziz and A. Hamouine. A survey of the extended finite element. *Computers & Structures*, 86(11):1141 – 1151, 2008. ISSN 0045-7949. doi: https://doi.org/10.1016/j.compstruc.2007.11.001.

[49] A. Hansbo and P. Hansbo. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Computer Methods in Applied Mechanics and Engineering*, 191(47–48):5537 – 5552, 2002. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/S0045-7825(02)00524-8.

[50] C. Lehrenfeld. *On a Space-Time Extended Finite Element Method for the Solution of a Class of Two-Phase Mass Transport Problems*. Dissertation, RWTH Aachen, February 2015.

[51] N. Moes, M. Cloirec, P. Cartraud, and J.-F. Remacle. A computational approach to handle complex microstructure geometries. *Computer Methods in Applied Mechanics and Engineering*, 192:3163–3177, 07 2003.

[52] E. Burman, S. Claus, P. Hansbo, M. G. Larson, and A. Massing. Cutfem: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, 104(7):472–501, 2015. ISSN 1097-0207. doi: 10.1002/nme.4823.

[53] K. W. Cheng and T.-P. Fries. Higher-order XFEM for curved strong and weak discontinuities. *International Journal for Numerical Methods in Engineering*, 82(5):564–590, 2010. ISSN 1097-0207. doi: 10.1002/nme.2768.

[54] M. Nikbakht. *Automated solution of partial differential equations with discontinuities using the Partition of Unity method*. PhD thesis, TU Delft, 2012.

[55] N. Moës, E. Béchet, and M. Tourbier. Imposing dirichlet boundary conditions in the extended finite element method. *International Journal for Numerical Methods in Engineering*, 67(12):1641–1669, 2006. ISSN 1097-0207. doi: 10.1002/nme.1675.

[56] J. E. Dolbow and H. Ji. On strategies for enforcing interfacial constraints and evaluating jump conditions with the extended finite element method. *International Journal for Numerical Methods in Engineering*, 61(14):2508–2535, 2004.

[57] I. Babuška. The finite element method with lagrangian multipliers. *Numerische Mathematik*, 20(3):179–192, Jun 1973. ISSN 0945-3245. doi: 10.1007/BF01436561.

[58] F. Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 8(R2):129–151, 1974.

[59] D. Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie.* Masterclass. Springer Berlin Heidelberg, 2013. ISBN 9783662072332.

[60] E. Burman. Ghost penalty. *Comptes Rendus Mathematique*, 348(21):1217 – 1220, 2010. ISSN 1631-073X. doi: https://doi.org/10.1016/j.crma.2010.10.006.

[61] R. Bank and R. Santos. Analysis of some moving space-time finite element methods. *SIAM Journal on Numerical Analysis*, 30(1):1–18, 1993. doi: 10.1137/0730001.

[62] T.-P. Fries and A. Zilian. On time integration in the XFEM. *International Journal for Numerical Methods in Engineering*, 79(1):69–93, 2009. ISSN 1097-0207. doi: 10.1002/nme.2558.

[63] R.A. Adams and J.J.F. Fournier. *Sobolev Spaces.* Pure and Applied Mathematics. Elsevier Science, 2003. ISBN 9780080541297.

[64] B. Merriman, J. K. Bence, and S. J. Osher. Motion of multiple junctions: A level set approach. *Journal of Computational Physics*, 112(2):334 – 363, 1994. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1994.1105.

[65] H-K Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *Journal of Computational Physics*, 127(1):179 – 195, 1996. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1996.0167.

[66] S. J. Ruuth. A diffusion-generated approach to multiphase motion. *Journal of Computational Physics*, 145(1):166 – 192, 1998. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1998.6028.

[67] D. P. Starinshak, S. Karni, and P. L. Roe. A new level set model for multimaterial flows. *Journal of Computational Physics*, 262:1 – 16, 2014. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2013.12.036.

[68] T.P. Fries and D. Schöllhammer. Higher-order meshing of implicit geometries, part ii: Approximations on manifolds. *Computer Methods in Applied Mechanics and Engineering*, 326:270 – 297, 2017. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2017.07.037.

[69] P. Hansbo. Nitsche's method for interface problems in computa-tional mechanics. *GAMM-Mitteilungen*, 28(2):183–206, 2005. ISSN 1522-2608. doi: 10.1002/gamm.201490018.

[70] E. Burman. A penalty-free nonsymmetric nitsche-type method for the weak imposition of boundary conditions. *SIAM Journal on Numerical Analysis*, 50(4):1959–1981, 2012. doi: 10.1137/10081784X.

[71] E. Burman and P. Hansbo. Fictitious domain finite element methods using cut elements: Ii. a stabilized nitsche method. *Applied Numerical Mathematics*, 62(4):328 – 341, 2012. ISSN 0168-9274. doi: https://doi.org/10.1016/j.apnum.2011.01.008. Third Chilean Workshop on Numerical Analysis of Partial Differential Equations (WONAPDE 2010).

[72] A. N. Brooks and T. J.R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equation. *Computer Methods in Applied Mechanics and Engineering*, 32(1):199 – 259, 1982. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(82)90071-8.

[73] F. Hecht. New development in FreeFem++. *J. Numer. Math.*, 20(3-4):251–265, 2012. ISSN 1570-2820.

[74] COMSOL. *Multiphysics Reference Guide for COMSOL 5.3*, 2017.

[75] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Trans. Math. Softw.*, 40(2):9:1–9:37, March 2014. ISSN 0098-3500. doi: 10.1145/2566630.

[76] M. S. Alnæs. UFL: a finite element form language. In A. Logg, K.-A. Mardal, and G. N. Wells, editors, *Automated solution of differential equations by the finite element method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 17, pages 299–334. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

[77] R. C. Kirby and A. Logg. A compiler for variational forms. *ACM Trans. Math. Softw.*, 32(3):417–444, September 2006. ISSN 0098-3500. doi: 10.1145/1163641.1163644.

[78] K. B. Ølgaard and G. N. Wells. Optimizations for quadrature representations of finite element tensors through automated code generation. *ACM Trans. Math. Softw.*, 37(1): 8:1–8:23, January 2010. ISSN 0098-3500. doi: 10.1145/1644001.1644009.

[79] A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. FFC: the FEniCS form compiler. In A. Logg, K.-A. Mardal, and G. N. Wells, editors, *Automated solution of differential equations by the finite element method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 11, pages 223–234. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

[80] R. C. Kirby. Algorithm 839: Fiat, a new paradigm for computing finite element basis functions. *ACM Trans. Math. Softw.*, 30(4):502–516, December 2004. ISSN 0098-3500. doi: 10.1145/1039813.1039820.

[81] R. C. Kirby. FIAT: numerical construction of finite element basis functions. In A. Logg, K.-A. Mardal, and G. N. Wells, editors, *Automated solution of differential equations by the finite element method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 13, pages 243–252. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

[82] M. S. Alnæs, A. Logg, K.-A. Mardal, O. Skavhaug, and H. P. Langtangen. Unified framework for finite element assembly. *International Journal of Computational Science and Engineering*, 4(4):231–244, 2009. doi: 10.1504/2009.029160.

[83] M. S. Alnæs, A. Logg, and K.-A. Mardal. UFC: a finite element code generation interface. In A. Logg, K.-A. Mardal, and G. N. Wells, editors, *Automated solution of differential equations by the finite element method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 16, pages 279–298. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

[84] A. Logg and G. N. Wells. DOLFIN: Automated finite element computing. *ACM Trans Math Software*, 37(2):20:1–20:28, 2010.

[85] A. Logg, G. N. Wells, and J. Hake. DOLFIN: a C++/Python finite element library. In A. Logg, K.-A. Mardal, and G. N. Wells, editors, *Automated solution of differential equations by the finite element method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 10, pages 171–222. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

[86] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.8, Argonne National Laboratory, 2017.

[87] W. Schroeder, Ken M., and B. Lorensen. *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition*. Kitware, 4th edition, December 2006. ISBN 193093419X.

[88] M. Nikbakht and G. N. Wells. Automated modelling of evolving discontinuities. *Algorithms*, 2:1008–1030, 2009. doi: 10.3390/a2031008.

[89] M. Nikbakht and G. N.Wells. Modeling evolving discontinuities. In A. Logg, K.-A. Mardal, and G. N. Wells, editors, *Automated solution of differential equations by the finite element method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 30, pages 573–586. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

[90] U. Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., USA, 2015. ISBN 1930934300, 9781930934306.

[91] T. Williams, C. Kelley, and many others. Gnuplot 5.0: An interactive plotting program. http://www.gnuplot.info, 2015.

[92] C. M. Elliot. On the finite element approximation of an elliptic variational inequality arising from an implicit time discretization of the Stefan problem. *IMA Journal of Numerical analysis*, 1:115–125, 1981.

[93] T. Klock. *Numerical solution of the two-phase Stefan problem with XFEM and level set methods*. Master thesis, Universität Bremen, 2016.

[94] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155:410–438, November 1999. doi: 10.1006/jcph.1999.6345.

[95] R. F. Ausas, E. A. Dari, and G. C. Buscaglia. A geometric mass-preserving redistancing scheme for the level set function. *International Journal for Numerical Methods in Fluids*, 65(8):989–1010, 2011. ISSN 1097-0363. doi: 10.1002/fld.2227.

[96] H.G. Roos, M. Stynes, and L. Tobiska. *Robust numerical methods for singularly perturbed differential equations: Convection-diffusion-reaction and flow problems.* Springer Series in Computational Mathematics. Springer, 2008. ISBN 9783540344674.

[97] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Eng.*, pages 199–259, 1990. ISSN 0045-7825.

[98] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[99] A. Reusken and E. Loch. *On the Accuracy of the Level Set SUPG Method for Approximating Interfaces.* Bericht. Inst. für Geometrie und Praktische Mathematik, 2011.

[100] N. Anderson and Å. Björck. A new high order method of regula falsi type for computing a root of an equation. *BIT Numerical Mathematics*, 13(3):253–264, 1973.

[101] M. Bernauer. Motion planning for the two-phase Stefan problem in level set formulation. *Dissertation*, 2010.

[102] M. Jahn and A. Luttmann. Solving the Stefan problem with prescribed interface using an XFEM toolbox for FEniCS. Technical Report 16-03, ZeTeM, Bremen, 2016.

[103] M. Jahn and T. Klock. Numerical solution of the Stefan problem in level set formulation with the eXtended finite element method in FEniCS. Technical Report 17-01, ZeTeM, Bremen, 2017.

[104] M. Jahn, T. Klock, and A. Luttmann. Levelset methods (and XFEM) in FEniCS, 2015. FEniCS'15 Workshop at Imperial College London.

[105] M. Jahn, A. Luttmann, and T. Klock. An XFEM toolbox for FEniCS, 2016. FEniCS'16 Workshop at Simula Research Laboratory.

[106] M. Jahn. miXFEM - an XFEM toolbox to tackle multiphysics problems with FEniCS, 2018. FEniCS'18 Workshop at Oxford University.

[107] G. Dziuk. *Theorie und Numerik partieller Differentialgleichungen.* De-Gruyter-Studium. De Gruyter, 2010. ISBN 9783110148435.

[108] M. Jahn, A. Schmidt, and E. Bänsch. 3D finite element simulation of a material accumulation process including phase transistions and a capillary surface, 01.01.2012.

[109] K. Chongbunwatana. Simulation of vapour keyhole and weld pool dynamics during laser beam welding. *Production Engineering*, 8(4):499–511, Aug 2014. ISSN 1863-7353. doi: 10.1007/s11740-014-0555-x.

[110] A. Kaplan. A model of deep penetration laser welding based on calculation of the keyhole profile. *Journal of Physics D Applied Physics*, 27:1805–1814, September 1994. doi: 10.1088/0022-3727/27/9/002.

[111] D. Rosenthal. The theory of moving source of heat and its application to metal treatments. *ASME Transactions*, 48:848–866, 1946.

[112] H.S. Carslaw and J.C. Jaeger. *Conduction of heat in solids.* Oxford science publications. Clarendon Press, 1959.

[113] J. Montalvo-Urquizo, Z. Akbay, and A. Schmidt. Adaptive finite element models applied to the laser welding problem. *Computational Materials Science*, 46(1):245–254, 2009. ISSN 0927-0256. doi: http://dx.doi.org/10.1016/j.commatsci.2009.02.037.

[114] S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska. Quantitative benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids*, 60(11):1259–1288. doi: 10.1002/fld.1934.

[115] M. Griebel S. Groß M. Klitz A. Rüttgers J. Adelsberger, P. Esser. 3d incompressible two-phase flow benchmark computations for rising droplets. Technical Report 393, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Aachen, 2014.

[116] R. Temam. *Navier-Stokes Equations: Theory and Numerical Analysis.* Studies in mathematics and its applications. North-Holland, 1979. ISBN 9780444875594.

[117] E. Bänsch. *Numerical methods for the instationary Navier-Stokes equations with a free capillary surface.* Habilitationsschrift, Universität Freiburg, 1998.

[118] G.K. Batchelor. *An Introduction to Fluid Dynamics.* Cambridge Mathematical Library. Cambridge University Press, 2000. ISBN 9780521663960.