Level set methods for high-order unfitted discontinuous Galerkin schemes

Vom Fachbereich Maschinenbau
an der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades
eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

D i s s e r t a t i o n

von

Thomas Utz
aus Augsburg

Berichterstatter:	Prof. Dr.-Ing. M. Oberlack
Mitberichterstatter:	Prof. Dr. rer. nat. M. Schäfer
Tag der Einreichung:	16.03.2018
Tag der mündlichen Prüfung:	20.06.2018

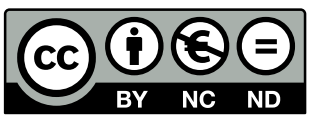
Darmstadt, 2018
D17

Erklärung

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 16. März 2018

Thomas Utz



Abstract

This work presents three algorithms for the level set modeling of phase boundaries. The application of these algorithms are high-order extended discontinuous Galerkin methods for multiphase flow simulations.

The first algorithm is a reinitialization method, which is based on solving an elliptic partial differential equation. The algorithm is high order accurate in global norms. This reinitialization technique can be applied to arbitrary problems by using a first-order solver as preconditioning.

The second algorithm is a high-order accurate solver for extending quantities from the interface into the domain. This is especially helpful for using a so called extension velocity for cases, in which the velocity of the interface is not given by a global field. Like the reinitialization algorithm, the method relies on solving an elliptic partial differential equation. Based on the underlying level-set, this problem might be ill-posed. An extension by an artificial viscosity allows stable solutions even for these cases.

The third algorithm is a coupling of these two algorithms to an upwind discretization of the level set transport equation using an implicit time stepping scheme. For sufficiently smooth problems, this coupling gives high order accuracy as well. Last, this coupled scheme is applied to the simulation of a rising bubble using an unfitted discontinuous Galerkin scheme, which shows good agreement with reference solutions from literature.

Zusammenfassung

Diese Arbeit beschreibt drei Algorithmen für die Modellierung von Phasengrenzen mittels der Level-Set Methode. Die Anwendung dafür ist die Simulation von Mehrphasenströmungen mittels einer discontinuous Galerkin Methode mit hoher Ansatzordnung und Erweiterung für geschnittene Zellen.

Der erste Algorithmus behandelt das sogenannte Reinitialisierungsproblem mittels der Lösung eines elliptischen Problems. Dieser Algorithmus zeigt Konvergenzverhalten hoher Ordnung in globalen Normen. Durch den Einsatz eines Präkonditionierers, basierend auf einem Verfahren erster Ordnung kann die Methode auf beliebige Problemstellungen angewendet werden.

Der zweite Algorithmus behandelt die Ausbreitung von Größen von der Phasengrenze in das Rechengebiet hinein. Diese Fragestellung ist besonders im Fall der sogenannten Extension Velocity relevant, bei der der Geschwindigkeitswert an der Phasengrenze nicht als globales Feld gegeben ist, das im gesamten Rechengebiet definiert ist. Wie der Algorithmus für Reinitialisierung basiert diese Methode auf der Lösung einer elliptischen Differentialgleichung. In Abhängigkeit des Level-Set Felds kann das Problem schlecht gestellt sein. Eine Erweiterung um eine sogenannte künstliche Viskosität stabilisiert den Löser auch für diese Fälle.

Der Dritte Algorithmus koppelt diese beiden Algorithmen mit einem Upwind-Fluss für die Level Set Transportgleichung unter Verwendung einer impliziten Zeitdiskretisierung. Für glatte Probleme zeigt dieser Algorithmus ebenfalls eine hohe Fehlerordnung. Abschließend wird dieses gekoppelte Schema auf die Simulation einer aufsteigenden Blase mittels einer nicht randangepassten discontinuous Galerkin Methode angewendet. Die Ergebnisse zeigen gute Übereinstimmung mit Referenzlösungen aus der Literatur.

Acknowledgments

This work was supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at Technische Universität Darmstadt and by the German Science Foundation (DFG) within the Priority Program (SPP) 1506 "Transport Processes at Fluidic Interfaces".

I would like to thank my supervisor Prof. Dr. Martin Oberlack for the opportunity to do this PhD thesis at the Chair of Fluid Dynamics and for his encouragement during the past four years. In addition I would like to thank Prof. Dr. Michael Schäfer for being my second supervisor.

I am deeply grateful to Dr. Florian Kummer and Dr. Björn Müller for their patience and never ending support for the BoSSS-code and countless other topics concerning my work. With both of them I had very fruitful and eye-opening discussions throughout the whole course of this project. I would like to thank Lauritz Beck for his implementation of the Fast Marching Algorithm and Dr. Florian Kummer and Martin Smuda for the effort they put into the multiphase Navier-Stokes solver. In addition I am thankful to all current and past members of the BoSSS team for the countless discussions which gave me a lot of input for my work. In particular I would like to thank (in alphabetical order) Markus Geisenhofer, Benedikt Klein, Stephan Krämer-Eis, Dennis Krause, Martin Smuda, and Prof. Dr.-Ing. habil. Yongqi Wang for sharing their ideas.

I am very thankful to all colleagues at the Chair of Fluid Dynamics, where I always enjoyed the nice working atmosphere and being a member of the team. I would like to thank the team who organised the mechanics 1 lecture with me, (in alphabetical order) Pascal Cabos, Markus Geisenhofer, Stefanie Kraheberger, and Patrick Steinberg. In particular I would like to thank Markus Geisenhofer for being a great office mate and sharing his ideas even late in the evening.

Last of all I would like to thank my whole family and all my friends for the encouragement and ongoing support. In particular I am deeply grateful to my fiancée Laura Hofmann, who always supported me throughout this strenuous endeavour.

Contents

Nomenclature

α	Penalty parameter at the interface $\mathfrak{I}$
∂	Boundary of a spatial object
δ_{ij}	Kronecker delta
ϵ	Artificial viscosity for the stabilization of the extension equation
η	Penalty parameter at the cell edge e
κ	Curvature
λ	Convergence Criterion for the Motion Algorithm
μ	viscosity
Ω	Spatial Domain
Ω_h	Numerical approximation of the spatial domain Ω
φ	Level set
ϕ	Basis function
ψ	Potential function for reinitialization
ρ	density
$\tau_{\mathfrak{I}}$	Tangential plane on the interface $\mathfrak{I}$
$\mathfrak{A}$	Part of the domain occupied by the fluid marked A
$\mathfrak{B}$	Part of the domain occupied by the fluid marked B
D	Value at a dirichlet boundary
e	Edge of a computational cell K
E_c	Error along the interface $\mathfrak{I}$
E_{L^n}	Error in the n-Norm of a domain
E_{SD}	Error measure for the deviation from signed distance
f	Flux function
$\tilde{f}$	Numerical flux
h	Characteristic size of the computational cell K
$\mathfrak{I}$	The interface separating $\mathfrak{A}$ and $\mathfrak{B}$
K	Computational cell
k	Polynomial degree of a DG field

$\mathfrak{K}$	Mesh/Grid containing the cells K
$\mathfrak{K}_{Cut}$	Cut cell grid
$\vec{n}$	Normal vector
N	Value at a Neumann boundary
$\mathbf{P}$	Projection
$\mathbb{P}$	Space of polynomial functions
p	Pressure
$\mathbb{R}$	Space of real numbers
$\vec{t}$	Tangential vector
t	Time
u	General unknown variable
$\vec{u}$	Velocity
v	Test function
$\mathbb{V}_{\mathrm{DG}}$	Space of DG functions
$\mathbb{V}_{\mathrm{DG}}^{X}$	Space of extended DG functions
$\vec{x}_c$	The closest point on the interface
$\mathbb{Z}$	Space of integers

Abbreviations

BoSSS Bounded Support Spectral Solver

BDF backward differentiation formula

DG discontinuous Galerkin method

EOC experimental order of convergence

FEM finite element method

FVM finite volume method

HMF hierarchical moment-fitting

ODE ordinary differential equation

PDE partial differential equation

SIP symmetric interior penalty

XFEM extended finite element method

XDG extended discontinuous Galerkin method

1 Introduction

Increasing computational power has led to the widespread acceptance of numerical methods to solve engineering problems in fluid mechanics. The well established and widely accepted method in research and development are so called low-order methods which suffer from one major drawback: Their computational cost increases faster than their accuracy, especially for three dimensional simulations and time dependent problems.
To circumvent this, one must resort to so called higher-order methods, which deliver superior accuracy, and their increase in accuracy scales larger than linearly with the computational effort. These methods however are still a topic of active research, since many well-established algorithms for low order methods do not deliver the required accuracy and therefore need to be replaced. Here lies the topic of this thesis: The discontinuous Galerkin method (DG) method is one of the most popular high-order methods, since it allows the discretization of arbitrary geometries with unstructured meshes and easy parallelization. The goal is to apply this method to the simulation of multiphase flows, such as e.g. the motion of an air bubble in water. The numerical methods for such flows are still a matter of active research even in the context of low-order methods. These simulations additionally require special algorithms to model not only the behavior of the fluid phases, but also the motion of the interface separating them, for which this work uses a method called level set.

The level set method relies on employing a smooth function, which is defined on the whole computational domain, just like the pressure or velocity field. This so called level set function defines the interface between two phases by the positions where it is zero. In contrast to other popular methods, this allows the simulation of flows with changing topologies, such as the merge or split-up of droplets. In addition, the method can be combined with so called sharp interface models. These models directly implement jump conditions at the interface and thus allow high-order accuracy for these kind of simulations. However, when applying a level set method, one needs two additional algorithms:

First, especially when dealing with topological changes, the level-set function might become very steep or flat, which might lead to increased errors in tracking the interface. In addition, many applications require the gradient of the level set function to be a good approximation to the normal of the fluid interface. To reconstruct both properties, an algorithm called reinitialization is needed.

Second, in many cases the interest in the simulations lies in quantities, which are only defined at the interface itself. When these quantities interact with a globally defined field, one needs to extend the surface quantity into the domain. The example most relevant for multiphase flows is the extension of the interface velocity into the domain,

which moves the globally defined level-set function. For both algorithms there are well established choices available from literature, but these approaches are not generally suited for high-order discontinuous Galerkin methods. The main contribution of this work are two ideas for reinitialization and extension, which are combined into a motion algorithm for level set functions. This motion algorithm is then applied to the multiphase flow simulations of a rising bubble.

1.1 Goals and outline of this work

This thesis presents a method to model the phase boundary in multi physics simulations. This so called level set method requires additional algorithms. The goal of this work is to present and evaluate these algorithms individually, together and in combination with multiphase flow simulations. Chapter 2 introduces the continuum mechanical models describing such multiphase flows. The numerical method used in this thesis to solve arising partial differential equations (PDEs) from these models is the discontinuous Galerkin method (DG). Chapter 3 gives a brief introduction to the method and to the extensions needed when moving from single phase to multiphase problems. This so called level set method requires two additional algorithms, which are the core part of this thesis:

The first algorithm, called reinitialization, ensures smoothness of the level-set function. Chapter 4 discusses a novel procedure for level set reinitialization in the context of unfitted DG methods. Main parts of this chapter are based on the publication Utz et al. (2017b) by the author of this thesis.

The second algorithm is needed, for quantities, which are only defined at the interface but interact with globally defined fields. Prime example is the velocity at the interface, which moves the globally defined level set function. Chapter 5 discusses a novel procedure for extending quantities from a predefined interface into a domain. The author's second publication, Utz and Kummer (2017), is the basis for this chapter. Chapter 6 briefly discusses a method for surface equations, which has some similarities with the method presented in chapter 5.

Chapter 7 describes the combination of the two algorithms from chapters 4 and 5 with an advection equation to calculate the motion of a surface. In the last chapter 8, this motion algorithm is coupled to a multiphase Navier-Stokes simulation, where we compare the simulation of a rising bubble to the results obtained by Heimann et al. (2013) and the reference publication by Hysing et al. (2009).

1.2 The framework BoSSS

All numerical schemes presented are incorporated in the discontinuous Galerkin framework BoSSS. This framework is a collection of multiple solvers, which are continuously developed at the institute of fluid dynamics at TU Darmstadt. The main strengths of

BoSSS is its object oriented programming approach, since it is written in the programming language C#. Its modularized approach allows quick development of various kind of solvers, such as incompressible flow and low Mach number flow solver by Klein (2015), multiphase flow solver based on an extended DG approach by Kummer (2012, 2016); Utz and Kummer (2017) and Utz et al. (2017b), compressible flow solver incorporating the immersed boundary method and local time stepping by Müller (2014); Müller et al. (2013) and Müller et al. (2017) or even solver for particle-laden flows by Krause and Kummer (2017). Due to the platform-independence of C# and the incorporation of MPI-parallelism, the BoSSS code runs on Windows, MacOS and Linux alike from personal computers up to high-performance supercomputers such as the Lichtenberg-Cluster at TU Darmstadt. Due to a continuous-integration based development work flow, the latest developments in BoSSS are published on the website `github.com/fdydarmstadt/bosss`, where it is available to the general public.

2 Mathematical model

This chapter introduces the continuum-mechanical models for viscous flows in the vicinity of phase boundaries and the modeling of the interface itself using a level set method. In addition it introduces the partial differential equations which are the basis for the level set algorithms shown here. Central parts of this chapter are based on the two peer-reviewed publications Utz et al. (2017b); Utz and Kummer (2017) and the book chapter Utz et al. (2017a), which have all been published by the author of this thesis.

2.1 Multiphase flows

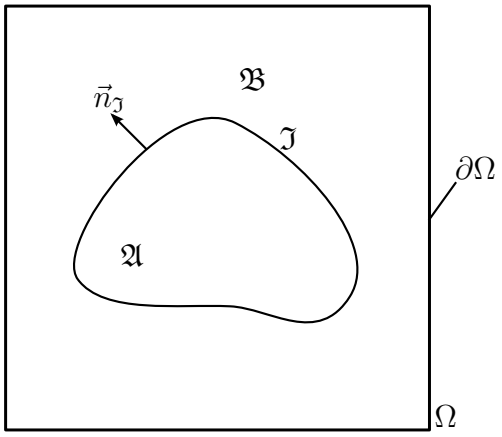


Figure 2.1: Computational Domain with two phases

Consider a domain Ω, such as depicted in figure 2.1, in which an incompressible multiphase flow problem is to be solved, i.e. the unknown quantities are the velocity field $\vec{u}$ and the pressure field p and the position of the interface $\mathfrak{I}$. For the sake of simplicity, the problems considered here are limited to two phases $\mathfrak{A}$ and $\mathfrak{B}$ with piece-wise constant density ρ and dynamic viscosity μ:

$$\rho(\vec{x}) = \begin{cases} \rho_\mathfrak{A} & \text{for } \vec{x} \in \mathfrak{A} \\ \rho_\mathfrak{B} & \text{for } \vec{x} \in \mathfrak{B} \end{cases} \quad \text{and} \quad \mu(\vec{x}) = \begin{cases} \mu_\mathfrak{A} & \text{for } \vec{x} \in \mathfrak{A} \\ \mu_\mathfrak{B} & \text{for } \vec{x} \in \mathfrak{B} \end{cases} . \tag{2.1}$$

The two phases are separated by an interface $\mathfrak{I}$ with the normal on the interface $\vec{n}_\mathfrak{I}$ pointing from $\mathfrak{A}$ to $\mathfrak{B}$. In both phases $\Omega \setminus \mathfrak{I}$, the flow is modelled by the incompressible Navier-Stokes equations, which are the conservation of momentum

$$\partial_t \vec{u} + \nabla \cdot (\rho \vec{u} \otimes \vec{u}) = -\nabla p + \nabla \cdot \mu(\nabla \vec{u} + (\nabla \vec{u})^T) - \vec{F} \quad \text{in } \Omega, \tag{2.2a}$$

and the conservation of mass or continuity equation

$$\nabla \cdot \vec{u} = 0 \quad \text{in } \Omega. \tag{2.2b}$$

At the interface $\mathfrak{I}$, these two equation contract to the jump condition in the stresses and the jump condition in the velocity field, see for example Wang and Oberlack (2011). This jump operator of a vector quantity $\vec{y}$ is defined as

$$[\![\vec{y}]\!] = (\vec{y_{\mathfrak{A}}} - \vec{y_{\mathfrak{B}}})\,. \tag{2.3}$$

Then the jump condition in the stresses is written as

$$\left[\!\left[p\vec{n}_{\mathfrak{I}} - \mu(\nabla\vec{u} + (\nabla\vec{u})^T)\vec{n}_{\mathfrak{I}}\right]\!\right] = \sigma\kappa\vec{n}_{\mathfrak{I}} \quad \text{on } \mathfrak{I}, \tag{2.4a}$$

where σ denotes the surface tension, κ the mean curvature of $\mathfrak{I}$ and $\vec{n}_{\mathfrak{I}}$ the normal of the interface $\mathfrak{I}$. The jump condition in the velocity field is

$$[\![\vec{u}]\!] = 0 \quad \text{on } \mathfrak{I}. \tag{2.4b}$$

Furthermore, one has to specify boundary conditions

$$\vec{u} = \vec{u}_{\mathrm{D}} \quad \text{on } \partial\Omega_{\mathrm{D}} \tag{2.5}$$

$$\mu\left(\nabla\vec{u} + \nabla\vec{u}^T\right)\cdot\vec{n}_{\Omega} - p\vec{n}_{\Omega} = 0 \quad \text{on } \partial\Omega_{\mathrm{N}}, \tag{2.6}$$

at the Dirichlet $\partial\Omega_{\mathrm{D}}$ and Neumann boundary $\partial\Omega_{\mathrm{N}}$, where $\vec{n}_{\Omega}$ is the outward pointing normal. And initial conditions

$$u(t=0) = 0 \quad \text{in } \Omega. \tag{2.7}$$

2.2 Surface transport of surfactants

The PDEs in the last section are defined on the whole domain Ω. It is however possible to define a PDE not on the whole domain, but on a submanifold, such as the interface $\mathfrak{I}$. One application for such surface-PDEs is for example the dynamics of surface active agents, so called surfactants. Looking at the flow of two insoluble liquids, such as water and oil, such a surfactant may for example be detergents, which are soluble in both phases. However, these detergents accumulate at the interface between the two phases, where their concentration is orders of magnitude higher than in the bulk phase, see e.g. Dziuk and Elliott (2013) and Gross et al. (2015). Thus, a simplified approximation for the dynamics of these surfactants is to model them as quantities, which are only defined at the interface and which do not interact with the surrounding bulk phases.

To accurately simulate diffusion along the surface, one must be able to discretize the Laplace-Beltrami operator $\Delta_{\mathfrak{I}}$, which is the projection of the Laplace Operator onto

the curved interface $\mathfrak{I}$. The surface-derivative $\nabla_\mathfrak{I}$ is defined as the projection of the gradient onto the surface

$$\nabla_\mathfrak{I} = P_\mathfrak{I}\nabla = (I - \vec{n}_\mathfrak{I} \otimes \vec{n}_\mathfrak{I})\nabla \tag{2.8}$$

In this work, we limit ourselves to the model problem of the surface-poisson equation on closed surfaces

$$\nabla_\mathfrak{I} \cdot \nabla_\mathfrak{I} u = f \quad \text{on } \mathfrak{I} \tag{2.9a}$$

$$\oint_\mathfrak{I} u = 0. \tag{2.9b}$$

2.3 Level set methods

In addition to describing the dynamics of quantities in the phases $\mathfrak{A}$ and $\mathfrak{B}$ and their coupling across the interface $\mathfrak{I}$, we need a model to describe the motion of the interface itself. In the literature, there are two different approaches available: interface tracking and interface capturing, see e.g. Gross and Reusken (2011) for an introduction and overview. Interface tracking explicitly describes the position of the interface by an explicit function for the interface position or by deforming a computational grid, which is aligned with the interface. The alternative are interface capturing techniques, which employ additional measures to implicitly describe the interface, such as particles (see e.g. the marker and cell method by Harlow and Welch (1965)) or globally defined functions such as the volume of fluid method by Hirt and Nichols (1981). The level set method by Osher and Sethian (1988) follows the second idea: a $D-1$ dimensional manifold embedded in a domain Ω of D dimensions (or in multiphase physics, the interface $\mathfrak{I}$) divides the domain into three parts: a phase $\mathfrak{A}$ on one side of the interface $\mathfrak{I}$, the phase $\mathfrak{B}$ on the other side and the interface $\mathfrak{I}$ itself. To implicitly describe this configuration, a global function, the level φ is chosen such that it has the signed distance property

$$\varphi(\vec{x}) = \begin{cases} \operatorname{dist}(\vec{x}, \mathfrak{I}(\varphi)) & \text{in } \mathfrak{A}(\varphi) \\ -\operatorname{dist}(\vec{x}, \mathfrak{I}(\varphi)) & \text{in } \mathfrak{B}(\varphi). \end{cases} \tag{2.10}$$

Then from a given level set φ given, the partitioning of the domain can be recomputed as

$$\mathfrak{A}(\varphi) = \{\vec{x} \in \Omega : \varphi(\vec{x}) > 0\} \tag{2.11a}$$

$$\mathfrak{B}(\varphi) = \{\vec{x} \in \Omega : \varphi(\vec{x}) < 0\} \tag{2.11b}$$

$$\mathfrak{I}(\varphi) = \{\vec{x} \in \Omega : \varphi(\vec{x}) = 0\}. \tag{2.11c}$$

If the interface is given, the signed distance formulation (2.10) can be computed by solving the Eikonal equation

$$|\nabla\varphi| - 1 = 0 \quad \text{in } \Omega \tag{2.12a}$$

with the boundary condition

$$\varphi = 0 \quad \text{on } \tilde{\mathfrak{I}} \tag{2.12b}$$

for a predefined interface $\tilde{\mathfrak{I}}$.
This definition of the interface has the property, that the normal on the interface can be computed in the whole domain by the relationship

$$\vec{n}_{\mathfrak{I}} = \frac{\nabla\varphi}{\|\nabla\varphi\|} = \nabla\varphi. \tag{2.13}$$

The motion of the interface is then given by the scalar transport equation

$$\frac{\partial\varphi}{\partial t} + \vec{u}_{\text{ext}} \cdot \nabla\varphi = 0, \tag{2.14}$$

with the velocity field $\vec{u}_{\text{ext}}$ being a smooth extension of the interface velocity $\vec{u}_{\mathfrak{I}}$, i.e.

$$\vec{u} = \vec{u}_{\mathfrak{I}} \quad \text{on } \mathfrak{I}. \tag{2.15}$$

Depending on this velocity field, this evolution may cause the level-set to become very steep or flat close to the interface. There are two possibilities to remedy this: the first is to use a process called reinitialization. The goal of reinitialization is to restore the signed distance properties of the level-set, i.e. to find a solution of equation (2.12). Starting point is an initial function $\tilde{\varphi}$, which defines the interface $\tilde{\mathfrak{I}} = \mathfrak{I}(\tilde{\varphi})$ and does not have signed distance properties.

The second possibility is to move the level set by an extension velocity, which is computed from

$$\nabla\vec{u}_{\text{ext},i} \cdot \nabla\varphi = 0 \quad \forall i \in \{1, \dots, D\}. \tag{2.16}$$

It can be shown, that a level set field retains its signed distance properties, if the velocity is constructed like that, see A.

Level-set methods are very popular in computational physics and engineering to describe the movement of fronts Adalsteinsson and Sethian (1995); Osher and Sethian (1988) or phase boundaries in multiphase problems Hou et al. (1997); Mulder et al. (1992); Sussman et al. (1994). In this work, we focus on multiphase flows in the context of high-order discontinuous Galerkin methods. There, reinitialization may suffer from instability and should provide the same accuracy as the flow solver. We present and critically examine a novel approach for these issues.
In the context of classical low order numerical schemes, the level-set method is well established and available for a multitude of physical applications, see Osher and Fedkiw (2001) for a review. They have further applications in computer graphics,

where they are used in image segmentation, see Li et al. (2005) and Liu et al. (2013) or way finding algorithms, see Kimmel and Sethian (1998). In computational engineering, high-order methods are becoming increasingly popular, since they allow an increased accuracy of computations, with the same computational effort compared to low-order methods. Combining the discontinuous Galerkin method with the level set method is popular to simulate multiphase flows, since the low numerical dissipation allows highly accurate computation of the level-set movement, see Sussman and Hussaini (2003); Grooss and Hesthaven (2006); Marchandise et al. (2007); Owkes and Desjardins (2013)and Pochet et al. (2013).

3 A discontinuous Galerkin method for unfitted problems

This chapter very briefly introduces the discontinuous Galerkin method (DG) for the example of a scalar conservation law. For a more detailed introduction, the reader might refer to the overview papers Cockburn et al. (2000) and the textbooks by Li (2006), Hesthaven and Warburton (2008) Di Pietro and Ern (2012). The method can be extended to problems which include interfaces, which separate the computational domain into two or more parts, the method presented here is the extended discontinuous Galerkin method (XDG) proposed by Kummer (2012, 2016). The time domain is discretized by implicit methods, which need to by extended to unfitted problems. This thesis uses the method by Kummer et al. (2018).

This introduction is based on the two peer-reviewed publications Utz et al. (2017b); Utz and Kummer (2017) and the book chapter Utz et al. (2017a), which have all been published by the author of this thesis.

3.1 State of the art

Discontinuous Galerkin methods were originally developed by Reed and Hill (1973) to simulate neutron transport and has been extended to general conservation laws by Cockburn and Shu (1991), to convection-diffusion by Cockburn and Shu (1998) and to the Stokes and Navier-Stokes problem by Cockburn et al. (2005); Girault et al. (2005) and Shahbazi et al. (2007). Recently, the method has gained quite some popularity, due to the fact that modern discontinuous Galerkin methods share the easy use of unstructured meshes and with the finite element method and the conservation properties with the finite volume method. Thus, many insights gained using DG may be applied to the other two methods and vice-versa. One example is the SIMPLE-algorithm, which is popular for the solution of the saddle-point system arising from the discretization of the Navier-Stokes. This algorithm, which was originally developed for the finite volume method, has been successfully applied to DG discretizations by Klein et al. (2013a,b). Another example is the adaption of the extended finite element method (XFEM) to DG in the context of multiphase flows by Kummer (2012); Heimann et al. (2013); Fechter and Munz (2015) and Saye (2017). Due to the flux formulation, for the DG method, only information across an edge between two cells has to be exchanged, which limits communication in parallel computations to a minimum, which makes the method scale up to thousands of processors, even on complicated geometries, see Beck et al. (2014) and Sonntag and Munz (2017).

3.2 Introduction to the discontinuous Galerkin method

The discontinuous Galerkin method is a numerical method to discretize conservation laws for quantities u of the form

$$\partial_t u + \nabla \cdot f(u) = 0, \tag{3.1}$$

which shall be solved on a domain Ω. The first step to do so, is to derive the weak form of the conservation law by multiplying it by a test function v and integrating over the domain. Integration by parts then yields

$$\partial_t \int_\Omega uv \,\mathrm{d}V + \int_\Omega \nabla \cdot f(u)\, v \,\mathrm{d}V = \partial_t \int_\Omega uv \,\mathrm{d}V - \int_\Omega f(u) \cdot \nabla v \,\mathrm{d}V + \int_{\partial\Omega} f(u) \cdot \vec{n}_\Omega v \,\mathrm{d}S = 0. \tag{3.2}$$

This weak form is the starting point for many computational methods for partial differential equations. Note, that this weak form includes an integration over the spatial domain, but not over the time domain. Thus, all discretization approximating the integrals given here only approximate the spatial domain, time discretizations are discussed in the section 3.5. While the discontinuous Galerkin method can be used for time discretization as well (so-called space-time DG, see section 3.5.2), this introduction focuses on the spatial discretization only.

For the discontinuous Galerkin method, the D-dimensional domain $\Omega \subset \mathbb{R}^D$ is approximated by the computational Domain $\Omega_h = \bigcup_{K \in \mathfrak{K}} K$ which consists non-overlapping cells K with a characteristic mesh size h. Together, these cells form the grid (or mesh) $\mathfrak{K} = \{K_1, \ldots, K_J\}$. The outward pointing normal of each cell is denoted by $\vec{n}_e$. The skeleton Γ of this grid is defined as the union of all cell edges

$$\Gamma = \bigcup_{K \in \mathfrak{K}} \partial K. \tag{3.3}$$

On this grid the weak form (5.11) is interpreted in a cell local way. To do so, the test function v and the trial functions u are defined on each cell separately. Thus, they are discontinuous across the cell interfaces. Typically, the functions are chosen as polynomials of degree k. This means, we choose the test and trial functions

$$u, v \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}) \tag{3.4a}$$

from the broken polynomial space

$$\mathbb{V}_{\mathrm{DG}}(\mathfrak{K}) := \{f \in L^2(\Omega_h);\ \forall K \in \Omega_h : f|_K \in \mathbb{P}_k(K)\}. \tag{3.4b}$$

We denote the value at the inner side of an element by the superscript "$-$", the outside by the superscript "$+$". Then, we can define the jump and mean operators:

$$\{\vec{u}\} = \frac{1}{2}\left(\vec{u}^{+} + \vec{u}^{-}\right) \tag{3.5a}$$

$$[\![u]\!] = u^{+} + u^{-} \tag{3.5b}$$

Using this definitions, the functions u and v can be plugged into the weak form (3.2). However, these leads to a decoupling of the individual cells from each other. Therefore we need to introduce a numerical flux $\tilde{f}$, which couples adjacent cells. Doing so we end up with the spatial discretization

$$\begin{aligned} &\text{find } u \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}) \text{ s.t.} \\ \partial_t \sum_{K\in\mathfrak{K}} \int_{K_i} uv \,\mathrm{d}V - \sum_{K\in\mathfrak{K}} \int_{K_i} f(u)\cdot\nabla v \,\mathrm{d}V + \sum_{e\in\Gamma} \int_e \tilde{f}(u^{+}, u^{-}, \vec{n}_e)\,[\![v]\!]\,\mathrm{d}S \\ &= 0 \quad \forall v \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}) \end{aligned}$$

since the sum over all elements and edges is the same, as the integral over the domain, this is usually written as

$$\begin{aligned} &\text{find } u \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}) \text{ s.t.} \\ \partial_t \int_{\Omega_h} uv \,\mathrm{d}V - \int_{\Omega_h} f(u)\cdot\nabla v \,\mathrm{d}V + \int_{\Gamma} \tilde{f}(u^{+}, u^{-}, \vec{n}_e)\,[\![v]\!]\,\mathrm{d}S \\ &= 0 \quad \forall v \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}). \end{aligned} \tag{3.6}$$

This numerical flux must be continuous, consistent with the original PDE, i.e.

$$\tilde{f}(u, u, \vec{n}) = f(u)\cdot\vec{n}, \tag{3.7}$$

conservative

$$\tilde{f}(u^{+}, u^{-}, \vec{n}) = -\tilde{f}(u^{-}, u^{+}, -\vec{n}) \tag{3.8}$$

and enforce the boundary conditions at the edge of the domain. Since the function space $\mathbb{V}_{\mathrm{DG}}$ is polynomial, the function u can be written as a linear combination of basis function $\phi_i \in \mathbb{V}_{\mathrm{DG}}$

$$u = \sum_i \hat{u}_i \phi_i \tag{3.9}$$

with $\hat{u}_i$ being the scalar coefficients. We choose orthonormal basis functions for ϕ_i, i.e. they have the property

$$\int_K \phi_i \phi_j \,\mathrm{d}V = \delta_{ij} \tag{3.10}$$

with the Kronecker Delta δ_{ij}.

In the end, the discretized form (3.6) can be written as the nonlinear vector-equation

$$\mathbf{N}(\vec{\hat{u}}) = \vec{b}, \tag{3.11}$$

which simplifies for linear PDEs to the linear system

$$\mathbf{A}\vec{\hat{u}} = \vec{b}. \tag{3.12}$$

This linear system can be solved by appropriate direct or iterative methods, see the textbook by Meister (2011). The direct methods used in Bounded Support Spectral Solver (BoSSS) are the PARDISO package by Schenk and Gärtner (2004) and the MUMPS package Amestoy et al. (2006). Since the systems shown in this thesis are comparably small, they are exclusively solved by such direct methods, instead of employing iterative methods.

3.3 Adaptation of DG to unfitted problems

The previous chapter introduced a method for a general conservation law which is defined on the whole domain Ω. This method however is not suited for problems that include conditions at the interface, such as (2.4a) or (2.4b). This chapter introduces an extension to standard DG schemes that allows the solution of such problems. For details on the discretization, the reader might refer to the publications Kummer (2012) and Kummer (2016). The starting point is a steady state two phase problem with an interface condition, which might be modeled as

$$\nabla \cdot f_{\Omega_i}(u) = 0 \quad \text{in } \Omega \quad \forall \Omega_i = \{\mathfrak{A}, \mathfrak{B}\} \tag{3.13a}$$

$$[\![f(u)]\!] \cdot \vec{n}_{\mathfrak{I}} = 0 \quad \text{on } \mathfrak{I}, \tag{3.13b}$$

where material parameters in the fluxes $f_{\mathfrak{A}}$ and $f_{\mathfrak{B}}$ differ. This additional interface condition may cause a kink or jump of the quantity u across the interface. If the interface is not aligned with the grid, which is the general case, this jump or kink cannot be approximated using the piecewise smooth function space $\mathbb{V}_{\mathrm{DG}}$ that we introduced previously. Therefore we employ a method first presented by Kummer (2012) which is closely related to the XFEM method in the context of multiphase flows) and therefore called unfitted or extended discontinuous Galerkin method (XDG). See e.g. Gross and Reusken (2011); Sauerland (2013) and Lehrenfeld (2015) for XFEM related literature in the context of multiphase flows.

The interface $\mathfrak{I}$ might be arbitrarily positioned in the domain. Then some cells are located fully in the phase $\mathfrak{A}$ or $\mathfrak{B}$. Cells which include the interface are called cut cells. Thus,we can define a new grid, the so called cut cell grid which is

$$\mathfrak{K}_{Cut} = \{K_1 \cap \mathfrak{A}, K_1 \cap \mathfrak{B}, \ldots, K_j \cap \mathfrak{A}, K_j \cap \mathfrak{B}\} \tag{3.14}$$

For the ease of notation, this includes cells with measure zero. Using a DG space on this cut cell mesh is equivalent to the extended $\mathbb{V}_{\text{DG}}$ space on the original mesh:

$$\mathbb{V}_{\text{DG}}^{X}(\mathfrak{K}) = \mathbb{V}_{\text{DG}}(\mathfrak{K}_{Cut}) \tag{3.15}$$

Simply speaking, on cells, which do not contain the interface, these are the same cell local polynomials. On cells which do contain the interface (and thus are cut by it), this defines two separate degrees of freedom on the two halves of the cell $u|_{K\cap\mathfrak{A}}$ and $u|_{K\cap\mathfrak{B}}$. This requires individual test functions on each half $v|_{K\cap\mathfrak{A}}$ and $v|_{K\cap\mathfrak{B}}$. With this definition, the spatial discretization of the two-phase conservation law (3.13) is

$$\begin{aligned} &\text{find } u \in \mathbb{V}_{\text{DG}}^{X}(\mathfrak{K}) \text{ s.t.} \\ -\int_{\mathfrak{A}\cup\mathfrak{B}} f(u)\cdot\nabla v \,\mathrm{d}V &+ \int_{\Gamma\cap(\mathfrak{A}\cup\mathfrak{B})} \tilde{f}(u^+, u^-, \vec{n}_e)\,[\![v]\!]\,\mathrm{d}S \\ &+ \int_{\mathfrak{I}} (\tilde{f}(u_{\mathfrak{A}}, u_{\mathfrak{B}}, \vec{n}_{\mathfrak{I}}))\,[\![v]\!]\,\mathrm{d}S \\ &= 0 \quad \forall v \in \mathbb{V}_{\text{DG}}^{X}(\mathfrak{K}). \end{aligned} \tag{3.16}$$

This gives a well-defined method, which has the same form as the original DG-scheme but is defined on cut-cells $K\cap\mathfrak{A}$ and $K\cap\mathfrak{B}$ instead of the original cells K with fluxes across cut edges $\partial K\cap\mathfrak{A}$, $\partial K\cap\mathfrak{B}$ and the interface $\mathfrak{I}$. Again the numerical flux $\tilde{f}$ must be chosen appropriately to the given differential equations and the boundary and jump conditions. This means, the weak form remains unchanged, but the method is now defined on cut cells $K\cap\mathfrak{A}$ which may have an arbitrary shape. This requires an integration method on these cut cells.

3.4 The hierarchical moment fitting method

When employing a sharp interface method to discretize a multiphysics problem, such as (3.16) the burden of discretization lies in the numerical integration over the interface and over the individual parts of the cells cut by it. Typical integration techniques for such an integral require a reconstruction of the interface, see e.g. Min and Gibou (2007, 2008); Engwer (2009); Müller et al. (2012); Gross and Reusken (2011). However, this appraoch is limited in its accuracy due to runtime restrictions, see Min and Gibou (2007). We therefore chose a different approach introduced by Müller et al. (2013) called hierarchical moment-fitting (HMF) which allows accurate integration over cut cells and surfaces without an explicit reconstruction of the interface.

The idea of Müller et al. (2013) is to construct integration rules, such as the Gauß quadrature, but individually tailored to the individual cut cell. The concept behind such an integration rule is to approximate the integration of a function f_i over a domain

Ω, by sum of the function multiplied by weights $\mathfrak{W} = \{w_i\}_{i=1,\dots,N}$ evaluated at the points $\mathfrak{X} = \{\vec{x}_i\}_{i=1,\dots,N}$.

$$\sum_{i=1}^{N} f(x_i) w_i = \int_\Omega f \,\mathrm{d}V \tag{3.17}$$

For multiple functions $\mathfrak{B} = \{f_j\}_{j=1,\dots,M}$, then gives the system

$$\underbrace{\begin{pmatrix} f_1(\vec{x}_1) & \dots & f_1(\vec{x}_N) \\ \vdots & \ddots & \vdots \\ f_M(\vec{x}_1) & \dots & f_M(\vec{x}_N) \end{pmatrix}}_{=:A} \begin{pmatrix} w_1 \\ \vdots \\ w_N \end{pmatrix} = \underbrace{\begin{pmatrix} \int_\Omega f_1 \,\mathrm{d}V \\ \vdots \\ \int_\Omega f_M \,\mathrm{d}V \end{pmatrix}}_{=:b}. \tag{3.18}$$

The idea behind HMF is, to use this system the other way around: By picking a specific basis $\mathfrak{B}$ of $\mathbb{R}^D$, the right hand side of this system can be evaluated exactly. If the points $\mathfrak{X}$ are fixed, this leads to a linear system of the form

$$A_{ji} w_i = \int_\Omega f_j \,\mathrm{d}V. \tag{3.19}$$

The exact integration of the right hand side of (3.18) is achieved by exploiting Gauss's theorem

$$\int_\Omega \nabla \cdot \vec{f} \,\mathrm{d}V = \oint_{\partial\Omega} \vec{f} \cdot \vec{n}_\Omega \,\mathrm{d}S. \tag{3.20}$$

This means, if an integration rule over the boundary of a domain $\partial\Omega$ and its normals are available, one can generate a quadrature rule for the whole domain. The idea can be applied even further: if the function space $\mathfrak{B}$ is chosen from divergence free functions $\vec{f}^*$, the relationship can be further used to generate quadrature rules for the , if the integral over the rest of the domain is known.

$$\int_{\mathfrak{A}} \nabla \cdot \vec{f}^* \,\mathrm{d}V = \oint_{\partial\mathfrak{A}} \vec{f}^* \cdot \vec{n}_\Omega \,\mathrm{d}S = \oint_{(\partial\Omega\cap\mathfrak{A})\cup\mathfrak{I}} \vec{f}^* \cdot \vec{n}_\Omega \,\mathrm{d}S = \int_{\partial\Omega\cap\mathfrak{A}} \vec{f}^* \cdot \vec{n}_\Omega \,\mathrm{d}S + \int_{\mathfrak{I}} \vec{f}^* \cdot \vec{n}_{\mathfrak{I}} \,\mathrm{d}S = 0 \tag{3.21}$$

Using this approach Müller et al. (2013) and Kummer (2016) develop a hierarchy of quadrature methods,which allow accurate integration over surfaces and cut cells in arbitrary spatial dimensions. Their method is used for all such integrations throughout this thesis. Since the details of the HMF method are not repeated here, the reader might refer to the original publications by Müller et al. (2013) and Kummer (2016) and the PhD thesis by Müller (2014).

3.5 Temporal discretization

The precious chapters discussed a discretization of the spatial domain for the example of a scalar conservation law (3.1). This resulted in the semi-discrete equation (3.6). In general this equation can be written as the system of ordinary differential equations (ODEs) for an unknown vector $\vec{u}$

$$\partial_t \vec{u} = F(\vec{u}) \quad \text{in } t \in (0, T). \tag{3.22}$$

For a detailed introduction to schemes for ordinary differential equations of this kind, the reader might refer to one of the numerous introductory books on numerics, e.g. Schwarz and Köckler (2013). In the context of discontinuous Galerkin methods, Di Pietro and Ern (2012) and Hesthaven and Warburton (2008) give good introduction.

All time stepping schemes aim to calculate the quantity $u(t^n)$ at a certain time level t^n from values at previous time levels $t^{n-1}, t^{n-2}, \dots$, where $\Delta t = t^n - t^{n-1}$ is the time step size. The methods for solving this equation are typically split into two groups: explicit methods, which involve evaluations of F at time steps which have already been computed, and implicit methods, which involve this function at points in time, which have not yet been computed, and thus require an inversion of F.

While explicit methods are quite popular in the context of DG methods for hyperbolic equations, they suffer from the drawback of being limited by a maximal time step size Δt. This time step size typically scales with the grid size as $\Delta t \sim \frac{h}{k}$ for hyperbolic and even $\Delta t \sim \left(\frac{h}{k}\right)^2$ for parabolic problems, where h is the size of the computational cells and k is the polynomial degree for u see e.g. Gassner et al. (2007). In contrast to this, many implicit methods are designed such that they do not have any or at least not such a severe restriction to their time step size.

3.5.1 Implicit time stepping schemes

This thesis aims to avoid the restriction of the times step size by employing implicit time stepping schemes of the form

$$\frac{\beta_0}{\gamma \Delta t} u(t^n) - \alpha F(u(t^n)) = -\sum_{i=1}^{s} \frac{\beta_i}{\gamma \Delta t} u(t^{n-i}) + (1-\alpha) F(u(t^n)). \tag{3.23}$$

The coefficients for these backward differentiation formula (BDF) are given in the table 3.1.

Calculating the unknown $u(t^n)$ requires an inversion of the function F. In the context of DG methods this is the solution of the system (3.6) arising from the discretization. From the schemes available from table 3.1, the Crank-Nicholson Scheme is chosen for all time-dependent equations in this thesis. The reason for this is, that only the BDF 1, the BDF 2 and the Crank-Nicholson scheme are unconditionally stable, while there

Table 3.1: Coefficients for the BDF schemes of different order s.

Scheme s	α	γ	β_0	β_1	β_2	β_3	β_4
BDF 1 (Implicit Euler)	1	1	1	-1			
BDF 2	1	2	3	-4	1		
BDF 3	1	6	11	-18	9	-2	
BDF 4	1	12	25	-48	36	-16	3
Crank-Nicholson	$\frac{1}{2}$	1	1	-1			

are time step restriction for the higher order BDF schemes. The implicit Euler scheme (BDF 1) is only first order accurate, which causes the time stepping to be the limiting factor for the accuracy of simulations. The second order BDF scheme gives the same accuracy as the Crank-Nicholson scheme, but additionally requires values at t^{n-2}.

3.5.2 Extension to moving interfaces

The XDG idea of extending the polynomial space on elements cut by the interface can be used to discretize problems involving a moving interface. In this case however, both parts of the domain $\mathfrak{A}(t)$ and $\mathfrak{B}(t)$ are time dependent, thus the spatially discrete but temporally continuous form of the PDE cannot be discretized in time by applying an algorithm for solving systems of ODEs.

$$\int_{\mathfrak{A}(t)} \partial_t uv \,\mathrm{d}V \neq \partial_t \int_{\mathfrak{A}(t)} uv \,\mathrm{d}V \tag{3.24}$$

Therefore we need to use a more general approach to solving problems including a moving interface. To do so, the original conservation law (3.1) can be rewritten as

$$\partial_t u + \nabla \cdot f(u) = \begin{pmatrix} \partial_t \\ \nabla \end{pmatrix} \cdot \begin{pmatrix} u \\ f(u) \end{pmatrix} = \nabla^* \cdot f^*(u) = 0. \tag{3.25}$$

A Rankine-Hugoniot condition with an interface velocity $s = \vec{u}_\mathfrak{I} \cdot \vec{n}_\mathfrak{I}$ can be expressed as

$$[\![f \cdot \vec{n}_\Omega]\!] - [\![su]\!] = \left[\!\!\left[\begin{pmatrix} u \\ f(u) \end{pmatrix} \cdot \begin{pmatrix} -s \\ \vec{n}_\mathfrak{I} \end{pmatrix} \cdot \right]\!\!\right] = [\![f^*(u) \cdot \vec{n}^*_\mathfrak{I}]\!] = 0 \quad \text{on } \mathfrak{I}^* \tag{3.26}$$

The time interval and spatial domain on which this problem is to be solved, can also be rewritten as

$$\Omega^* = (0,T) \times \Omega(t) \tag{3.27a}$$
$$\mathfrak{A}^* = (0,T) \times \mathfrak{A}(t) \tag{3.27b}$$
$$\mathfrak{B}^* = (0,T) \times \mathfrak{B}(t) \tag{3.27c}$$
$$\mathfrak{I}^* = (0,T) \times \mathfrak{I}(t). \tag{3.27d}$$

Thus, the whole multiphase conservation law is

$$\nabla^* \cdot f^*(u) = 0 \quad \text{on } \Omega^* \tag{3.28}$$

$$[\![f^*(u) \cdot \vec{n}^*_{\mathfrak{I}}]\!] = 0 \quad \text{on } \mathfrak{I}^*. \tag{3.29}$$

With the only difference being the index $*$ this is the same as the steady state equation (3.13). This is the so called space-time formulation of the problem. While imagining an additional dimension is rather challenging, the numerical methods for such an equation are exactly the same as for a PDE that only defined on a spatial domain. Of course such an equation can be discretized by any Galerkin method just like any other conservation law. In the context of discontinuous Galerkin methods, this approach is called space-time DG and has been successfully applied to problems involving moving interfaces and unfitting discretizations by Lehrenfeld (2014, 2015) or Weller and Bänsch (2017). For a general introduction to space time DG, the reader might refer to the textbook by Thomée (2006).

This space-time domain is shown in figure 3.1. Conceptually, designing a numerical method for this domain is the same as for a domain with independent variables x and y instead of $\vec{x}$ and t.

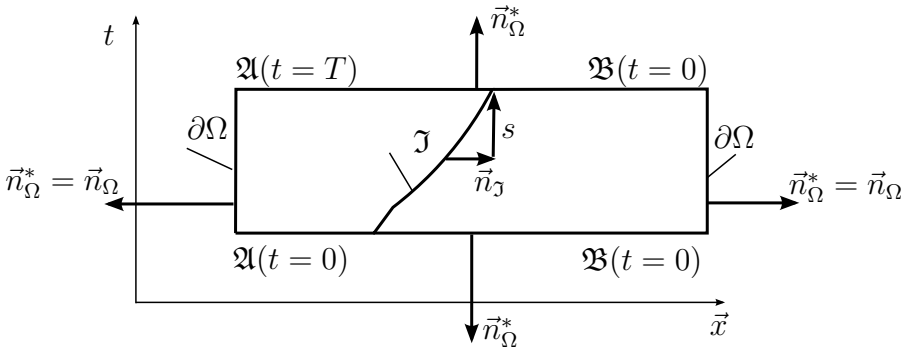


Figure 3.1: Space-Time domain including a moving interface $\mathfrak{I}$

Multiplying equation (3.29) by a space-time test function v^* and integrating by parts over the space-time domain gives

$$-\int_{\Omega^*} f^*(u) \cdot \nabla^* v^* \,\mathrm{d}V^* + \int_{\partial\Omega^*} f^*(u) \cdot \vec{n}_{\Omega^*}\, v^* \,\mathrm{d}S^* + \int_{\mathfrak{I}^*} [\![f^*(u) \cdot \vec{n}^*_{\mathfrak{I}}]\!]\, v^* \,\mathrm{d}S^* = 0. \tag{3.30}$$

This equation can be solved directly by employing an unfitted discontinuous Galerkin discretization for the space-time domain. This however requires the solution of large linear systems, since all temporal degrees of freedom must be solved at the same time. For special choices of the temporal test functions and an XDG discretization in space, the method can be reduced to classical implicit time stepping schemes such as the implicit Euler or the Crank-Nicholson timestepping scheme, with additional terms considering the motion of the interface Kummer et al. (2018).

This is the approach used here, following Kummer et al. (2018). The integrals over the space-time domain and its boundaries can be rewritten into individual integration steps over time and space. For the general case (3.30) this then reads

find $u \in \mathbb{V}_{\mathrm{DG}}^{X}$ s.t.

$$\begin{aligned} &- \int_{t^n}^{t^{n+1}} \int_{\mathfrak{A}(t)\cup\mathfrak{B}(t)} f(u) \cdot \nabla v \,\mathrm{d}V \,\mathrm{d}t \\ &+ \int_{\mathfrak{A}(t)\cup\mathfrak{B}(t)} uv \,\mathrm{d}V \Big|_{t=t^{n+1}} - \int_{\mathfrak{A}(t)\cup\mathfrak{B}(t)} uv \,\mathrm{d}V \Big|_{t=t^{n}} \\ &+ \int_{t^n}^{t^{n+1}} \int_{\Gamma\cap(\mathfrak{A}(t)\cup\mathfrak{B}(t))} \tilde{f}(u^+, u^-, \vec{n}_e) \,[\![v]\!] \,\mathrm{d}S \,\mathrm{d}t \\ &+ \int_{t^n}^{t^{n+1}} \int_{\mathfrak{I}} (\tilde{f}(u_{\mathfrak{A}}, u_{\mathfrak{B}}, \vec{n}_{\mathfrak{I}}) - s\,[\![u]\!])\,[\![v]\!] \,\mathrm{d}S \,\mathrm{d}t \\ &\qquad = 0 \quad \forall v \in \mathbb{V}_{\mathrm{DG}}^{X}(\mathfrak{K}). \end{aligned} \tag{3.31}$$

It is important to note, that it is required to evaluate the mass matrices, i.e. the first two terms in (3.31) at both time levels individually. Failing to do so leads to a consistency error of the time discretization as noted by Fries and Zilian (2009) and Kummer et al. (2018).

For the simulations shown in the last chapter, 8 a Crank-Nicholson time stepper is used. This means, the temporal integrals in equation (3.31) are approximated as

$$\int_{t^n}^{t^{n+1}} (\dots)\,\mathrm{d}t = \frac{\Delta t}{2}\Big((\dots)|_{t^n} + (\dots)|_{t^{n+1}}\Big) \tag{3.32}$$

For further details on the method, especially on the treatment of small cut cells and interfaces which move across cell boundaries, the reader might refer to the original publication by Kummer et al. (2018).

4 Level set reinitialization

This chapter discusses an algorithm to restore the signed-distance property of the level-set function, which is based on solving an elliptic PDE in contrast to commonly used algorithms. Main parts of this chapter are based on the publication Utz et al. (2017b), where this algorithm was first introduced and only minor formulations have been changed. Section 4.4 extends the method from the original publication by a preconditioning method involving a first order fast marching scheme.

Let's revisit some definitions from chapter 2: If the level set function is defined as in equation (2.10),

$$\varphi(\vec{x}) = \begin{cases} \quad \text{dist}(\vec{x}, \mathfrak{I}(\varphi)) & \text{in } \mathfrak{A}(\varphi) \\ - \ \text{dist}(\vec{x}, \mathfrak{I}(\varphi)) & \text{in } \mathfrak{B}(\varphi) \end{cases} \qquad \text{(see 2.10)}$$

the norm of the gradient is always one, which defines the Eikonal equation

$$|\nabla\varphi| = 1. \qquad \text{(see 2.12)}$$

The normal on the interface can be computed as

$$\vec{n}_\mathfrak{I} = \frac{\nabla\varphi}{\|\nabla\varphi\|} = \nabla\varphi. \qquad \text{(see 2.13)}$$

This means, the closest point on the interface can be computed as

$$x_{\text{cp}} = x - \varphi(x)\frac{\nabla\varphi}{\|\nabla\varphi\|}. \qquad (4.1)$$

As already mentioned in section 2.3, the level set may loose its signed distance properties due to advection. The dashed lines in figure 4.1 depict two possible situations, in which this becomes troublesome. First, with decreasing level set gradient, determining the position of the interface becomes increasingly bad conditioned. Second, the level set might become disturbed or even oscillate, such that the level set gradient becomes very steep or even switches its sign. In such a case, the equation (4.1) does not give convergent results and the normal calculated from equation (2.13) is no feasible approximation to the interface gradient anymore, which is problematic for the quadrature technique shown in section 3.4. Thus, we need a procedure, to recover the signed distance property (2.13) from a given level set field.

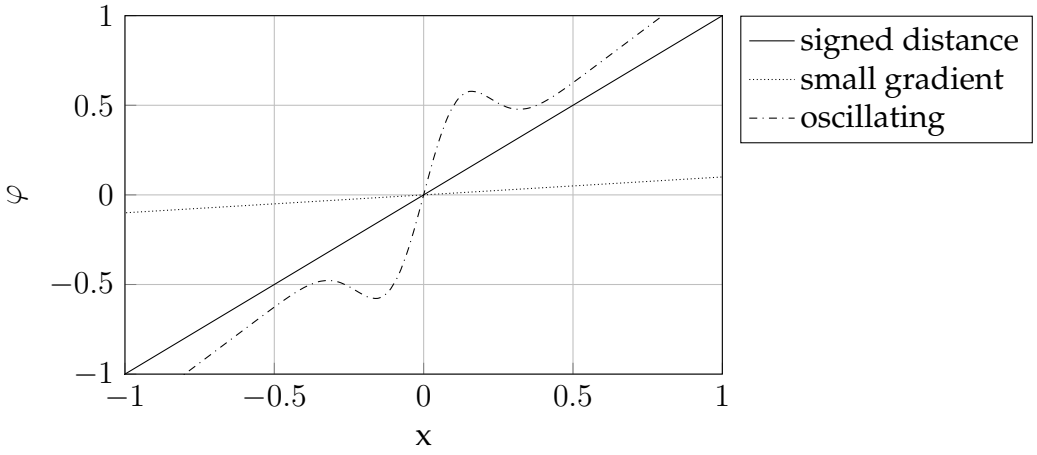


Figure 4.1: Deviation of the level set from the signed distance property

4.1 Algorithms for level set reinitialization

The techniques used for reinitialization can be divided in multiple groups: direct and geometry-based approaches, which aim to solve (2.10) and PDE based algorithms, which seek for a solution of (2.12).

4.1.1 Direct and geometric algorithms

Direct approaches calculate the distance to the interface for every grid point in the computational domain individually, usually by including some kind of search algorithm. In the discontinuous Galerkin context, this approach is used by Desjardins and Pitsch (2009); Marchandise et al. (2007); Marchandise (2006); Pochet et al. (2013) and Saye (2014).

Fast marching algorithms iteratively extend the field point by point starting from the zero contour. The method was originally developed in the finite difference context by Adalsteinsson and Sethian (1995, 1999) and is well established therein with contributions to improve the accuracy to second order by Chéné et al. (2007) and Chopp (2001) and to third order by Luddens et al. (2015). The method has been extended to finite element methods on unstructured grids and general triangulated surfaces by Kimmel and Sethian (1998) and was further improved by Elias et al. (2007) and Groß et al. (2006). Sussman and Hussaini (2003) applied fast marching to a high-order discontinuous Galerkin method on structured grids.

Fast sweeping methods are a second possibility. The main idea is to update the information at one point, or in one cell, based on the values of the surrounding cells by ”Gauss-Seidel iterations with alternating sweeping orderings”(Zhao (2005)). Fast sweeping is well established for several numerical methods: Originally, the method was designed for the finite difference method by Tsai et al. (2003) and Zhao (2005). In this context it was later extended for efficient parallel implementations Jeong and

Whitaker (2008) and Detrixhe et al. (2013). Wu and Zhang (2014) extended the method to achieve third order accuracy. Qian et al. (2007) applied the method to finite element method (FEM) with first order accuracy, Xia et al. (2008) published an application of fast sweeping to DG on triangular grids with second order accuracy . For triangular grids, the method was further improved for parallel performance by Fortmeier and Bücker (2011); Fu et al. (2011, 2013) and Shakoor et al. (2015). Cheng and Shu (2007) applied the method to DG simulations on structured, quadrilateral grids and multiple groups improved this to second order accuracy (see Li et al. (2008); Zhang et al. (2011) and Luo (2013)) and third order accuracy (see Wu and Zhang (2014)). These DG-based approaches rely on solving the Eikonal equation (2.12) or a reformulation for each cell locally.

To the author's knowledge neither fast marching, nor fast sweeping algorithms are available, which allow higher than third-order accuracy on unstructured grids.

4.1.2 PDE-based algorithms

While the previous algorithms rely on a point wise or cell wise update of the level-set, PDE-based approaches reformulate the problem (2.12) and solve a global system. The resulting PDEs can be incorporated into the equation describing the level-set motion or solved individually, or may serve as the base for local solvers in a fast marching or sweeping method. These algorithms can be divided into approaches, which directly solve the Hamilton-Jacobi problem (see Cheng and Shu (2007); Hu and Shu (1999)), hyperbolic (see Sussman et al. (1994)) and parabolic (see Li et al. Li et al. (2005)) approaches. A recent addition to PDE based reinitialization techniques is the approach by Basting and Kuzmin (2012), which is based on solving an elliptic problem. The approach developed there has caught some attention in the literature (see Basting and Kuzmin (2014); Holmgren and Kreiss (2015); Rasthofer (2015)) and was considered a promising alternative for level-set reinitialization, but not investigated further. Therefore this approach was extended to DG by the author of this thesis (see the publication Utz et al. (2017b)).

4.1.3 Hyperbolic reinitialization

The most popular choice for PDE-based algorithm is to reformulate (2.12) as the time dependent, hyperbolic problem

$$\frac{\partial \varphi}{\partial t} - \text{sign}(\tilde{\varphi})\left(|\nabla \varphi| - 1\right) = 0 \quad \text{in } \Omega \tag{4.2a}$$

with the initial condition

$$\varphi(\vec{x}, t = 0) = \tilde{\varphi}(\vec{x}). \tag{4.2b}$$

Then, the steady-state solution of this problem is the solution of the Eikonal equation. This idea was originally developed by Sussman et al. (1994) for a finite difference

discretization and was further improved by Russo and Smereka (2000). Della Rocca and Blanquart (2014) included a ghost-cell method to allow reinitialization, if the zero-contour intersects the boundary of the domain. In the context of finite element and discontinuous Galerkin methods, special measures have to be taken, to stabilize the algorithm: Grooss and Hesthaven (2006) modify the hyperbolic reinitialization to flatten the level-set function far from the interface, thus avoiding the influence of singularities outside a narrow region around the interface. Afterwards, they remove high frequency oscillations by a filtering technique. Hyperbolic reinitialization methods may suffer from a shift and oscillations of the zero-isocontour, such that the pseudo-time stepping does not convergence to a steady state. This has been reported in the context of discontinuous Galerkin methods by Mousavi (2014) and in the context of finite element methods by Basting and Kuzmin (2012), as well as in the finite difference context, where Hartmann et al. (2008, 2010) solve the issue by explicitly calculating the signed distance solution in the cells cut by the interface $\mathfrak{I}$. Therefore, this thesis focuses on a different approach by using a method based on solving an elliptic system.

4.2 Elliptic reinitialization

Elliptic reinitialization aims to avoid oscillations in the zero-isocontour and stability issues, associated with hyperbolic reinitialization. The idea is to consider an energy functional R of the form

$$R(\varphi) = \int_{\Omega} \psi\left(|\nabla\varphi|\right) \, \mathrm{d}\vec{x} \tag{4.3}$$

with the potential function $\psi(|\nabla\varphi|)$. The simplest choice for the potential is the single well $\psi(s) = 1/2\,(s-1)^2$:

$$R(\varphi) = \frac{1}{2}\int_{\Omega} \left(|\nabla\varphi| - 1\right)^2 \, \mathrm{d}\vec{x},$$

which is discussed in section 4.2.1. The function has an associated diffusion rate d, which is defined as

$$d(s) = \frac{\mathrm{d}\psi(s)}{\mathrm{d}(s)}\frac{1}{s}. \tag{4.4}$$

Minimizing this functional is equivalent to searching for the steady-state-solution of the problem

$$\frac{\partial\varphi}{\partial t} + \frac{\partial R}{\partial\varphi} = 0, \tag{4.5}$$

see Li et al. (2005), which is called parabolic reinitialization.

Instead of evolving this time dependent PDE to steady state, Basting and Kuzmin (2012) propose to directly solve the minimization problem

$$\min R(\varphi) \quad \text{s.t. } \varphi = 0 \text{ on } \mathfrak{I}(\tilde{\varphi}). \tag{4.6}$$

The constraint at the interface can be enforced by a penalty term

$$P(\varphi) = \alpha \int_{\mathfrak{I}(\tilde{\varphi})} \frac{\varphi^2}{2} \,\mathrm{d}S, \tag{4.7}$$

where α is a sufficiently large penalty parameter. This penalty term is discussed in detail in section 4.2.3.

Taking the functional derivative gives the variational form (or weak form)

$$\delta R(\varphi, v) + \delta P(\varphi, v) = \int_{\Omega} d(|\nabla\varphi|)\nabla\varphi \cdot \nabla v \,\mathrm{d}\vec{x} + \alpha \int_{\mathfrak{I}(\tilde{\varphi})} \varphi v \,\mathrm{d}S = 0 \quad \forall v \tag{4.8}$$

with the test function v. The strong form of the minimization problem is

$$\frac{\partial R}{\partial \varphi} = \nabla \cdot (d\,(|\nabla\varphi|)\,\nabla\varphi) = 0 \quad \text{in } \Omega \tag{4.9a}$$

$$\varphi = 0 \quad \text{on } \mathfrak{I}(\tilde{\varphi}). \tag{4.9b}$$

Note, that equation (4.9a) can be split into a linear part $\Delta\varphi$ and a nonlinear part $f(\varphi)$

$$\Delta\varphi - \nabla \cdot (1 - d\,(|\nabla\varphi|)\,\nabla\varphi) = \Delta\varphi - f(\varphi) = 0 \tag{4.10}$$

This is used later to define an iteration scheme, where a Poisson equation has to be solved in each iteration step.

4.2.1 Potential functions

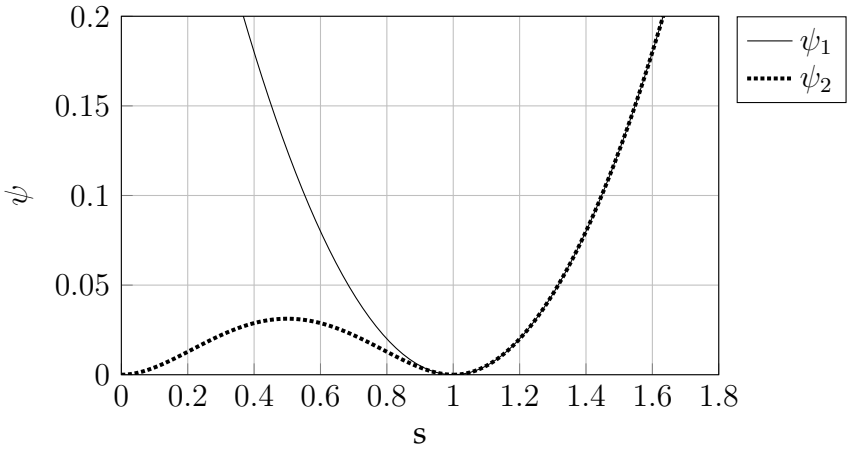


Figure 4.2: Single-well (ψ_1) and double-well (ψ_2) potential function

Basting and Kuzmin (2012) use two different formulations for the potential function ψ, see figure 4.2: the single-well potential

$$\psi_1(s) = 1/2(s-1)^2 \tag{4.11a}$$

$$d_1(s) = 1 - \frac{1}{s} \tag{4.11b}$$

and the double-well potential

$$\psi_2(s) = \begin{cases} 1/2s^2(s-1)^2 & \text{if } s \leq 1 \\ 1/2(s-1)^2 & \text{if } s > 1 \end{cases} \tag{4.12a}$$

$$d_2(s) = \begin{cases} 2s^2 - 3s + 1 & \text{if } s \leq 1 \\ 1 - \frac{1}{s} & \text{if } s > 1 \end{cases}. \tag{4.12b}$$

Note, that for ψ_1 only $s = |\nabla\varphi| = 1$ is a solution, while ψ_2 possesses the additional solution $s = |\nabla\varphi| = 0$. As Basting and Kuzmin (2012) and Li et al. (2010) already pointed out, the single well potential leads to numerical instabilities in regions, where $\nabla\varphi$ goes to zero, both references introduce a double well potential to circumvent this problem. Both references demonstrate the reinitialization in a narrow band region around the interface with such a double well potential. Far from the interface, the solution would converge to a constant function. However, as we will show in section 4.3.3, the double-well potential may flatten the level-set function in any region where the level-set gradient is overly low. In the context of a multiphysics solver, it is therefore advisable to execute the algorithm before such flat regions may occur.

4.2.2 Discretization

The starting point for the DG discretization of equation (4.10) is to multiply it by the test function v and to integrate the result over each cell individually and sum over the whole grid. The resulting terms are called consistency terms, since they give a discretization consistent with the original PDE. To achieve a stable method, the symmetry term in a_2, the penalty term a_3 are added. The result is the standard symmetric interior penalty method for the linear terms $a_1 \ldots a_3$ (see, e.g. Di Pietro and Ern (2012)). No additional terms are added for the nonlinear terms, thus we end up with the incomplete interior penalty method for the nonlinear terms b_1 and b_2. Last,

the penalty term a_4 enforces the boundary condition at the interface (see, e.g. Di Pietro and Ern (2012)):

find $\varphi \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K})$ s.t.

$$\underbrace{\int_{\Omega_h} \nabla\varphi \cdot \nabla v \,\mathrm{d}\vec{x}}_{a_1(\varphi,v),\text{ SIP volume term}} \underbrace{- \int_{\Gamma\setminus\partial\Omega} \left(\{\nabla\varphi\} \cdot [\![v]\!] + \{\nabla v\} \cdot [\![\varphi]\!]\right) \mathrm{d}S}_{a_2(\varphi,v),\text{ SIP consistency \& symmetry term}} +$$

$$\underbrace{\int_{\Gamma\setminus\partial\Omega} \eta\, [\![\varphi]\!] \cdot [\![v]\!] \,\mathrm{d}S}_{a_3(\varphi,v),\text{ SIP penalty term}} \underbrace{- \int_{\Omega_h} \left(1 - d\left(|\nabla\varphi|\right)\right) \nabla\varphi \cdot \nabla v \,\mathrm{d}\vec{x}}_{b_1(\varphi,v),\text{ nonlinear volume part}} +$$

$$\underbrace{\int_{\Gamma\setminus\partial\Omega} \left\{\left(1 - d\left(|\nabla\varphi|\right)\right) \nabla\varphi\right\} \cdot [\![v]\!] \,\mathrm{d}S}_{b_2(\varphi,v),\text{ nonlinear consistency part}} + \underbrace{\int_{\mathfrak{I}(\tilde{\varphi})} \alpha\varphi v \,\mathrm{d}S}_{a_4(\varphi,v),\text{ B.C. at Interface}} = 0$$

$$\forall v \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}) \quad (4.13)$$

In short, the above problem reads

find $\varphi \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K})$ s.t.

$$\sum_{i=1}^{4} a_i(\varphi, v) =: a(\varphi, v) = b(\varphi, v) := -\sum_{i=1}^{2} b_i(\varphi, v)$$

$$\forall v \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}). \quad (4.14)$$

Therein, the terms $a_1, \ldots, a_4$ are linear and symmetric, whereas the terms b_1 and b_2 are nonlinear and in general not symmetric. It should be noted, that the nonsymmetric discretization of the nonlinear part is known to deliver sub-optimal convergence rates of h^p, see e.g. Rivière (2008). The volume integrals a_1 and b_1 and the surface integrals on the cell edges a_2, a_3 and b_2 can be obtained using standard Gauß-quadrature, but the surface integral a_4 is only defined implicitly by the initial level-set field $\tilde{\varphi}$. Therefore, this integration is performed using the hierarchical moment fitting method by Müller et al. (2013), which is explained in more detail in 3.4.

4.2.3 Choice of the penalty parameter

The parameter η is the penalty parameter, which weakly enforces the inter-element continuity, and α the penalty for the boundary condition at the interface. Both values are chosen identical as determined by Hillewaert (2013):

$$\alpha = \eta = \max\{c_{K^+}, c_{K^-}\} \quad (4.15)$$

where

$$c_K = C(k)\frac{|\partial K|}{|K|}. \tag{4.16}$$

The numerical examples in this work, all consider quadrilaterals, for which Hillewaert (2013) gives the sharp estimate for the constant C depending on the polynomial degree k

$$C(k) \geq (k+1)^2. \tag{4.17}$$

For triangular ($D = 2$) and tetrahedral ($D = 3$) elements, Shahbazi (2005) gives the factor $C(k)$

$$C(k) \geq \frac{(k+1)(k+D)}{D}. \tag{4.18}$$

4.2.4 Iteration procedure

Following Basting and Kuzmin (2012), we solve this nonlinear system using the fix point iteration

$$a(\varphi^{n+1}, v) = b(\varphi^n, v) \quad \forall v \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}). \tag{4.19}$$

Since the resulting linear systems considered here are quite small, we use the direct solver PARDISO (see Schenk and Gärtner (2004)). For larger problems, one might exploit the symmetry of the system (4.19) by using a variant of the conjugate gradient algorithm.

4.3 Numerical test cases

After a discussion of suitable error measures for the reinitialization, we present three different setups for our reinitialization procedure: First, the focus is on the test case of a circular interface. We compare our method to the reference solution by Basting and Kuzmin (2012), examine the influence of the penalty parameter at the interface α and the convergence behaviour of the method. Second, we show how the choice of the initial conditions and the potential function affects the convergence of the scheme. Third, we show the convergence properties of the method on the example of a distorted droplet. Last, we show stability of the algorithm even for geometrical features that contain singularities and only span few grid cells.

4.3.1 Error measures

Suitable norms are needed to measure the quality of the reinitialization procedure. If the signed distance solution for a geometry is given analytically, one can employ the standard global norms

$$E_{L^2} = \|\varphi - \varphi_{\text{exact}}\|_2 = \left(\int_{\Omega_h} (\varphi - \varphi_{\text{exact}})^2 \, \mathrm{d}\vec{x} \right)^{1/2} \tag{4.20a}$$

$$E_{L^1} = \|\varphi - \varphi_{\text{exact}}\|_1 = \left(\int_{\Omega_h} |\varphi - \varphi_{\text{exact}}| \, \mathrm{d}\vec{x} \right) \tag{4.20b}$$

$$E_{L^\infty} = \|\varphi - \varphi_{\text{exact}}\|_\infty = \max_{\vec{x} \in \Omega_h} |\varphi(\vec{x}) - \varphi_{\text{exact}}(\vec{x})|. \tag{4.20c}$$

It is well known that convergence in the L_2-norm breaks down to first order if the analytical solution contains singularities. If the interest lies in the accuracy of the method, the error calculation may be restricted to a narrow-band region around the interface, which does not include the singularity. In this part of the domain, one might expect higher order accuracy. We choose a different approach: In the test cases 4.3.2.3 and 4.3.4 we limit the computational domain, such that possible singularities lie outside the domain and cannot affect the accuracy of the method. Later we demonstrate the stability of the method for cases, where the singularities lie in computational cells next to the interface.

If no analytical solution is available, one might consider different options:
For two-phase flow simulations, the main goal of reinitialization is to have an approximate signed distance function while maintaining the shape of the interface.

A measure for the deviation of the level-set from its original position is the approach employed by Basting and Kuzmin (2012), who integrate the new level-set field over the original contour:

$$E_c = \left(\int_{\mathfrak{I}(\tilde{\varphi})} \varphi^2 ds \right)^{\frac{1}{2}} \tag{4.21}$$

If the level-set shifts its position, it will have a nonzero value at the point, where it was before. Since the level-set has signed distance properties after the reinitialization, this value is a measure for the spatial deviation from the original position.

Last, the deviation from the signed distance field can be measured by

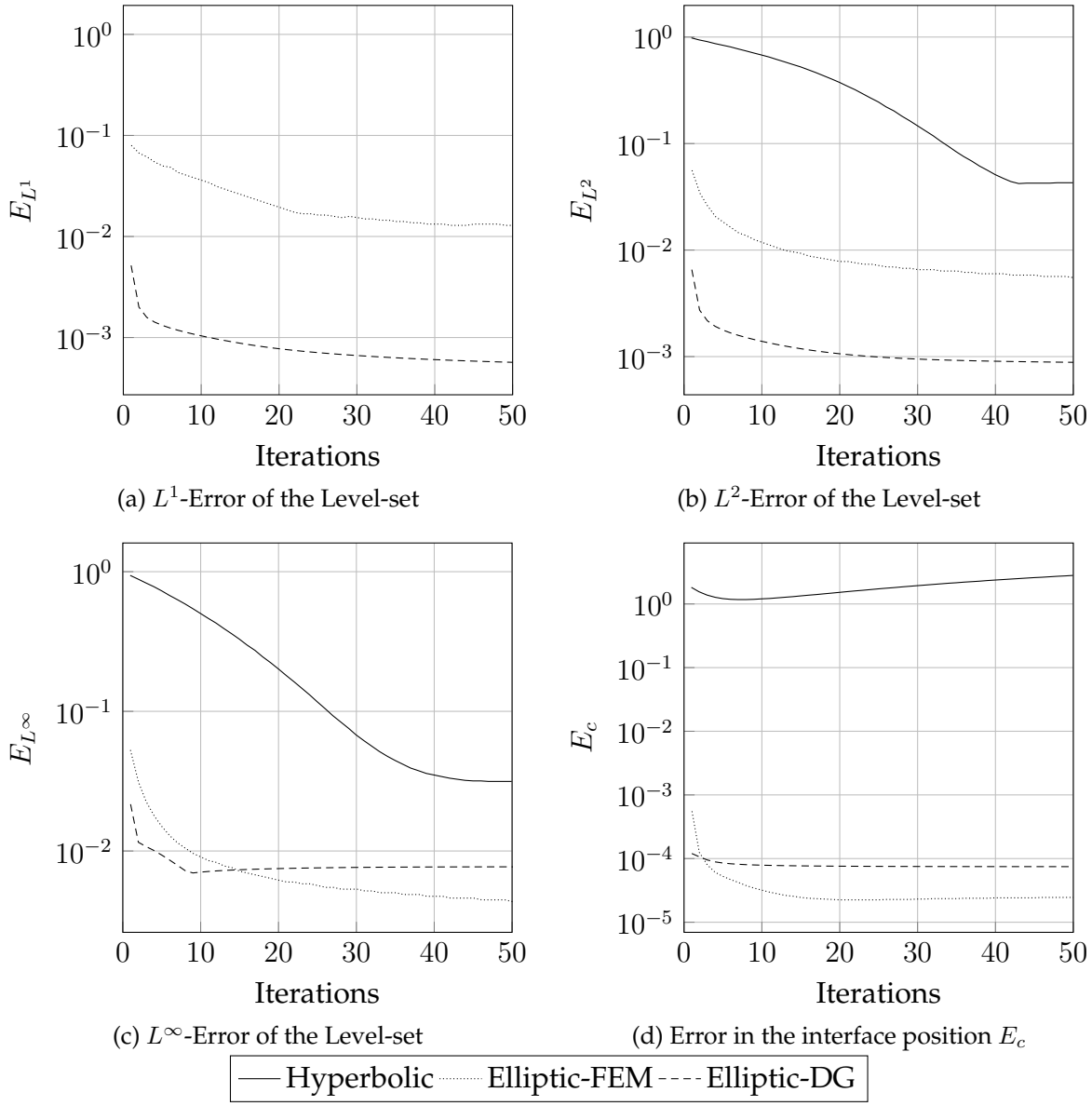
$$E_{\text{SD}} = \|(1 - |\nabla\varphi|)\|_2 = \left(\int_{\Omega_h} (1 - |\nabla\varphi|)^2 \, \mathrm{d}\vec{x} \right)^{\frac{1}{2}}. \tag{4.22}$$

Note, that this norm does not measure deviations in the interface position.

It is important to note, that only the integral error measures E_{SD} (4.22) and E_{L^1}, E_{L^2} and E_{L^∞} (4.20) are norms of the level-set φ. Thus, only in these norms, convergence rates can be expected.

4.3.2 Circular interface

4.3.2.1 Comparison with reference solution



(a) L^1-Error of the Level-set

(b) L^2-Error of the Level-set

(c) L^∞-Error of the Level-set

(d) Error in the interface position E_c

Figure 4.3: Comparison of PDE based approaches, data for hyperbolic and FEM implementation of elliptic reinitialization from Basting and Kuzmin (2012)

As a first test case, we compare our method, with the reference by Basting and Kuzmin (2012). The test case consists of a circular interface with a radius of 0.5, which is located centered in the domain $\Omega = (0, 1) \times (0, 1)$. The signed distance profile for this circle is

$$\varphi_{\text{exact}}(\vec{x}) = 0.25 - \sqrt{(x - 0.5)^2 + (y - 0.5)^2}. \tag{4.23}$$

We use the same initial condition and discretization as in the reference: The computational domain is subdivided by regular refinement into triangles of equal size. By repeating this four times, we end up with 32 elements per side. Linear polynomials (i.e. $p = 1$) are chosen. The potential function used here is the single-well potential ψ_1. The level-set is initialized as $\tilde{\varphi} = 2\varphi_{\text{exact}}$ and stochastically distorted with a maximal amplitude of $1/2h = 1/64$ on all cells which do not contain the interface. Figure 4.3 compares the results of the FEM implementation of the elliptic reinitialization by Basting and Kuzmin (2012), their implementation of a hyperbolic procedure and our method in multiple error measures.

The hyperbolic reinitialization produces much higher errors in all norms, compared to both elliptic procedures. Looking at the global E_{L^1} and E_{L^2} norms, the DG variant produces significantly lower errors than the FEM implementation. However, the error in the E_{L^∞} norm is slightly higher than in the FEM case. This might be due to the coupling terms in the gradient across the boundaries of the computational cells, which penalize strong variations in the gradient across cells. While the original formulation of basting and Kuzmin Basting and Kuzmin (2012) does not include these DG-specific coupling terms , the term a_2 in equation (4.13) leads to a smoothing of kinks at cell edges, which causes a larger error at the singularity in the center of the circle. The error at the interface of the DG implementation is slightly larger than for the FEM, while still way below the hyperbolic variant. The main reason for this behaviour is the weak enforcement of the boundary condition at the interface, which heavily depends on the choice of the penalty parameter. This error can be reduced by increasing the penalty parameter, at the cost of an increased condition number of the resulting linear systems, as we demonstrate in the following section 4.3.2.2.

4.3.2.2 Influence of the penalty parameter

To test the influence of the penalty parameter at the interface, we vary this parameter between 10^1 and 10^7, see figure 4.4. As a test case, we used the same case of a circle on a triangular grid as before. While the penalty factor has a large impact on the error of the zero contour, it only influences the global measures E_{L^1}, E_{L^2} and E_{L^∞}, if the penalty factor at the interface is smaller than the calculated value. Shahbazi (2005) derived the minimal penalty to achieve convergence in these global norms. We choose $\alpha = \eta$ for three different reasons: First, the global error norms do not change with an increase in the penalty. Second, while the condition number of the operator $a(u, v)$ is relatively constant for $\alpha < \eta$, it rises significantly for $\alpha > \eta$ which may lead to higher computational cost for solving the linear systems by an iterative solver.

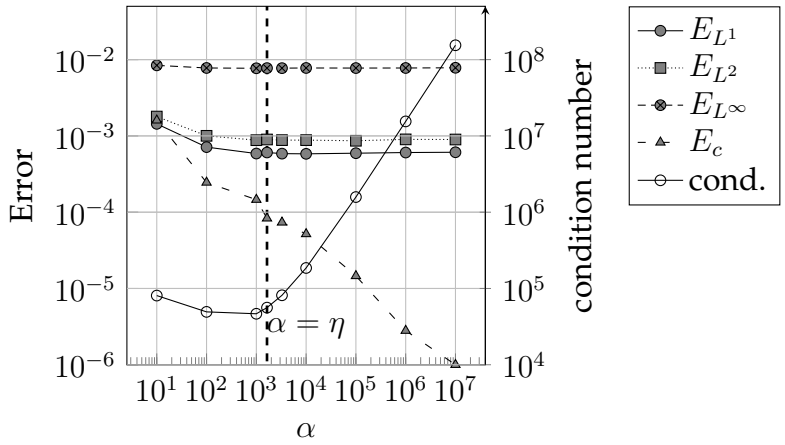


Figure 4.4: Error in different measures and condition number of the associated linear system for varying penalty parameter α

Third, while the error in the contour E_c drops with increasing α, this does not alter the h-convergence behavior of this norm.

4.3.2.3 Convergence behaviour

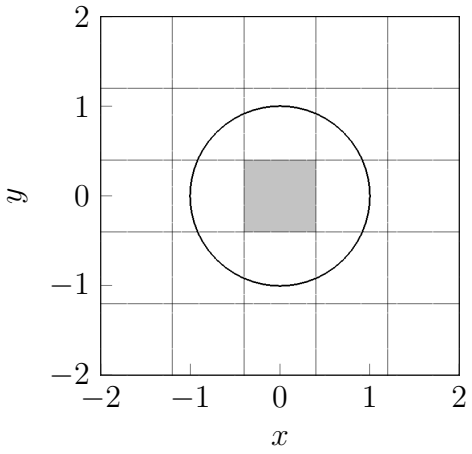


Figure 4.5: Coarsest grid for convergence tests on a circular interface. The grey cell is not part of the domain.

As a third test case for the circular interface, we look at h-convergence for polynomial degrees k up to 5. The computational domain $\Omega_h = (-2, 2)^2 \setminus (-0.4, 0.4)^2$ is discretized by a structured quadrilateral grid. The coarsest grid for this domain is shown in figure 4.5 . It consists of 5×5 cells: The center cell is not a part of the domain, thus avoiding that the singularity there spoils the convergence rate. The grids used for the convergence study are obtained by subdividing the grid from figure 4.5equidistantly. In contrast to the previous examples, we use the double-well potential ψ_2, since this potential performs better in the vicinity of singularities, as we will see later in section 4.3.3. As the initial condition we use the quadratic representation of the circle

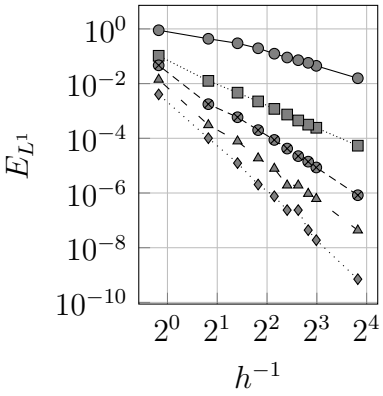
$$\tilde{\varphi} = x^2 + y^2 - 1. \tag{4.24}$$

Figure 4.6 compares the results to the analytical solution

$$\varphi_{\text{exact}} = \sqrt{x^2 + y^2} - 1. \tag{4.25}$$

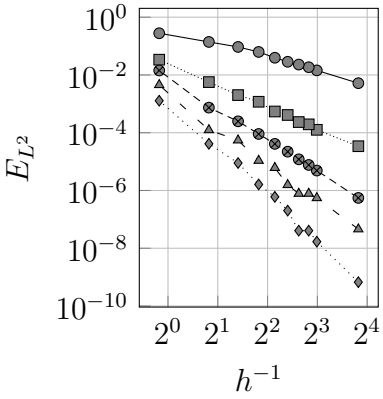
in figure . In the global norms, the convergence rates rise with the polynomial degree: E_{L^1} converges with approximately $h^{k+1/2}$, E_{L^2} with approximately h^k. This suboptimal convergence behaviour is to be expected due to the use of the incomplete interior penalty discretization for the nonlinear term Rivière (2008). The signed-distance property converges with a convergence rate larger than h^{k+2}. The high convergence rate of E_{SD} might be explained by the relatively simple geometry of this test case - for the more complex geometry in section 4.3.4 the convergence rate is h^{k+2}. The shift of the level-set contour shows a different behaviour: while beyond $h^{-1} > 2^3$, the error converges with approximately h^k, we see a lower error bound for polynomial degrees of $k = 4$ and $k = 5$. This lower error bound might be explained by the approximate integration rules used for the boundary condition and the calculation of the error measure itself. Nevertheless, we will not further investigate this, because we show in

section 4.3.4, that this error measure is limited to first order accuracy for practical test cases.



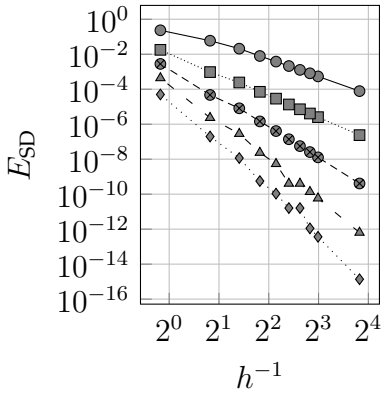
P	EOC
1	1.47
2	2.72
3	3.84
4	4.47
5	5.62

(a) Global E_{L^1} of the level-set field



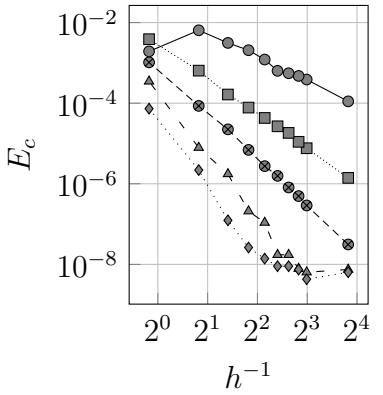
P	EOC
1	1.45
2	2.49
3	3.58
4	4.03
5	5.2

(b) Global E_{L^2} of the level-set field



P	EOC
1	2.95
2	4.03
3	5.63
4	7.28
5	8.78

(c) Deviation from signed Distance E_{SD}



P	EOC
1	1.26
2	2.85
3	3.76
4	5.02
5	4.48

(d) Shift of the level-set contour, E_c

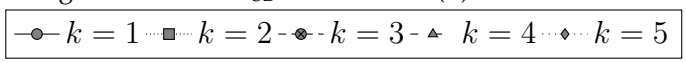


Figure 4.6: Results of the h-convergence study for the circular interface

4.3.3 Flat interface

As a second setup, we consider a level-set, which has two zero contours

$$\tilde{\varphi}_1 = 4 - (x - 0.5)^2, \tag{4.26}$$

see figure 4.7. The analytical solution for the reinitialization of this function is

$$\varphi_{\text{a}} = \begin{cases} 2 + (x - 0.5) & \text{if } x \leq 0.5 \\ 2 - (x - 0.5) & \text{if } x > 0.5 \end{cases} \tag{4.27}$$

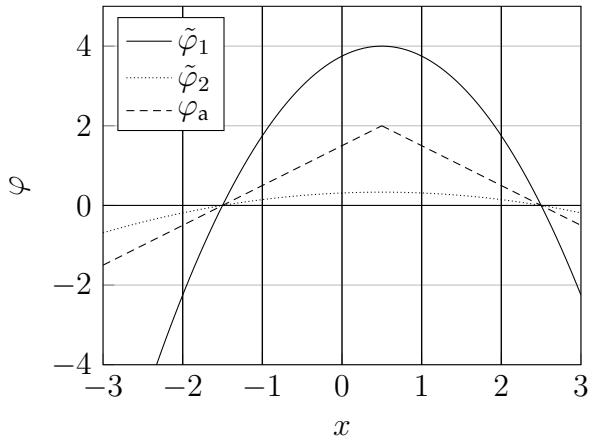


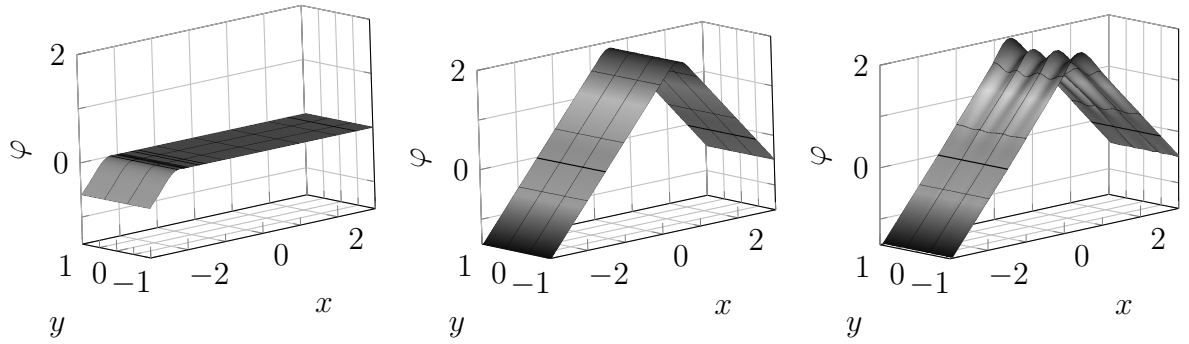
Figure 4.7: Initial level-sets $\tilde{\varphi}_1$ and $\tilde{\varphi}_2$ and analytical solution φ_a. The vertical lines coincide with the numerical grid.

Scaling $\tilde{\varphi}_1$ by an arbitrary factor γ smaller than 1 does not change the analytical solution, but may affect the numerical solution depending on the chosen potential ψ. For the numerical testcase we choose $\gamma = 1/12$.

$$\tilde{\varphi}_2 = \gamma \tilde{\varphi}_1 \tag{4.28}$$

For this numerical test case, we use a grid of $[6 \times 3]$ elements spanning the domain $[-3, 3] \times [-1, 1]$. The initial contours $\tilde{\varphi}_1$ and $\tilde{\varphi}_2$, the analytical result $\varphi_{\mathbf{a}}$, and the partition of the domain in x-direction are shown in figure 4.7 .

Figure 4.8 shows three different computations for the initial conditions shown in figure 4.7 . The convergence criterion is $\|u^n - u^{n-1}\| \leq 5 * 10^{-12}$, the polynomial degree is $k = 5$. The first figure 4.8a shows the converged result for the initial condition $\tilde{\varphi}_2$. Almost in the whole domain, the solution is flattened out to a constant value of $\varphi = 0$. This shows the drawback of the double-well potential ψ_2: since the gradient in the initial condition is much closer to zero than it is to one, see figure 4.7, the gradient in the converged solution is zero, which causes the flattening of the solution. The third figure 4.8c shows the same test case with the potential function ψ_1. Since this potential function allows only $|\nabla\varphi| = 1$, the solution starts to oscillate in the vicinity of the singularity. These oscillations spread out from the affected cell and pollute the whole solution. The only case with a satisfactory result is the figure 4.8b, which shows the result for the initial condition $\tilde{\varphi}_1$ and the double-well potential ψ_2: we achieve good agreement with the signed-distance solution around the interface. In the center cell, the singularity is smoothed out, without oscillations and pollution of the nearby cells. Therefore, we will use the double-well potential for all cases, where singularities might occur. To avoid this behaviour, section 4.4 introduces a preconditioning technique.



(a) ψ_2, $\tilde{\varphi} = \tilde{\varphi}_2$,converged after 213 iteration steps

(b) ψ_2, $\tilde{\varphi} = \tilde{\varphi}_1$, converged after 97 iteration steps

(c) ψ_1, $\tilde{\varphi} = \tilde{\varphi}_2$, aborted after 1000 iteration steps

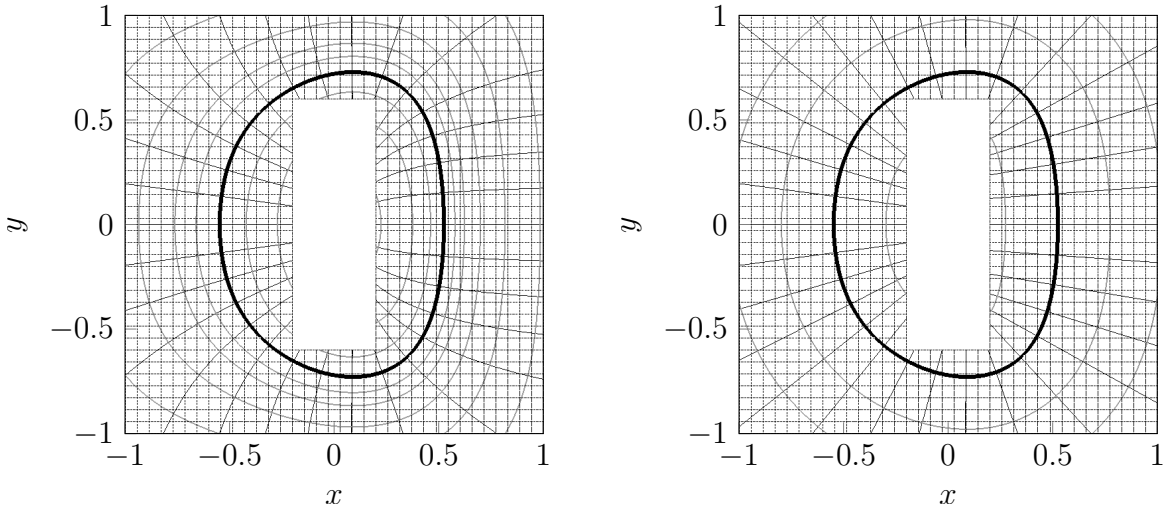
Figure 4.8: Solution for different potential function in the vicinity of a singularity. Polynomial degree $k = 5$, thick black line $\varphi = 0$, the figure is colored according to the value of φ

4.3.4 Distorted circle

In multiphase flows, one often faces bubbles, which deviate from a spherical shape. Such a shape might be described by the initial level-set

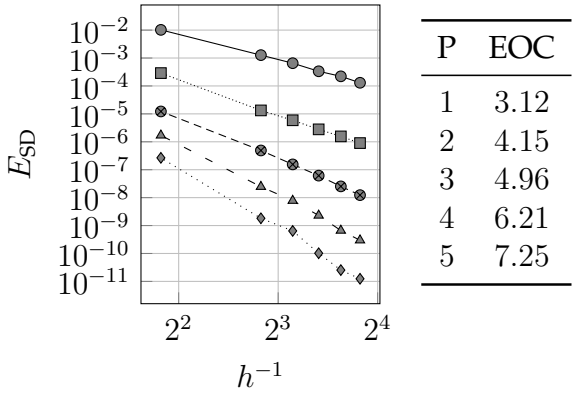
$$\tilde{\varphi}_0 = (x + 0.5)^4 + y^4 + (x - 0.5)^2 + y^2 - 1.1. \tag{4.29}$$

Figure 4.9 shows the contour of the interface, isolines of the level-set and streamlines following the level-set gradient for both the initial condition and the converged solution. In contrast to the previous test cases, isolines of the level-set function are not concentric, see figure 4.9a. For this test case, the computational domain was chosen as $\Omega_h = (-1, 1)^2 \setminus (-0.2, 0.2) \times (-0.6, 0.6)$. Like this, the singularity in the middle of the contour lies outside the computational domain and cannot spoil the accuracy of the calculation. The potential function is chosen as the double-well ψ_2. In the context of multiphase flows, this is usually a justified approximation. Since the level-set is needed only within a narrow region around the interface and the surface curvature radius is larger than the grid size, the singularities in the level-set can be excluded. However, we will show in section 4.3.5, that our method is stable for this case, i.e. when dealing with level set features of the same length-scale as the curvature radius. For this test case no analytical solution is available. Therefore, we can only measure the deviation in the absolute Gradient E_{SD}, eq. (4.22), and from the zero contour E_C, eq. (4.21). Figure 4.10 shows the h-convergence for polynomial degrees from $k = 1$ to $k = 5$. While the level-set gradient converges with a rate of $E_{\text{SD}} \propto h^{k+2}$, which is even higher than expected for norms of the gradient h^k, the contour error does not give high order accuracy and converges with first order $E_c \propto h$.

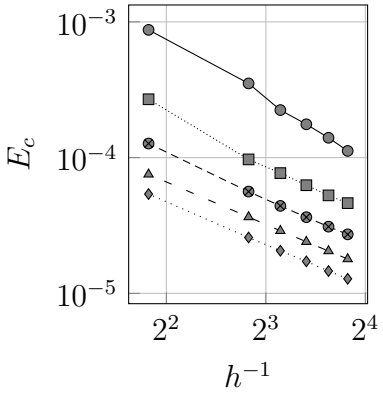


(a) Initial Conditions (b) After Reinit

Figure 4.9: Distorted circle on a grid $[40 \times 40]$ showing level-set isolines (zero isocontour in bold line) and streamlines of the gradient (extending outwards), polynomial degree $k = 5$



P	EOC
1	3.12
2	4.15
3	4.96
4	6.21
5	7.25



P	EOC
1	1.61
2	1.58
3	1.22
4	1.28
5	1.08

(a) Deviation from signed distance (b) Shift of the level-set contour

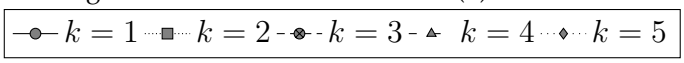


Figure 4.10: Results of the h-convergence study for the distorted circle test case

4.3.5 Ellipse

Last, we consider a very coarse discretization of an ellipsoidal interface in the domain $\Omega = [-2, 2]^2$ with the initial level-set

$$\varphi = (x\ (1 + 0.3))^2 + (y\ (1 - 0.3))^2 - 1. \tag{4.30}$$

We discretize this domain with a grid of only 5×5 grid cells, see figure 4.13. To avoid oscillations due to the singularity, we choose the potential function as the double-well potential ψ_2. After the first iteration, the level-set function already closely approaches the exact solution, as shown in the cross-sectional cut, see figure 4.11. Even on this coarse grid, the initial interface is not moved by the reinitialization procedure, as depicted in figures 4.11 and 4.13. Looking at figure 4.12, the method shows long term stability.

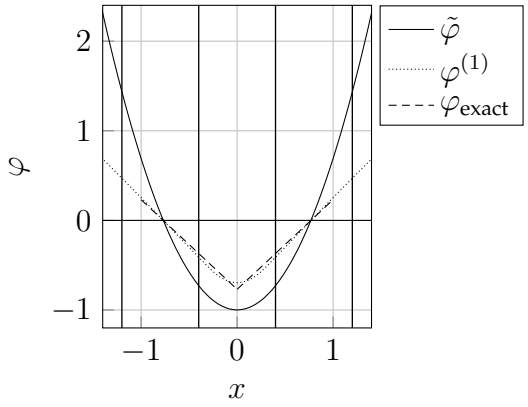


Figure 4.11: Cross-section of the level-set $\varphi^{(1)}$ at $y = 0$ after the first iteration. The vertical lines coincide with the numerical grid. Analytical solution φ_{exact} and the initial conditions $\tilde{\varphi}$ shown for comparison

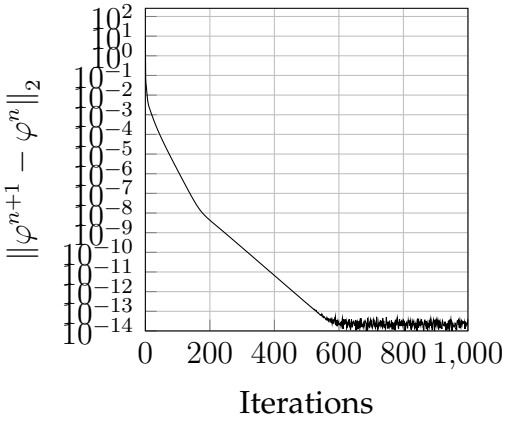
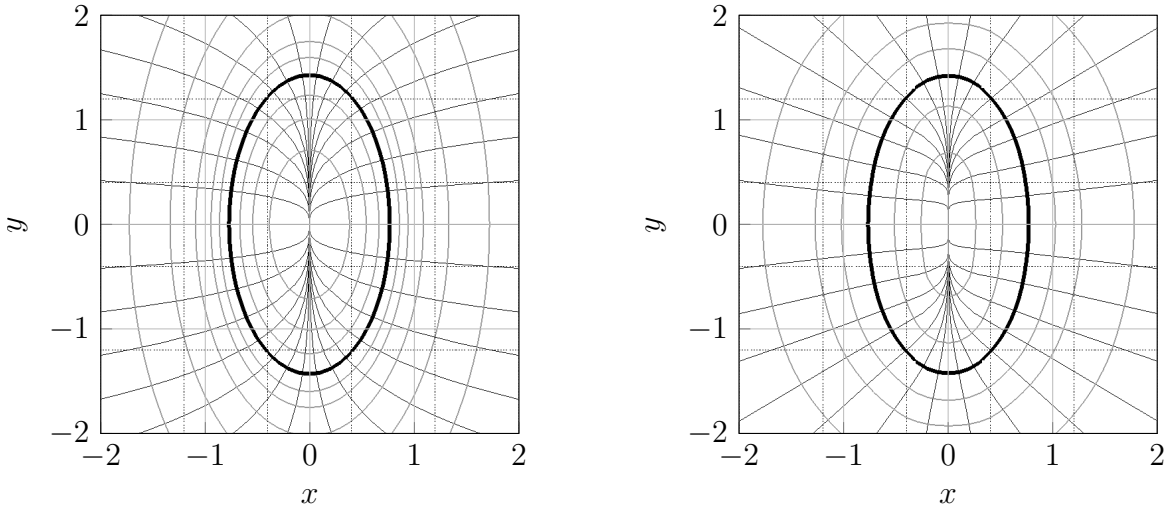


Figure 4.12: L_2 change rate of the level-set field



(a) Initial condition

(b) After reinitialization

Figure 4.13: Ellipse on a coarse grid showing level-set isolines and outwards extending normal vector , polynomial degree $k = 3$

4.3.6 Torus

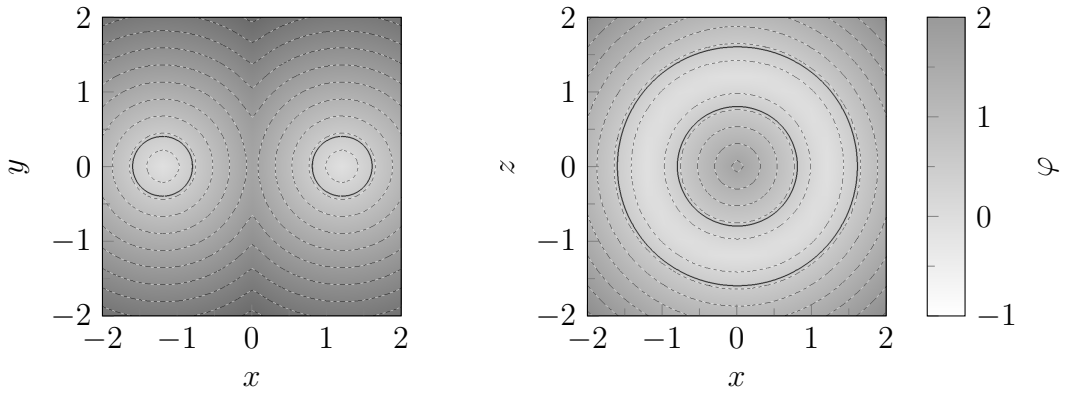
As an additional testcase to Utz et al. (2017b) we look at the reinitialization of a three dimensional contour: One possibility to describe the surface of a torus is via the level set.

$$\varphi(x, y, z) = 5\left(z^2 + \left(\sqrt{x^2 + y^2} - R\right)^2 - r^2\right) \tag{4.31}$$

The signed distance solution of this torus is

$$\varphi(x, y, z) = \sqrt{z^2 + \left(\sqrt{x^2 + y^2} - R\right)^2} - r \tag{4.32}$$

We initialize the level set using equation 4.31 on the domain $\Omega = (-2, 2)^3$ using a equidistant grid of 20^3 cells and a polynomial degree of $k = 2$. The geometry parameters are chosen as $r = 0.4$ and $R = 1.2$. This test case converges to the analytical solution 4.32, even with the curved singularity at the center of the small radius, as can be seen in the two sections 4.14.



(a) Section at $z = 0$ (b) Section at $y = 0$

Figure 4.14: Reinitialization of the level set function for a torus, solid lines: $\varphi = 0$, dashed lines: $\varphi =$ const.

4.4 Preconditioning

In addition to the publication Utz et al. (2017b), a preconditioning to the method is proposed: As shown in the previous sections, the reinitialization method presented here delivers accurate and stable results for the second potential ψ_2, when the gradient is sufficiently large. However, one can construct two kinds of initial conditions, which cause a divergence of the algorithm. The first case is, when the gradient is small, which leads to a flattening of the level set as shown in section 4.3.3. The second case is an oscillation of the initial conditions, where the level-set gradient switches its sign. Then, as shown in the introductory example, 4.1, both variants of the method shown here do not converge to the exact solution, see figure 4.15.

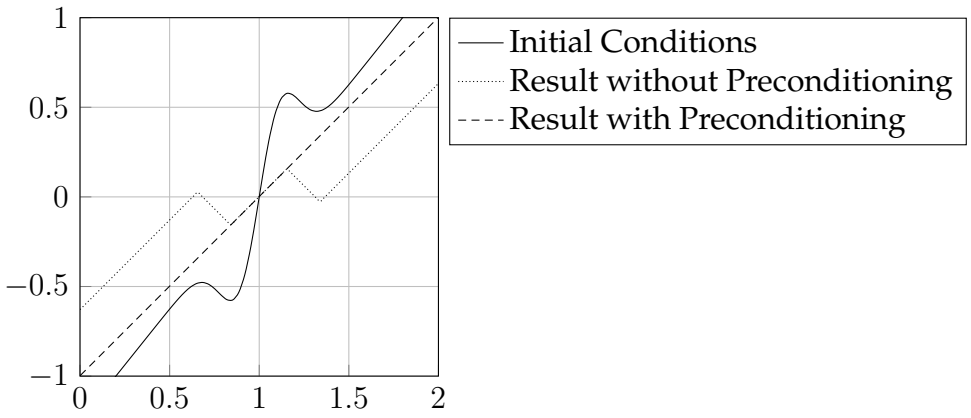


Figure 4.15: Reinitialization of oscillating initial conditions with and without preconditioning

4.4.1 Coupling of fast marching and elliptic reinitialization

To circumvent this issue, a three-step method for the reinitialization is introduced:

1. Reinitialization of cells, which include the interface $\mathfrak{I}$ using the single well potential ψ_1. I.e. $K \in \mathfrak{K} : \|K \cap \mathfrak{I}\| \neq 0$. This avoids that the level set function becomes flat, as shown in figure 4.8a. Then all cut cells have signed distance property.

2. Reinitialization of all other cells $K \in \mathfrak{K} : \|K \cap \mathfrak{I}\| = 0$ using a fast marching procedure (see e.g. Adalsteinsson and Sethian (1995)). This gives a first-order approximation on the actual solution. However, the procedure ensures, that the sign of the gradient is correct and oscillations as shown in figure 4.15 are removed.

3. Perform the algorithm described above with the potential function ψ_2. This ensures stable results in the vicinity of singularities. An unintentional flattening of the interface is avoided by the preceding fast marching step.

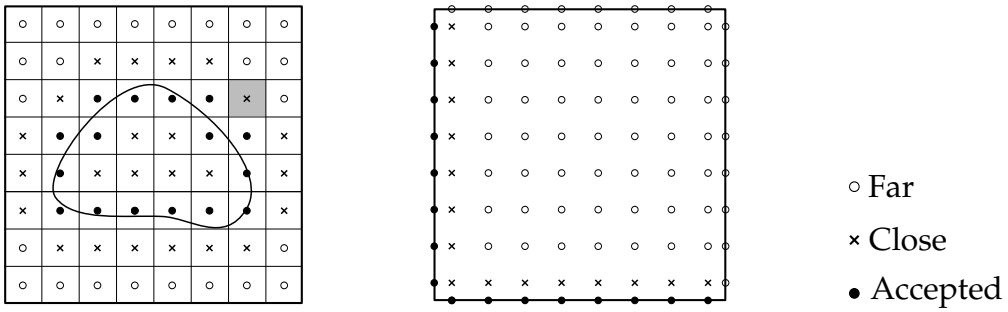
pterThe fast marching procedure by Adalsteinsson and Sethian (1995) is based on separating points on a grid into three groups, see figure 4.16: "Accepted" points, which are points, where the level set is already a solution of the Eikonal equation (2.12), "Close" points, which are adjacent to the accepted points and which are recalculated in the next loop of the marching procedure and "Far" points, which have not been calculated yet.

Each loop of the fast marching procedure consists of the following steps:

1. Recalculate the close points by an upwind finite difference discretization of the Eikonal equation (2.12)

2. Mark the point with the smallest value from "Close" as "Accepted"

3. Move all points which are adjacent to the smallest value from "Far" to "Close"



(a) Selection of the cell

(b) Fast marching on the individual cell

Figure 4.16: Reinitialization using fast marching for the whole domain and individual cells, the sub cell grid on the cell is depicted for the cell marked in gray

4. repeat loop

In the DG solver presented here, this process is implemented using a multi grid approach: On the coarse level, each cell in the grid is treated as an individual data point with the cell average being the value of this data point, see figure 4.16a. This coarse step determines, which cell is to be recomputed next. On this cell, a new grid is defined, see figure 4.16b. After computing this cell-based grid, its solution is L^2 projected onto the DG polynomials of the level-set.

4.4.2 One dimensional test case

Figure 4.15 shows a one-dimensional test case on the domain $\Omega = (0, 2)$ for an initial level set function

$$\varphi = \sin(5x)) \exp(-(5x))^2) + \frac{5}{4}x. \tag{4.33}$$

The analytical solutions for this initial condition is the straight line through the origin

$$\varphi_{\text{exact}} = x. \tag{4.34}$$

Note, that without preconditioning the calculation does not only converge to a result, which is not the signed distance solution, but also this incorrect solution intersects the $\varphi = 0$ axis in additional positions. This behavior is especially bad in the context of multiphase simulations, where these intersection points mean the emergence of so called spurious bubble, i.e. bubbles appear and disappear in unphysical locations. With the preconditioning algorithm however, this unwanted behavior is avoided altogether and the calculation converges to the exact solution.

4.4.3 Two dimensional test case

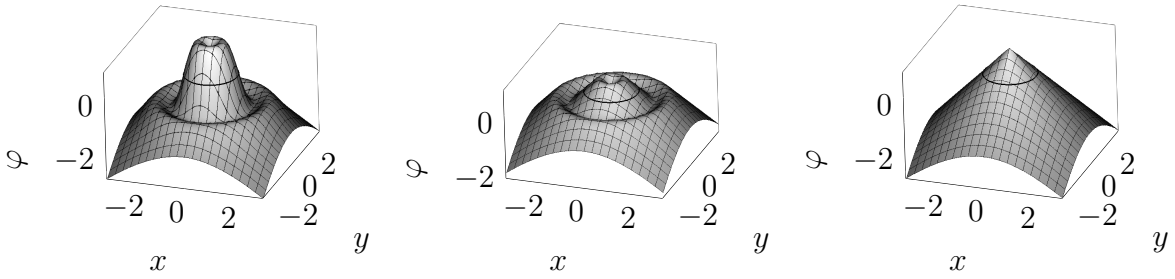
Figure 4.17 shows the results for this preconditioning technique for a two dimensional test case. On the domain $\Omega = (-3, 3)^2$ using a equidistant grid of 20^2 cells, the level set is initialized as

$$\varphi(r, \theta) = 1 - r - \sin(3(r-1)) \exp(-(r-1)^2) \tag{4.35}$$

in cylindrical coordinates, which is similar to the previous test case in one dimension. The analytical solution is

$$\varphi_{\text{exact}} = 1 - r = 1 - \sqrt{x^2 + y^2} \tag{4.36}$$

The polynomial degree is chosen as $k = 2$. The results are similar to the one dimensional case: without preconditioning, the level set retains the wrong gradient direction as shown in figure 4.17b, in this two dimensional case, the solution even diverges after several iterations. With preconditioning, the algorithm approximates the exact solution well, as shown in figure 4.17c.



(a) Initial Conditions

(b) First iteration without preconditioning, result does not converge

(c) Result with preconditioning

Figure 4.17: Reinitialization of a level set with oscillating initial conditions

5 The Extension Problem

This chapter covers the extension of a quantity from the interface into the domain using an elliptic PDE. Main parts of this chapter are rewritten versions of the publication Utz and Kummer (2017) by the author, and has been slightly altered and extended since the publication of the paper.

5.1 The extension problem in multiphysics simulations

The problem of extending a quantity from a sub-manifold into the domain in which it is embedded arises in many computational physics applications. A prominent example is the simulation of moving interfaces using an interface capturing method, such as the level set method. Especially when dealing with non-material interfaces, the motion of the interface does not coincide with the bulk velocity at the interface itself. Prominent examples of such problems are the simulation of the Stefan problem (see Chen et al. (1997); Chessa et al. (2002); Gibou et al. (2002, 2003) and Javierre et al. (2005)), multiphase flows with and without phase transition (see Zoby et al. (2012) and McCaslin and Desjardins (2014)) and mean curvature flows (see Adalsteinsson and Sethian (1999); Peng et al. (1999) and Schulte (2012)). Another case, where such an extension is needed, is the solution of surface equations by an embedding approach as in Kallendorf (2017).

In this article, we focus on methods which use an implicit surface description by a level set method. This method describes a surface $\mathfrak{I}$ via the zero-isocontour of a sufficiently smooth function φ, which separates the domain $\Omega \subset \mathbb{R}^D$ into two parts $\mathfrak{A}$ and $\mathfrak{B}$. Revisiting chapter 2 the level set is usually chosen as the signed distance to the interface, see equation (2.10). This can be rewritten by defining the closest point $\vec{x}_c$

$$\varphi(\vec{x}) = \begin{cases} |\vec{x}_c(x) - \vec{x}| & \text{in } \mathfrak{A} \\ -\ |\vec{x}_c(x) - \vec{x}| & \text{in } \mathfrak{B} \\ 0 & \text{on } \mathfrak{I}. \end{cases} \tag{5.1}$$

where $\vec{x}_c \in \mathfrak{I}$ is the closest point on the surface for every point $\vec{x} \in \Omega$ in the domain, i.e.

$$\vec{x}_c(\vec{x}) = \arg\min_{\vec{x}_c \in \mathfrak{I}} (\ |\vec{x}_c - \vec{x}|). \tag{5.2}$$

With such a level set function, we choose the extension field u at any point to be the value at the closest point on the interface, i.e.

$$u(\vec{x}) = u_0\left(\vec{x}_c(x)\right). \tag{5.3}$$

This is equivalent to the solution of the PDE:

$$\nabla\varphi \cdot \nabla u = 0 \qquad \text{in } \Omega, \tag{5.4a}$$
$$u = u_0 \qquad \text{on } \mathfrak{I}. \tag{5.4b}$$

This problem is well posed for closed interfaces. However, when the interface intersects the boundary as shown in figure 5.1 , one has to specify additional Dirichlet boundary conditions there

$$u = u_{\mathrm{D}} \quad \text{on } \{\partial\Omega \mid \varphi\nabla\varphi \cdot \vec{n}_\Omega > 0\}, \tag{5.4c}$$

since the characteristics in the area marked in gray do not originate from the interface.

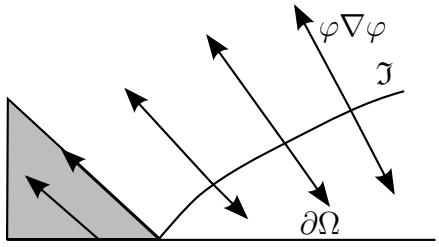


Figure 5.1: Additional boundary conditions, when the interface and the boundary intersect. The problematic region is shown in grey. The arrows depict the direction in which information propagates from the interface into the domain.

One possibility of solving the extension problem (5.4) is to solve equation (5.3) directly by using some kind of search algorithm, see e.g. Zoby et al. (2012); Gibou and Fedkiw (2005).

Another option is to interpret (5.4) as the steady state limit of an advection equation

$$\partial_t u + \nabla\varphi \cdot \nabla u = 0 \qquad \text{in } \Omega \tag{5.5a}$$
$$u = u_0 \qquad \text{on } \mathfrak{I}, \tag{5.5b}$$

which is solved using a suitable time stepping scheme. This idea has been applied in the context of finite difference schemes by multiple authors, see e.g. Chen et al. (1997); Peng et al. (1999); Gibou et al. (2002, 2003), in the context of high-order discontinuous Galerkin methods such a method would require a shock limiting scheme, which are available for general advection problems, see e.g. Persson and Peraire (2006); Klöckner et al. (2011) and level-set reinitialization Karakus et al. (2016), but have not been applied to extension problems yet.

However, most established methods directly seek for a solution of equation (5.4). Using continuous finite elements, equation (5.4) can be solved by a Galerkin Least-Squares approach Chessa et al. (2002); Sauerland (2013); Pawel Stapór (2016).

In the context of second-order methods, the most popular algorithms are the fast marching Adalsteinsson and Sethian (1999); Chopp (2000); Frolkovic et al. (2015) and the fast sweeping methods Aslam (2004); Frolkovic et al. (2015). While these methods are unparalleled in terms of speed for low order methods, in the context of discontinuous Galerkin methods, they require cell-local solver. For level-set reinitialization, such solvers up to order three Zhang et al. (2011); Wu and Zhang (2014) have been published, but to the authors' knowledge no such solver is available for the extension problem.

In this article, we choose another PDE-based approach, which may serve as the basis of a marching or sweeping algorithm. The starting point is to reformulate (5.4) into an ellipitc PDE following the same idea as for the reinitialization problem presented in chapter 4. While the Eikonal equation for level-set reinitialization leads to a non-linear elliptic PDE, this approach for the extension problem results in an PDE for anisotropic diffusion. Using anisotropic diffusion for the extension problem is briefly mentioned, but not examined further in the master's thesis of Schulte (2012) in the context of a continuous FEM. After introducing this reformulation of the problem into an elliptic PDE, we give an upwind discretization of the resulting PDE. This gives high-order accuracy even in the presence of singularities, which are necessarily present in concave regions of the level set. Last, we demonstrate the properties of our scheme for several test cases. Depending on the level set field, the problem might be ill-posed. Therefore, section 5.5 introduces an additional isotropic viscosity, which stabilizes the method even for such ill-posed problems.

Note that there are other possibilities than (5.4) to obtain a smooth extension of a quantity from the interface: In the context of finite difference methods Aslam (2004); Aslam et al. (2014) introduce an extension procedure for multidimensional extrapolation, which requires solving multiple equations similar to (5.4) involving higher-order derivatives of u. Moroney et al. (2017) introduce a finite difference scheme which completely avoids the dependence on the level-set gradient by solving a biharmonic equation. This requires solving a globally coupled system but allows the incorporation of boundary conditions.

5.2 Reformulation of the extension problem as an elliptic PDE

In contrast to Utz and Kummer (2017), the problem is slightly modified. Instead of solving equation (5.4), the gradient of the level set function is normalized $\frac{\nabla\varphi}{|\nabla\varphi|} = \widehat{\nabla\varphi}$, which is the expression for the interface normal vector $\widehat{\nabla\varphi}|_\mathfrak{I} = \vec{n}_\mathfrak{I}$. This yields the PDE

$$\widehat{\nabla\varphi} \cdot \nabla u = 0. \tag{5.6}$$

Note, that solutions to this equations are also solutions to the original problem (5.4). However, in regions, where $|\nabla\varphi|$ becomes small, the problem is better conditioned since $|\widehat{\nabla\varphi}| = 1$ and thus leads to more stable results. As in the previous chapter, the problem (5.6) is reformulated into the variational minimization problem

$$\min R(u) = \int_\Omega \psi(\widehat{\nabla\varphi} \cdot \nabla u)\,\mathrm{d}V \tag{5.7a}$$

$$\text{s.t. } u = u_0 \quad \text{on } \mathfrak{I}, \tag{5.7b}$$

where ψ is a smooth potential with a minimum at $\psi(0)$. Introducing the Gâteaux derivate $\partial R(u)[v]$ and a Lagrange multiplier λ, the solution of this minimization problem is the solution of the variational problem

$$\text{find } u \text{ s.t. } \partial R(u)[v] - \lambda \oint_\mathfrak{I} (u - u_0) v\,\mathrm{d}S = 0 \quad \forall v, \tag{5.8}$$

where

$$\partial R(u)[v] = \int_\Omega \nabla v \cdot \frac{\partial \psi(\widehat{\nabla\varphi} \cdot \nabla u)}{\partial(\nabla u)}\,\mathrm{d}V = \int_\Omega \nabla v \cdot \widehat{\nabla\varphi}\, \psi'(\widehat{\nabla\varphi} \cdot \nabla u)\,\mathrm{d}V. \tag{5.9}$$

Taking the simplest choice for ψ

$$\psi(s) = s^2/2 \tag{5.10}$$

gives

$$\partial R(u)[v] = \int_\Omega \nabla v \cdot \widehat{\nabla\varphi} \left(\widehat{\nabla\varphi} \cdot \nabla u\right)\,\mathrm{d}V, \tag{5.11}$$

which can be rewritten by partial integration as

$$\partial R(u)[v] = -\int_\Omega \nabla \cdot \left(\widehat{\nabla\varphi} \cdot \nabla u \widehat{\nabla\varphi}\right) v\,\mathrm{d}V + \int_{\partial\Omega} \left(\widehat{\nabla\varphi} \cdot \nabla u \widehat{\nabla\varphi}\right) \cdot \vec{n} v\,\mathrm{d}S. \tag{5.12}$$

Including the constraint (5.7b) and (5.4c) via a Lagrange multiplier λ gives

$$-\int_\Omega \nabla \cdot \left(\left(\widehat{\nabla\varphi} \otimes \widehat{\nabla\varphi}\right) \nabla u\right) v\,\mathrm{d}V + \int_{\partial\Omega} \left(\left(\widehat{\nabla\varphi} \otimes \widehat{\nabla\varphi}\right) \nabla u\right) \cdot \vec{n} v\,\mathrm{d}S$$
$$- \lambda \oint_\mathfrak{I} (u - u_0) v\,\mathrm{d}S - \lambda \int_{\partial\Omega_\mathrm{D}} (u - u_\mathrm{D}) v\,\mathrm{d}S = 0 \quad \forall v. \tag{5.13}$$

Interpreting this integral form locally, and rewriting the tensor $\underline{\underline{\kappa}} = \widehat{\nabla\varphi} \otimes \widehat{\nabla\varphi}$, we obtain the PDE

$$\nabla \cdot \left(\underline{\underline{\kappa}} \nabla u\right) = 0 \quad \text{in } \Omega \setminus \mathfrak{I} \tag{5.14a}$$

$$\left(\underline{\underline{\kappa}} \nabla u\right) \cdot \vec{n} = 0 \quad \text{on } \partial\Omega \tag{5.14b}$$

$$u = u_{\mathrm{D}} \quad \text{on } \partial\Omega_{\mathrm{D}} \tag{5.14c}$$

$$u = u_0 \quad \text{on } \mathfrak{I}. \tag{5.14d}$$

This equation describes the steady state limit of anisotropic diffusion with Neumann boundary conditions and the Dirichlet boundary conditions (5.7b) and (5.4c). Note that (5.13) and (5.14) are linear in u. Thus a discretization of the problem can be solved without iteration by a suitable linear solver.

5.3 Upwind discretization

Using equation (5.13), we discretize the problem by introducing a numerical flux $\tilde{f}$ and replace the Lagrange multiplier by a penalty term $2\eta = \lambda$ at the interface.

find $u \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K})$ s.t.

$$0 = a_{\mathrm{ext}}(u, \varphi, v) =$$

$$-\int_{\Omega_h} (\widehat{\nabla\varphi} \otimes \widehat{\nabla\varphi}) \nabla u \cdot \nabla v \,\mathrm{d}V + \int_{\Gamma} \tilde{f}(u, v, \varphi) \,\mathrm{d}S - \int_{\mathfrak{I}} 2\eta(u - u_0) v \,\mathrm{d}S = 0$$

$$\forall v \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}) \tag{5.15}$$

The integral over the interface is calculated using the hierarchical moment-fitting (HMF) quadrature shown in section 3.4.

5.3.1 The original symmetric interior penalty (SIP) flux for anisotropic diffusion

In this context, one of the standard choices for the numerical flux is the SIP flux see e.g. Ern et al. (2008)

$$\tilde{f} = \left(\left\{ \widehat{\nabla\varphi} \cdot \nabla u \right\} [\![v]\!] + \left\{ \widehat{\nabla\varphi} \cdot \nabla v \right\} [\![u]\!] \right) \left\{ \widehat{\nabla\varphi} \cdot \vec{n} \right\} - \eta [\![u]\!] [\![v]\!] . \tag{5.16}$$

Using the shorthand notation

$$\widehat{\nabla\varphi} \cdot \nabla u = \nabla u_\varphi \tag{5.17a}$$

$$\widehat{\nabla\varphi} \cdot \nabla v = \nabla v_\varphi \tag{5.17b}$$

$$\widehat{\nabla\varphi} \cdot \vec{n} = \vec{n}_\varphi \tag{5.17c}$$

for the components projected on the normalized gradient of the level set, this discretization

$$\tilde{f} = \left(\{\nabla u_\varphi\} [\![v]\!] + \{\nabla v_\varphi\} [\![u]\!]\right) \{\vec{n}_\varphi\} - \eta [\![u]\!] [\![v]\!] . \tag{5.18}$$

Following Shahbazi (2005) and Hillewaert (2013), the penalty parameter η is calculated from the selected polynomial degree k and the spatial dimension of the problem d as

$$\eta = \frac{(k+1)(k+d)}{d} \frac{\|\partial K\|}{\|K\|}. \tag{5.19}$$

Since it enforces a Dirichlet value at the interface $\mathfrak{I}$ and the Dirichlet Boundary $\partial\Omega_\mathrm{D}$, the penalty parameter is scaled by a factor of 2, see Shahbazi (2005).

This problem is linear and symmetric, thus the resulting linear system may be solved using a standard solver. However, this still leads to a globally coupled linear system, which has the major drawback that a global system must be solved and that singularities pollute the solution in the entire domain. These singularities arise in all interface geometries, where the curvature radius of the interface is smaller than the size of the computational domain. This means, all test cases shown below cannot be calculated using the SIP flux without diverging results.

Remark on the penalty parameter

In the original publication by the author, Utz and Kummer (2017), the penalty parameter η was scaled by an additional factor μ, which was calculated according to Ern et al. (2008) as

$$\mu = \begin{cases} \left(\frac{1}{\kappa_n^+} + \frac{1}{\kappa_n^-}\right)^{-1} & \forall \kappa_n^\pm \neq 0 \\ 0 & \text{else}, \end{cases} \tag{5.20a}$$

where

$$\kappa_n^\pm = (\vec{n}^T \underline{\underline{\kappa}} \vec{n})^\pm = (\vec{n} \cdot \widehat{\nabla\varphi}^\pm)^2 \tag{5.20b}$$

is the component of the diffusion tensor $\underline{\underline{\kappa}}$ in the direction of the edge-normal $\vec{n}$. Note that this value may become zero when the level set gradient is tangential to the cell face. This decoupling of the two neighboring cells in this case, leads to instability of the scheme in some test cases. Therefore the parameter is removed in the method presented here.

5.3.2 Adaption of the flux for upwinding

The initial problem can be seen as the long-term limit of the time-dependent advection problem (5.5) with $\nabla\varphi$ as the underlying velocity field of the advection. In this original problem, the solution emanates from the interface and is constant along the characteristics. This implies that information moves unidirectionally from the interface into the domain.

We exploit this property to facilitate a marching procedure for the reformulated system. Thus, we reintroduce this directional dependency in our discretization by only using the SIP flux on edges cut by the interface and upwind-fluxes on all other edges. Using the shorthand notation (5.17), this flux reads

$$\tilde{f} = \begin{cases} \left(\{\nabla u_\varphi\}\,[\![v]\!] + \{\nabla v_\varphi\}\,[\![u]\!]\right)\{\vec{n}_\varphi\} - \eta\,[\![u]\!]\,[\![v]\!] & \forall \partial K \cap \mathfrak{I} \neq \emptyset \\ \left(\{\nabla u_\varphi\}\,(-v^+) + (\nabla v_\varphi)^+\,[\![u]\!]\right)\{\vec{n}_\varphi\} - \eta\,[\![u]\!]\,(-v^+) & \forall \left\{\varphi\widehat{\overline{\nabla\varphi}}\cdot\vec{n}\right\} \geq 0 \\ \left(\{\nabla u_\varphi\}\,(v^-) + (\nabla v_\varphi)^-\,[\![u]\!]\right)\{\vec{n}_\varphi\} - \eta\,[\![u]\!]\,(v^-) & \forall \left\{\varphi\widehat{\overline{\nabla\varphi}}\cdot\vec{n}\right\} < 0, \end{cases} \tag{5.21}$$

where $\widehat{\overline{\nabla\varphi}}$ is the cell average of the normalized level set gradient, which ensures a unidirectional flux across each edge.

On the boundaries, the flux (5.21) simplifies to

$$\tilde{f} = \begin{cases} 0 & \forall \text{sgn}(\varphi\widehat{\nabla\varphi}\cdot\vec{n}) \geq 0 \\ \left(\nabla u_\varphi v + \nabla v_\varphi\,(u - u_\mathrm{D})\right)(\widehat{\nabla\varphi}\cdot\vec{n}_\Omega) - \eta(u - u_\mathrm{D})v & \forall \text{sgn}(\varphi\widehat{\nabla\varphi}\cdot\vec{n}) < 0. \end{cases} \tag{5.22}$$

The first line in (5.21) gives the flux between two cells cut by the interface $\mathfrak{I}$. On these edges we introduce a bidirectional coupling, since the value of the level set changes across such edges. The second and third line give the flux for all other edges in the grid. In regions where the level set is positive, information propagates from small to large level set values, i.e. in the direction of a positive gradient, and vice-versa in regions with a negative level set value. A positive value of $\varphi\widehat{\overline{\nabla\varphi}}\cdot\vec{n}$ means, that information propagates from the inner cell $(-)$ into the outer cell $(+)$. In this case the contribution of the inner cell is set to zero, which is equivalent to the Neumann boundary condition (5.14b). The flux for the outer cell can be interpreted as a mixed boundary condition, where the first term in (5.21) gives the gradient in that cell and the last two terms give the absolute value. For the opposite sign of $\varphi\widehat{\overline{\nabla\varphi}}\cdot\vec{n}$ the dependency is vice-versa.

Except for cut-cells, this choice of the fluxes results in a pure upwind matrix for all cells that do not contain the surface $\mathfrak{I}$. Two adjacent cut-cells, on the other hand, are coupled in both directions.

It is important to note that the gradient of the level set $\nabla\varphi$ is evaluated symbolically, thus limiting the order of convergence of the scheme to $\mathcal{O}(h^k)$. This may be circumvented by using a central-difference discretization of the gradient operator. However, this introduces a coupling between neighbouring cells. Due to this coupling, oscilla-

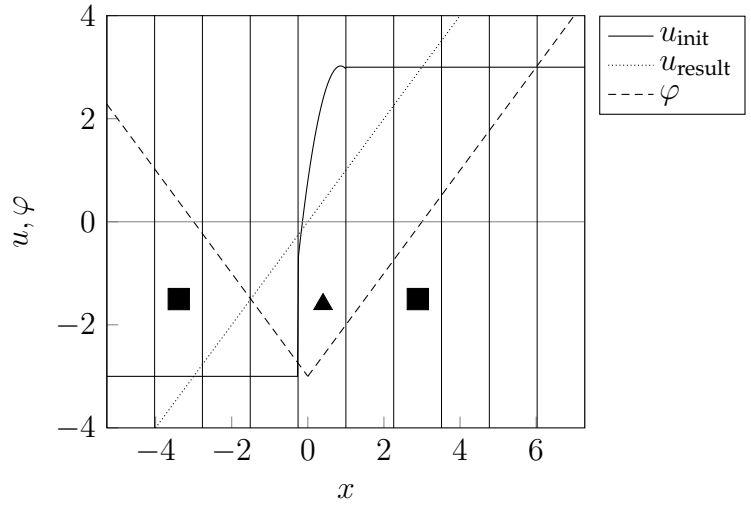
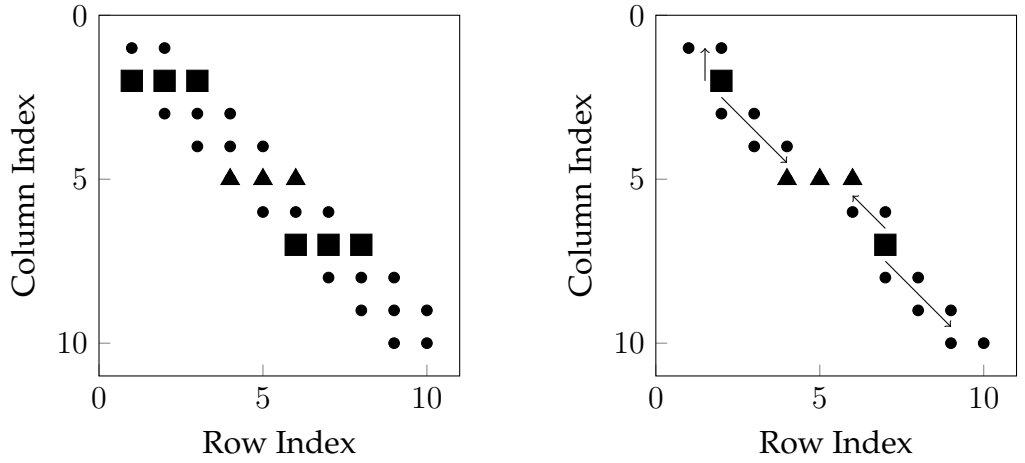


Figure 5.2: Extension and level set for the one-dimensional test case. The vertical lines coincide with the numerical grid. Two surfaces in a one-dimensional domain, the cells are numbered from left to right. The triangle denotes the cell with the singularity, the squares denote the cells containing the interface

tions in the level set at singularities may spoil the accuracy in adjacent cells, leading to a reduced robustness of the scheme. We will show this behaviour using the example of Zalesak's disk, see section 5.4.3.

In Figure 5.2 we show a simple one-dimensional test case. We consider two interface in domain $\Omega = [-2\pi + 1, 2\pi + 1]$ with the level set function $\varphi = |x| - \pi$ and the initial function at the interface $u_0 = x$. This gives two interfaces at $x_\mathfrak{I} = \pm\pi$, leading to the exact solution of $u_{\text{exact}} = \text{sgn}(x)\pi$. This exact solution cannot be represented by the polynomials Note that the overshoot in the center cell containing the interface does not extend into the domain. Figure 5.3 shows the sparsity pattern of the resulting matrix. The whole matrix can be solved cell by cell starting from the cells containing the surface (2 and 7).



(a) Sparsity pattern for the SIP flux, note that no marching is possible, since all cells are globally coupled

(b) Sparsity pattern for the upwind flux, the arrows denote the direction for the marching algorithm

Figure 5.3: Resulting sparsity patterns for the different flux formulations, all entries for one cell are depicted as one data point, the cut-cells are marked by a square, the cell containing the singularity is marked by a triangle, see Figure 5.2

5.4 Test cases

Note, that the actual numerical values presented here slightly differ from the results published in the original publication Utz and Kummer (2017) since the discretization has been altered as explained before. The general results however are the same as shown in the paper. First, we examine a circular level set on a structured grid consisting of quadrilateral elements. For this test case we show high order convergence, even if we do not exclude the point singularity at the centre of the circle from the calculation. Second, we look at two linear interfaces in a channel discretized by triangular elements. In contrast to the first test case, the singularity is a line that is not aligned with the faces of the grid. As for the first test case we obtain high order accuracy. Last, using the disk test case by Zalesak (1979) we demonstrate how the choice of evaluating the gradient of the level set influences the stability of our method . Evaluating the gradient locally leads to a stable method even on unstructured triangular grids.

5.4.1 Circular interface

We consider a circular interface in the rectangular domain $\Omega = [-1.9, 2.1]$. The level set function is $\varphi = \sqrt{x^2 + y^2} - 1$, such that the singularity does not coincide with the faces of the grid. The initial function at the interface is chosen as

$$u_0 = 1 - x^2 + y. \tag{5.23}$$

When dealing with a circular interface, we can reformulate the problem in cylindrical coordinates $u_0(r,\theta) = 1 - (r\cos(\theta))^2 + r\sin(\theta)$. Since the extension is constant on lines defined by $\theta = \text{const.}$, the analytical solution is given by

$$u_{\text{exact}}(r,\theta) = u_0(1,\theta) = 1 - (r\cos(\theta))^2 + r\sin(\theta) = \\ = y\frac{\sqrt{x^2+y^2}+y}{\sqrt{x^2+y^2}}. \tag{5.24}$$

The solution of the extension problem with our algorithm results in isocontours of the extended quantity emanating from the origin of the circle, as shown in Figure 5.4.

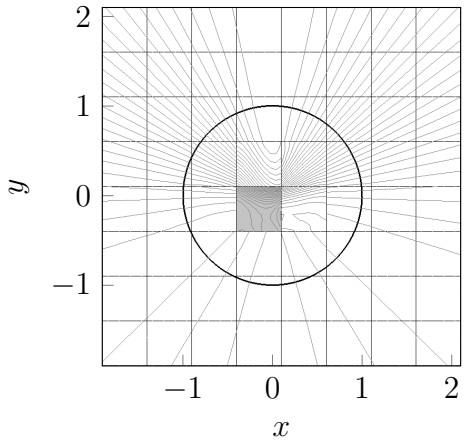
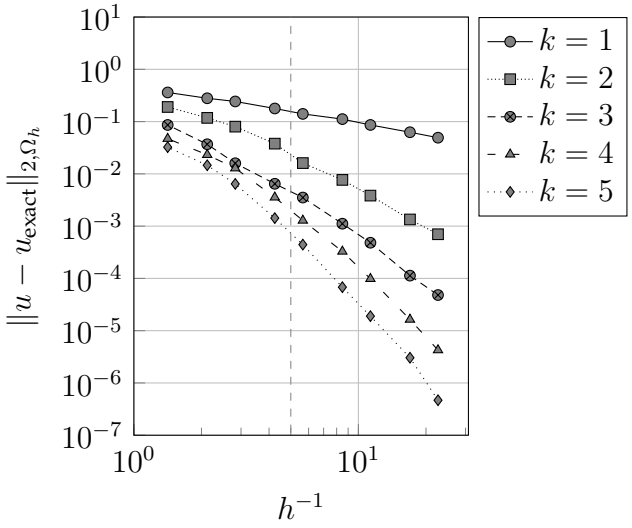


Figure 5.4: Circular test case, coarsest grid, $p = 3$; dark line: level set; light lines: isolines of the extension; grey background: cell with singularity

This solution contains a singularity at the origin $(x,y) = (0,0)$. It is well-known, that such singularities spoils the convergence properties of high-order algorithms. However, the fluxes in this thesis are designed in such a way that the coupling between uncut cells is unidirectional and always in the direction of increasing absolute value of the level set function. Cells that contain the singularity also contain extremal values in the level set function. Therefore, these cells do not influence the neighbouring cells, and thus the error in all cells that do not contain the singularity should not be affected. For this test case we calculate the error norm on a restricted domain $\tilde{\Omega} = [-1.9, 2.1]^2 \setminus [-0.4, 0.1]^2$. Looking at Figure 5.4, this means that we exclude the cell containing the singularity, which is depicted with a grey background. All finer grids are computed by successive splitting of the cells. In Figure 5.5 we compare the experimental order of convergence (EOC) of the L^2-error on the restricted domain for polynomial degrees k of up to 5. This test case gives a convergence rate of h^k. As mentioned before, this suboptimal convergence rate is due to the fact that the signed-distance level set cannot be represented by the polynomial basis exactly. Therefore, the level set itself exhibits an error of $\mathcal{O}(h^{k+1})$ and the error in the level set gradient is $\mathcal{O}(h^k)$. The latter influences the calculation of u.



k	EOC
1	0.77
2	2.31
3	3.14
4	4.15
5	4.84

Figure 5.5: Results of the h-convergence study for the circular test case.
Note: the data points left of the dashed line are not used for the calculation of the EOC, since they lie outside the asymptotic range

5.4.2 Linear interfaces

As a second test case we consider a line singularity that is not aligned with the grid edges. Thus, we set two linear interfaces on a regular triangular grid of the domain $\Omega = [-2\pi + 1, 2\pi + 1] \times [-\pi, \pi]$. The coarsest grid is depicted in Figure 5.6. This configuration gives a singularity, which is not perpendicular to the grid edges.

The two interfaces are represented by the level set function $\varphi = (x - 3)(x + 3)$. Note that this choice is no signed-distance representation for the interfaces at $x = \pm 3$. We choose the initial function at the interface to be

$$u_0 = x \sin(y). \tag{5.25}$$

Then, the analytical solution is given by

$$u_{\text{exact}} = 3 \,\text{sign}(x) \sin(y). \tag{5.26}$$

Note, that we calculate the error on the restricted domain $\tilde{\Omega} = \Omega \setminus [-\pi + 1, 1] \times [-\pi, \pi]$, since the singularity in the gray cells (Figure 5.6) only limits the accuracy in these cells.

Figure 5.7 shows the experimental order of convergence (EOC) for multiple polynomial orders. This test case gives an error of roughly h^{k+1} , which is the optimal convergence-rate for such a DG-scheme. This is due to the quadratic nature of the level set field and the linearity of the interface, which are exactly represented by the polynomial basis-functions. Thus, the calculation of the level set gradient does not limit the accuracy in this test case.

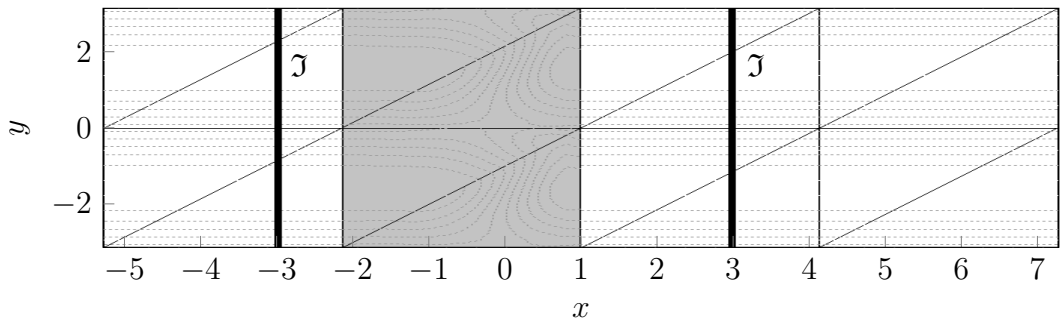
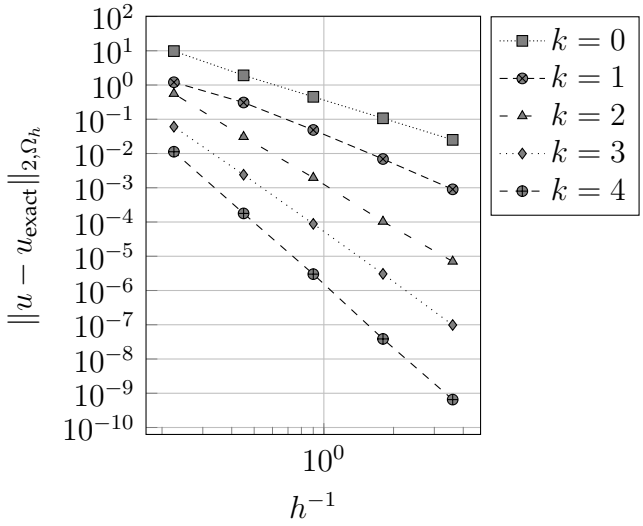


Figure 5.6: Channel with two interfaces, coarsest grid (16 triangular cells) $p = 5$, dark vertical lines: interface, dashed lines: isolines of the extension, grey background: cells with singularity



k	EOC
1	2.11
2	2.71
3	4.02
4	4.85
5	6.03

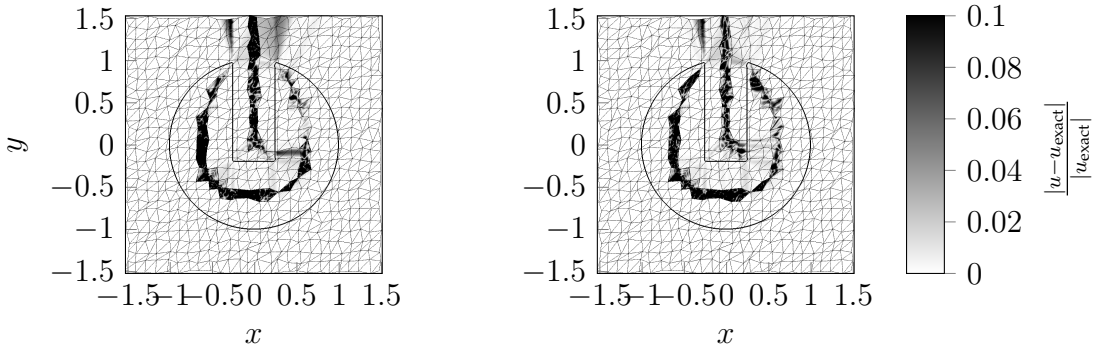
Figure 5.7: Results of the h-convergence study for the channel test case

5.4.3 Zalesaks disk

Next, the the method is applied to the more challenging geometry of the disk test case by Zalesak (1979). This test case consists of a circular contour with a rectangular cut out, leading to sharp kinks in the level set contour. We discretize the domain by a randomly distorted triangular grid, consisting of 25 cells in each spatial direction. In Figure 5.8, we compare the normalized local error in the extension field for the evaluation of the level set gradient $\nabla\varphi$ by a flux formulation and by a local derivative. Evaluating $\nabla\varphi$ by a central difference formulation causes oscillations in the level set function that pollute nearby cells since the value of the gradient is coupled to neighboring cells. This leads to comparably large errors, especially in the regions close to the kinks of the interface. If we evaluate $\nabla\varphi$ locally, possible oscillations in the gradient are limited to the cell containing the kinks, giving stable results.

This demonstrates the stability of our method even on highly irregular grids and for interfaces with high curvatures or kinks and geometries, which are of the same size

as the grid cells. Note that the chosen contour exhibits a curved kink in the level set contour inside the circular domain and three straight kinks in the rectangular part. Such discontinuities cannot be exactly represented by a polynomial basis, which causes oscillations in the level set.



(a) $\nabla\varphi$ evaluated via central differences

(b) $\nabla\varphi$ evaluated locally

Figure 5.8: Zalesak's disk $k = 3$, grey-scale: normalized local error, dark line: level set

5.4.4 Boundary conditions

The next test case is designed to evaluate the method including boundary conditions for the extension equation. The straight line at $x = 0$ can be represented by the level set function

$$\varphi = x + xy. \tag{5.27}$$

It is important to note, that this level set is not a signed distance representation of this geometry. Applying the method of characteristics, the extension problem can be solved analytically by using the coordinate transformation

$$s(x, y) = \sqrt{1 - x^2 + y^2 + 2y} - 1 \tag{5.28}$$

which introduces the parametrization of $s = y$ on $\mathfrak{I}$. Restricting the test case to the domain $\Omega = (-1, 1) \times (0, 1)$ requires additional boundary conditions at $x = 0$, since $s < 0$ does not map to a point on the interface. Therefore, the full boundary conditions are selected as:

$$u = 0 \quad \text{on } y = 0 \tag{5.29a}$$

$$u = x^5 - x^6 \quad \text{on } \mathfrak{I} \tag{5.29b}$$

These boundary conditions are chosen such that they are smooth at the origin and not exactly represented by the polynomials of order up to $k = 5$, which are tested in figure 5.10.

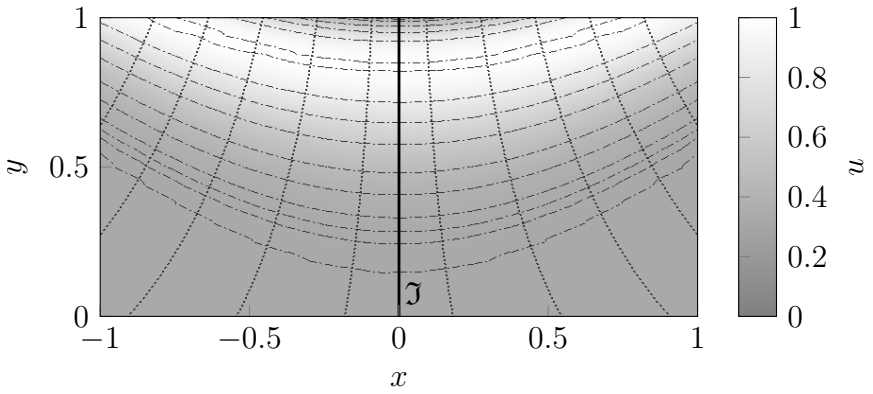
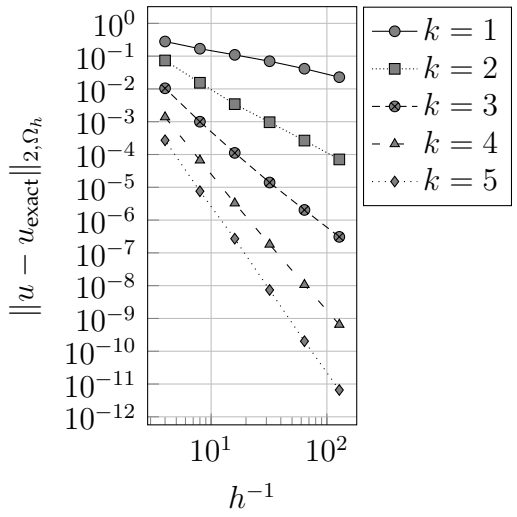


Figure 5.9: Setup for testing the boundary conditions

Using these boundary conditions, the analytical solution is

$$u = \begin{cases} 0 & \forall x | -x^2 + y^2 + 2y < 0 \\ s(x)^5 - s(x)^6 & \text{else,} \end{cases} \tag{5.30}$$

which is shown in gray scale in figure 5.9. To measure the convergence rate of the method, this problem is discretized on a domain of 4×4 cells, and all finer grids are created by successive splitting of the cells. As shown in figure 5.10, this gives the same accuracy of h^k as the tests involving only an extension from the interface.



k	EOC
1	0.71
2	1.99
3	3.00
4	4.20
5	5.07

Figure 5.10: Results of the h-convergence study for the boundary test case

5.4.5 Three dimensional test case

As an additional testcase to Utz and Kummer (2017), we again look at the problem of a torus. Revisiting section 4.3.6 signed distance level set of a torus with a small radius of $r = r_0$ and a large radius of R, see e.g. Abbena et al. (2006).

$$\varphi(x, y, z) = \sqrt{z^2 + \left(\sqrt{x^2 + y^2} - R\right)^2} - r_0 \tag{5.31}$$

For this geometry, a so called toroidal coordinate system can be defined using the transformations

$$x = (R + r\cos(\theta)) * \cos(\phi) \tag{5.32a}$$
$$y = (R + r\cos(\theta)) * \sin(\phi) \tag{5.32b}$$
$$z = r\sin(\theta), \tag{5.32c}$$

see Moon and Spencer (2012), with the inverse transformation

$$\phi = \arctan(\frac{x^2}{y^2}) \tag{5.33a}$$
$$\theta = \arctan(\frac{z^2}{\sqrt{x^2 + y^2} - R}) \tag{5.33b}$$
$$r = \sqrt{\left(\sqrt{x^2 + y^2} - R\right)^2 + z^2}. \tag{5.33c}$$

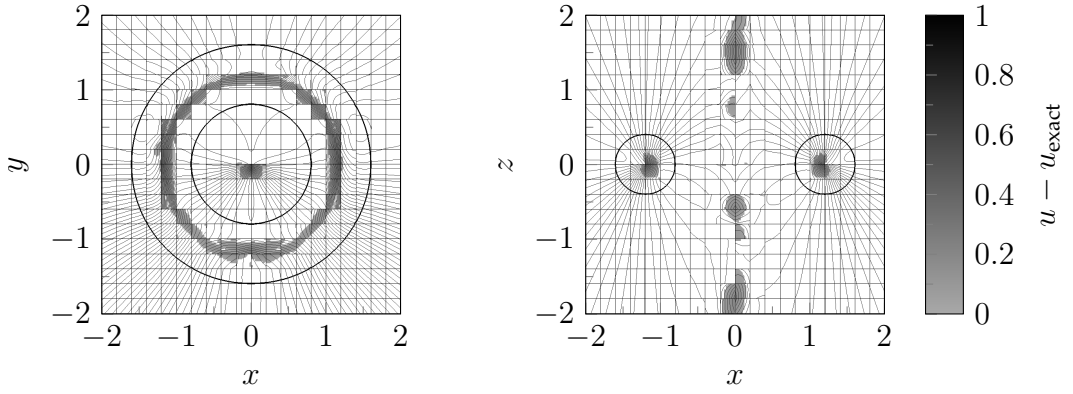
On the surface we employ the initial conditions

$$u_0 = 3 + x^2 + y + z \quad \text{on } \mathfrak{I} \tag{5.34}$$

using the transformation (5.32), we know, that the solution is constant along lines $\theta = \text{const.}$, $\phi = \text{const.}$ and has boundary conditions at $r = r_0$. Thus, the analytical solution is

$$u = 3 + x(\theta, \phi, r_0)^2 + y(\theta, \phi, r_0) + z(\theta, \phi, r_0) \tag{5.35}$$

As in the test case for the reinitialization problem, we choose the computational domain as $\Omega = (-2, 2)^3$ with an equidistant grid of 20^3 cells and a polynomial degree of $k = 2$, see figure 5.11. This test case converges to the analytical solution, with the regions of larger errors located on a ring along the center of the small radius $z = 0, x^2 + y^2 = r^2$ and in the center of the large radius $x = y = 0$.



(a) Section at $z = 0$ (b) Section at $y = 0$

Figure 5.11: Extension from a toroidal surface, errors above $\|u - u_{\text{exact}}\|_1 > 0.05$ are shown in gray scale

5.5 Stabilization of ill-posed problems

The discretization presented in this section and all test cases shown here share one assumption: the original PDE (5.4) is assumed to be well-posed. Only then, the reformulation in equation (5.14) has a solution. However, one can construct cases for the level-set function, where the original problem (5.4) does not have a unique solution. This is especially the case if the level set does not have signed properties. Technically, then (5.3) and (5.4) are not equivalent anymore, but such cases might occur, when solving the reinitialization problem numerically. In this case, one typically resorts to the so called vanishing viscosity solution.

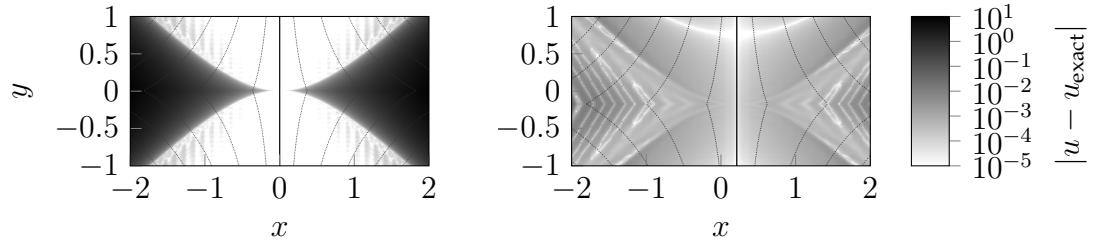
$$\lim_{\epsilon \to 0} \nabla\varphi \cdot \nabla u - \epsilon \Delta u = 0 \quad \text{in } \Omega \tag{5.36}$$

This topic is for example discussed for general Hamilton-Jacobi equation in Tugurlan (2008). Zhao et al. (1996) and Karakus et al. (2014) explicitly add such a term to their hyperbolic methods for level-set reinitialization for stability. this means we extend (5.15) to

$$\begin{aligned} &\text{find } u \in \mathbb{V}_{\text{DG}} \text{ s.t.} \\ a_{\text{ext}}(u, \varphi, v) + \epsilon\, s_{\text{ext}}(u, \varphi, v) &= 0 \\ &\forall v \in \mathbb{V}_{\text{DG}}, \end{aligned} \tag{5.37}$$

where the viscosity ϵ is added by a standard SIP flux in the stabilization term

$$s_{\text{ext}}(u, \varphi, v) = -\int_{\Omega_h} \nabla u \cdot \nabla v \, \mathrm{d}V + \int_{\Gamma \setminus \partial\Omega} (\{\nabla u\} [\![v]\!] + \{\nabla v\} [\![u]\!]) \cdot \vec{n} - \eta [\![u]\!] [\![v]\!] \, \mathrm{d}S \tag{5.38}$$



(a) $\epsilon = 0$, wrong results in the ill-posed region

(b) $\epsilon = 10^{-2}$, error is globally distributed

Figure 5.12: Ill-posed level-set

Our reformulated extension algorithm suffers from the same issue: As an example, we use the same level set function as in the previous section

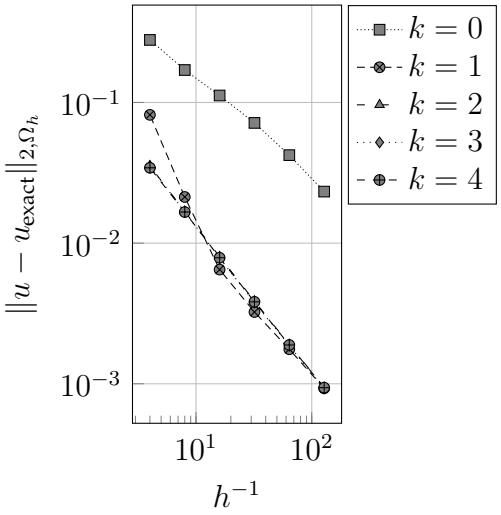
$$\varphi = x + x|y|, \tag{5.39}$$

but change the computational domain to $\Omega = (-2, 2) \times (-1, 1)$. In this case, there are no inflow boundary conditions (5.4c), since the vector $\varphi \nabla \varphi$ is pointing outwards of the domain on all boundaries. Then no solution exists inside the boundaries

$$1 - \sqrt{x^2 + 1} < y < 1 + \sqrt{x^2 + 1}. \tag{5.40}$$

Choosing the boundary condition at the interface symmetric to the x-axis $u_0 = \cos(\pi y)$, the vanishing viscosity solution is in analogy to (5.36).

$$u = \begin{cases} 0 & \forall \{x, y\} | 1 - x^2 + y^2 + 2|y| < 1 \\ \cos(\sqrt{1 - x^2 + y^2 + 2|y|} - 1) & \text{else} \end{cases} \tag{5.41}$$



k	EOC
1	0.70
2	1.26
3	1.05
4	1.05
5	1.04

Figure 5.13: Results of the h-convergence study with $\epsilon = 10^{-2}$

Figure 5.12 shows the results of this test case without and with artificial viscosity. Without artificial viscosity, the results are very accurate in the part of the domain, where the problem is well-posed, but the method fails in the region, where the extension equation is ill-posed. The results with artificial viscosity however show a uniformly distributed error, which is low in the whole domain, but much higher in the well posed part of the domain. This also reflects in the results shown in figure 5.13, which depicts the L^2 error for different cell sizes and polynomial degrees altering the PDE introduces a first order error, such that no high order convergence can be observed.

6 Outlook: Trace DG

This chapter briefly steps aside from the problem of multiphase flows and looks at a more abstract problem. The typical problems for computational engineers are the solution of partial differential equations in some domain Ω. This domain corresponds to the physical problem in question, thus it is either three-dimensional or of lower dimension. However, there are problems which are not defined on the whole domain, but on submanifolds, such as surfaces or space-curves in 3D or a line in 2D. Example solid mechanics problems on plates and shells, fluid flow on the surface of a soap bubble, or the motion of surface active substances at the interface between two fluid phases are introduced in section 2.2.

When solving problems that are defined on a sub-manifold of the computational domain, one faces similar challenges as with solving multiphysics problems. When solving such surface equations, there are two choices: Either the surface itself is explicitly reconstructed. Then one defines a surface grid with its own degrees of freedom, on which the equations can be solved, such as proposed e.g. Antonietti et al. (2015). The other possibility is to use an embedding approach: The idea behind this is to use elements and a function space which is defined in the whole domain. Of course, then one needs methods for integrating finite element spaces, which are cut by the surface as with multiphase physics.

This implies, that solutions to a differential equation on a surface are represented by functions that might vary perpendicular to the surface. Thus one needs some method of stabilization, otherwise the discrete problem is not well posed, see Dziuk and Elliott (2013). Since one is formally looking for solutions in a function space, which is derived by restricting the function space in the whole domain to the surface, these methods are sometimes referred to as Trace-FEM or Cut-FEM. This idea has recently gained some interest after its introduction by Demlow and Dziuk (2007) and has been adapted by Gross and Reusken (2011); Gross et al. (2015) and Olshanskii et al. (2014) for surface convection-diffusion and coupling to a bulk phase in the context of piecewise linear finite elements. In their 2017 Paper, Burman et al. (2017) extend the approach from continuous finite elements to a piecewise linear cut DG method, which they combine with a ghost-penalty stabilization. This is the first work to combine the Trace-FEM idea with a discontinuous Galerkin approach. Grande et al. (2016) are the first to introduce a high-order Cut-FEM approach for surface equations. In their paper they compare different stabilization methods, proving that the so called normal derivative stabilization gives high order accuracy.

A prototype example for partial differential equations on surfaces is the surface Poisson problem

$$\nabla_\mathfrak{I} \cdot (\nabla_\mathfrak{I} u) = f(u) \tag{6.1}$$

on a closed surface, which thus does not require any boundary conditions, but is subject to the periodicity of the domain.

6.1 Discretization

Here, we present a method which combines high order function spaces and the normal mode stabilization as presented by Grande et al. (2016) with a discontinuous Galerkin Ansatz as shown by Burman et al. (2017).

$$\begin{aligned} &\text{find } u \in \mathbb{V}_{\text{DG}} \text{ s.t.} \\ &a(u,v)_{\mathfrak{I}} + s(u,v)_{\Omega_h} = f(v)_{\mathfrak{I}} \end{aligned} \tag{6.2}$$

where $a(u,v)_{\mathfrak{I}}$ is chosen either as the tangential gradient form of the Laplace operator $a_1(u,v)_{\mathfrak{I}}$ or as the full gradient form $a_2(u,v)_{\mathfrak{I}}$.

$$\begin{aligned} a_1(u,v)_{\mathfrak{I}} = &\int_{\mathfrak{I}} \nabla_{\mathfrak{I}} u \cdot \nabla_{\mathfrak{I}} v \,\mathrm{d}\vec{x} \\ &+ \oint_{\Gamma\cap\mathfrak{I}} \left(\{\nabla_{\mathfrak{I}} u\}\cdot\vec{n}_\Gamma \,[\![v]\!] + \{\nabla_{\mathfrak{I}} v\}\cdot\vec{n}_\Gamma \,[\![u]\!]\right) \mathrm{d}S - \oint_{\Gamma\cap\mathfrak{I}} \eta \,[\![u]\!]\cdot[\![v]\!] \,\mathrm{d}S \end{aligned} \tag{6.3}$$

$$\begin{aligned} a_2(u,v)_{\mathfrak{I}} = &\int_{\mathfrak{I}} \nabla u \cdot \nabla v \,\mathrm{d}\vec{x} \\ &+ \oint_{\Gamma\cap\mathfrak{I}} \left(\{\nabla u\}\cdot\vec{n}_\Gamma \,[\![v]\!] + \{\nabla v\}\cdot\vec{n}_\Gamma \,[\![u]\!]\right) \mathrm{d}S - \oint_{\Gamma\cap\mathfrak{I}} \eta \,[\![u]\!]\cdot[\![v]\!] \,\mathrm{d}S \end{aligned} \tag{6.4}$$

$$f(v) = \int_{\mathfrak{I}} fv \,\mathrm{d}V. \tag{6.5}$$

Burman et al. (2016) show, that both forms give second order convergence, when using piecewise linear polynomials on the background mesh. However, numerical experiments using high order polynomials give evidence, that only the form a_2 leads to stable results, when looking at manifolds embedded in a three-dimensional domain. Thus, we limit our numerical test cases to the form a_2, like Burman et al. (2016).

In the literature, multiple stabilization terms $s(u,v)$ are discussed: Grande et al. (2016) and Burman et al. (2016) compare ghost penalty stabilitzation, full gradient surface stabilization , full gradient volume stabilization (only Grande et al. (2016)) and normal gradient stabilization.

However, both publications rule out most stabilization terms:

- Ghost penalization has the disadvantage, that with rising polynomial order, an increasing amount of derivatives has to be evaluated at the face: For piecewise

quadratic functions, it requires the evaluation of the Hessian of u, piecewise cubic function require the evaluation of a third order tensor and so forth. This increases the implementation effort enormously. Massjung (2012)

- full gradient surface stabilization may result in an unbounded condition number for polynomial order $k > 1$ Burman et al. (2016)
- the full gradient volume stabilization does not lead to a consistent discretization for $k > 1$
- Only the normal gradient stabilization gives stable results for arbitrary polynomial degrees k without the need for additional terms like the ghost penalty method

Therefore, the Trace-DG method presented here is stabilized using the normal mode approach, which reads

$$s(u,v)_{\Omega_h} = -\int_{\cup K \in \mathfrak{K}: \|K \cap \mathfrak{I}\| \neq 0} (\nabla\varphi \otimes \nabla\varphi)\nabla u \cdot \nabla v \,\mathrm{d}V + \int_{\cup e \in \Gamma: \|K \cap \mathfrak{I}\| \neq 0} -\eta \,[\![u]\!]\,[\![v]\!] \,\mathrm{d}S. \quad (6.6)$$

This means, to apply the volume and the penalty term of the discretization for the extension equation (5.15) on all cells and edges cute by the interface. Thus, the penalty parameter η is chosen in analogy to the extension problem by equation (5.19).

The surface projection operator, see equation (2.8) can be expressed as

$$P_{\mathfrak{I}}\nabla = I - \vec{n}_{\mathfrak{I}} \otimes \vec{n}_{\mathfrak{I}} = I - \nabla\varphi \otimes \nabla\varphi \quad (6.7)$$

for a signed distance level set. This is why the discretization a_2 does not result in a consistency error, when combined with this stabilization technique.

The integral over the interface is calculated using the hierarchical moment fitting quadrature by Müller et al. (2013). For the sake of simplicity, the method does not employ any treatment at the boundaries and is therefore limited to closed surfaces or surfaces, which are perpendicular to the boundaries of the domain.

6.2 Numerical test cases

This section aims to demonstrate the capabilities of the Trace-DG method. The example of a circular interface demonstrates the accuracy of the method, while the testcase of a square interface and a torus show the stability of the method. Since Trace-DG aims to solve surface PDEs, the errors relevant to the solution are located only at the interface itself. Thus the norms to measure the error are defined only at the interface.

$$\|u - u_{\text{exact}}\|_{2,\mathfrak{I}} = \sqrt{\oint_{\mathfrak{I}} (u - u_{\text{exact}})^2 \,\mathrm{d}S} \quad (6.8)$$

6.2.1 Poisson equation on a circle

As a first canonical testcase in two dimensions, let's consider the Poisson-equation in cylindrical coordinates r, ϕ, z, see e.g. Spurk (2013),

$$\Delta u = \nabla \cdot \nabla u = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial^2 u}{\partial \varphi^2} + \frac{\partial^2 u}{\partial z^2} = f. \tag{6.9}$$

The limit to the surface $r = \text{const.}$ in two dimensions, i.e. $\partial/\partial z = 0$, then gives the surface PDE on the circle

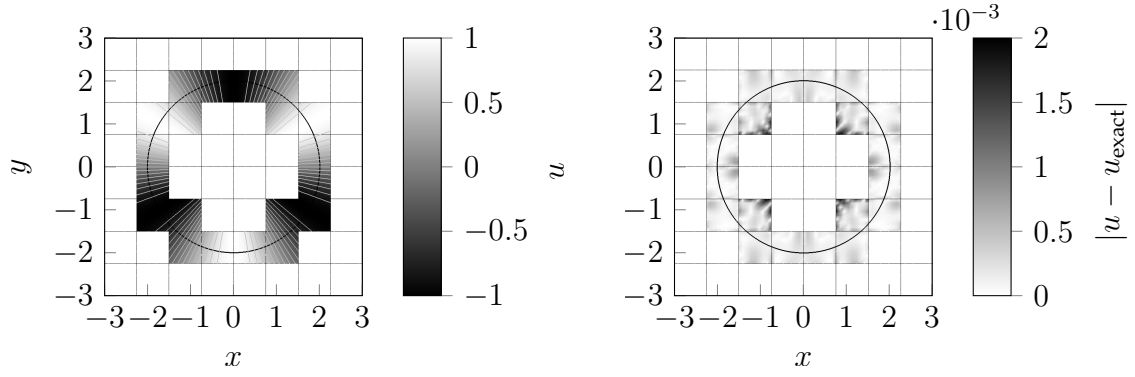
$$\Delta_\mathfrak{I} u = \frac{1}{r}\frac{\partial^2 u}{\partial \varphi^2} = f_\mathfrak{I}, \tag{6.10}$$

for which one analytical solution is

$$u = \sin(a\phi) \tag{6.11a}$$

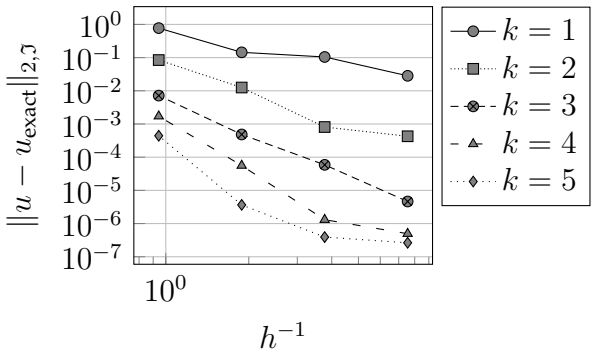
for the right hand side

$$f = -\frac{a^2}{r^2}\sin(a\phi). \tag{6.11b}$$



(a) Solution obtained by Trace-DG (b) Error

Figure 6.1: Results for the Circle Testcase, $k = 5$ on the coarsest grid.



k	EOC
1	1.48
2	2.68
3	3.48
4	4.07
5	3.54

Figure 6.2: Results of the h-convergence study for the poisson-equation on a circle

To examine the convergence behavior of the method, a circular interface with radius $r = 2$ is discretized on the domain $\Omega = [-3, 3]^2$, the meshes used for the study are equidistant with $8, 16, 32$ and 64 cells in both directions. The results for the coarsest grid and a polynomial degree of five is depicted in figure 6.1. Note, that the largest errors are located in cells, where the interface intersects the cell at a corner and thus the interface piece $\mathfrak{I} \cap K$ is quite small. Figure 6.2 shows the results for this test case: polynomial degrees one through four show the desired high order convergence rates, while $k = 5$ is slightly suboptimal. This is due to the fact, that the condition number of the resulting linear system becomes large, since the size of the interface section in each cell vary drastically see e.g. Kummer (2016); Kummer et al. (2018) and Müller et al. (2017) in the context of XDG methods or Legrain et al. (2012); Sauerland (2013) and Lehrenfeld (2015) in the context of extended finite element method (XFEM). Finding a suitable preconditioning strategy to remedy this might be the topic of further investigation.

6.2.2 Poisson equation on a rectangle

The second test case is the surface equation on the rectangle $[-1, 1] \times [-1, 1]$. A suitable level set function for such an interface is

$$\varphi = \inf(|x| - 1, |y| - 1). \tag{6.12}$$

The interface given by this level set is kinked along the lines

$$y = \pm x \quad \forall |x| < 1,\ |y| < 1. \tag{6.13}$$

Since this causes oscillations in the polynomial representation of the level set, high order convergence cannot be expected. However, the method is still required to give reasonably accurate results. For the right hand side

$$f = \frac{\pi^2}{4} \sin(\frac{\pi}{2}x) \sin(\frac{\pi}{2}y) \tag{6.14}$$

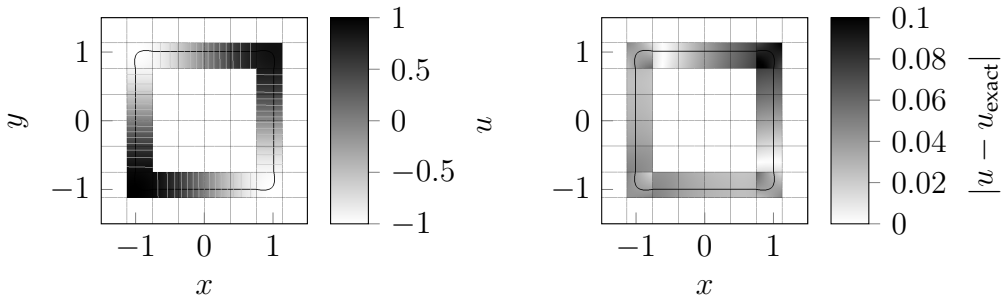
the analytical solution of the surface PDE is

$$u = \begin{cases} -\sin(\frac{\pi}{2}x) & \forall y \geq 0 \cap |y| \geq |x| \\ \sin(\frac{\pi}{2}x) & \forall y < 0 \cap |y| \geq |x| \\ -\sin(\frac{\pi}{2}y) & \forall x \geq 0 \cap |y| \leq |x| \\ \sin(\frac{\pi}{2}y) & \text{else} \end{cases} \tag{6.15}$$

since the surface derivative operator is simply $\frac{\partial}{\partial x}$ at $y = \pm 1$ and $\frac{\partial}{\partial y}$ at $x = \pm 1$.

Figure 6.3 shows the results of this test case for a polynomial degree of $k = 2$ discretized on the domain $\Omega = [-1.5, 1.5]^2$ using eight elements in each spatial direction. The result approximates the solution well. The kinks in the contour at $(x, y)^T = (\pm 1, \pm 1)^T$

however cannot be exactly represented by the level. Therefore, the error in these cells dominates.



(a) Solution from Trace-DG (b) Error

Figure 6.3: Results for the Square test case for a polynomial degree of $k = 2$

6.2.3 Poisson equation on a torus

As a final test case, a setup by Burman et al. (2016) and Grande et al. (2016) is selected: The surface of a torus in 3D is given by the level set

$$\varphi = \sqrt{z^2 + \left(\sqrt{(x^2 + y^2)} - R\right)^2} - r. \tag{6.16}$$

This torus is embedded in a three dimensional domain, which can be parametrized by the toroidal coordinate system (see Moon and Spencer (2012))

$$\vec{x}(\phi, \theta, r) = R \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \\ 0 \end{pmatrix} + r \begin{pmatrix} \cos(\phi)\cos(\theta) \\ \sin(\phi)\cos(\theta) \\ \sin(\theta) \end{pmatrix}, \quad (\phi, \theta) \in [0, 2\pi)^2 . \tag{6.17}$$

With $R = \text{const.}$ and $r = \text{const.}$ giving the surface of the torus. Using this coordinate system, the analytical solution

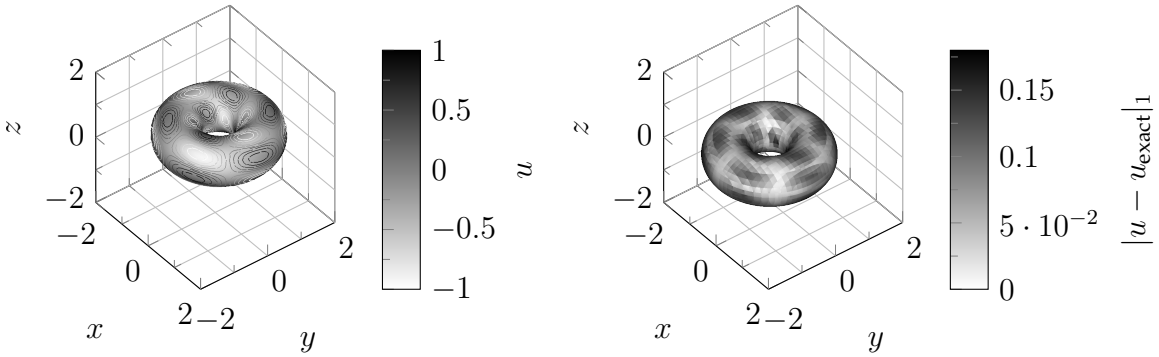
$$u(x) = \sin(3\phi)\cos(3\theta + \phi), \tag{6.18}$$

$$\begin{aligned} -f(x) = {} & \frac{9\sin(3\phi)\cos(3\theta+\phi)}{r^2} + \\ & + \frac{10\sin(3\phi)\cos(3\theta+\phi) + 6\cos(3\phi)\sin(3\theta+\phi)}{(R + r\cos(\theta))^2} \\ & - \frac{3\sin(\theta)\sin(3\phi)\sin(3\theta+\phi)}{r(R + r\cos(\theta))} \end{aligned} \tag{6.19}$$

is adopted from Burman et al. (2016) and Grande et al. (2016).

Figure 6.4 shows the results of the surface poisson equation for this right hand side on a torus with a large radius of $R = 1$ and a small radius $r = 0.6$. The domain is

chosen as $[-2, 2]^3$ discretized by 32 elements in each spatial direction and a polynomial degree of $k = 2$. The result approximates the analytical solution well. The figure 6.4b shows the distribution of the error along the surface. The error is largest, when the intersection of the interface with the underlying grid is rather small and along the inner ring, where the curvature of the interface is largest. Developing measures to improve this might be the scope of further investigation.



(a) Solution using Trace-DG

(b) Error compared to analytical solution

Figure 6.4: Solution of a surface Poisson equation on a toroidal surface

7 A combined motion algorithm for the level set

This chapter describes the coupling of the reinitialization algorithm and the extension algorithm shown in the previous section with a transport equation, to describe the motion of an interface in a prescribed velocity field. The goal of the combined level set algorithms is to move the level set with the velocity at the interface itself and to retain the signed distance property of the level set. This chapter aims to extensively test this combination of algorithms, before applying them to a discretization of the Navier-Stokes equations.

7.1 Coupling of the level set algorithms

Both reinitialization , see chapter 4 and extension, see chapter 5 require the computationally expensive solution of linear systems. To keep the computational cost low, the implicit Crank-Nicholson time stepping scheme is chosen, which allows comparably large time steps Δt. This however introduces a coupling of all three algorithms. This is why, the iterative procedure algorithm 1 is introduced, which allows the computation of the extension velocity based on the interface position at unknown time $t+1$.

For the motion of the interface the calculation of the extension velocity needs to be coupled with the advection step using it, by iteratively solving for the new level set position. Thus the loop in algorithm 1 is repeated until the level set converges within a threshold λ.

Algorithm 1 Motion Algorithm for the level set

procedure MOVELEVELSET(φ,Δt, $\vec{u}$)
  $\varphi^{t+1} \leftarrow \varphi^t$
  while $\|\varphi^{t+1} - \varphi_{\text{old}}\|_2 > \lambda$ **do** ▷ Perform until convergence
    $\varphi_{\text{old}} \leftarrow \varphi^{t+1}$
    $\vec{u}_{\text{ext}} = \text{Extension}(\vec{u}(t+1), \varphi^{t+1})$ ▷ $\nabla \vec{u}_{\text{ext},i} \cdot \nabla \varphi = 0$
    $\varphi^* \leftarrow \text{PerformAdvection}(\Delta t, \varphi^t, \vec{u}_{\text{ext}})$ ▷ $\partial_t \varphi^* + \vec{u}_{\text{ext}} \cdot \nabla \varphi^* = 0$
    $\varphi^{t+1} \leftarrow \text{ReInitialization}(\varphi^*)$ ▷ $|\nabla \varphi| = 1$
  end while
  return $\varphi(t+1)$
end procedure

Extension

The discretization of the advection equation requires a suitable velocity vector $\vec{u}_{\text{ext}}$. Here we follow the approach of Bernauer and Herzog (2011) of extending the individual components of the velocity. Thus, we solve the problem

$$\nabla \vec{u}_{\text{ext},i} \cdot \nabla \varphi = 0 \quad \text{in } \Omega \tag{7.1a}$$

$$\vec{u}_{\text{ext},i} = \vec{u}_i \quad \text{on } \Im \tag{7.1b}$$

for each component of the velocity $\vec{u}$ using the algorithm from chapter 5. This approach conserves the signed distance property of the level set, see appendix A in smooth regions. The motion of the level set might lead to situations, where this equation is ill-posed. Therefore, all numerical test cases use an additional viscosity term ensure stability.

Reinitialization

In this motion algorithm reinitialization is needed for three reasons: Firstly, for general geometries, the extension does not ensure the signed distance property close to singularities. Secondly adding a viscosity term to the extension algorithm also removes this property and third, as we will discuss in the next section 7.2, the transport equation for the level set lacks suitable boundary conditions for general cases, which might lead to oscillations close to boundaries. Aiming to simulate the motion of droplets and bubbles, all three phenomena appear comparably far from the interface itself: Assuming that the grid resolution is fine enough, singularities in the level set usually appear close to the center of closed contours. The extension algorithm presented here is based on an anisotropic viscosity of order $\mathcal{O}(1)$, while the isotropic part is of order $\mathcal{O}(\epsilon)$ which is usually chosen much smaller as $\epsilon \leq 10^{-2}$. Oscillations due to missing boundary conditions only happen at the boundary cells themselves, while the region of interest lies inside the domain. In addition, the test case from section 4.3.4 showed, that the reinitialization introduces a first order error in the position of the interface for general geometries. We therefore use two different reinitialization strategies:
Reinitialization on cells cut by the interface is only required, if the interface is in a cell including the boundary or is strongly curved, which both might introduce oscillations. This case requires the reinitialization algorithm with preconditioning, as shown in 4.4. If the curvature radius of the interface is much larger than the cell size, the interface is located far from the boundaries and the velocity field is rather smooth, it is sufficient to use reinitialization only for smoothing out oscillations at the boundary. In this case only the fast marching algorithm 4.4.1 is used on the so called "far cells", which are all cells which do not share any point with cells cut by the interface.

Since the motion algorithm performs an iterative loop for time stepping, it is sufficient to perform only a single step of reinitialization in every iteration of the loop.

7.2 Discretization of the level set transport equation

To move the level set with a velocity $\vec{u}$, we need to solve an advection type equation of the following form

$$\frac{\partial \varphi}{\partial t} + \vec{u} \cdot \nabla \varphi = 0. \tag{7.2}$$

Note, that for arbitrary velocity fields, this equation is not in divergence form. Therefore we rewrite it as.

$$\frac{\partial \varphi}{\partial t} + \nabla \cdot (\vec{u}\varphi) = \varphi \nabla \cdot \vec{u}. \tag{7.3}$$

Since we will use an extension velocity, this velocity field will not be divergence free and therefore, the source term on the right hand side of equation (7.3) will be non-zero. We discretize this equation in DG as

find $\varphi \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K})$ s.t.

$$\frac{\partial}{\partial t}\int_{\Omega_h} \varphi v \,\mathrm{d}V - \int_{\Omega_h} \varphi \vec{u} \cdot \nabla v \,\mathrm{d}V + \oint_{\Gamma} \tilde{f}(u, v, \varphi) \,\mathrm{d}S - \int_{\Omega_h} \varphi \nabla \cdot \vec{u} v \,\mathrm{d}V = 0 \quad \forall v \in \mathbb{V}_{\mathrm{DG}}(\mathfrak{K}) \tag{7.4}$$

introducing a numerical flux $\tilde{f}$.

Since we are dealing with a hyperbolic equation, a good choice is the local Lax-Friedrichs flux, which reduces to the well known upwind flux for the linear, scalar problem at hand:

$$\tilde{f} = \{\vec{u}\varphi\} \cdot \vec{n} \,[\![v]\!] + |\left\{\overline{\vec{u}} \cdot \vec{n}\right\}| \,[\![\varphi]\!]\,[\![v]\!]\,, \tag{7.5}$$

see e.g. Shahbazi et al. (2007) or Hesthaven and Warburton (2008), where $\overline{\vec{u}}$ denotes the cell average of the velocity vector.

For the temporal discretization we apply a Crank-Nicholson time stepping scheme, see section 3.5.1.

A note on boundary conditions

It is important to note, that neither equation (7.2) nor the discretization (7.4) introduce any boundary condition for φ. Thus, this advection problem is ill-posed. To be well posed, such an advection equation requires Dirichlet boundary conditions if $\vec{u} \cdot \vec{n}_\Omega < 0$. Loch (2013) discusses two possibilities for imposing boundary conditions, which both introduce additional instabilities for a signed-distance level set. This publication therefore does not introduce boundary conditions per se, but sets the flux at inflow boundary conditions to

$$\tilde{f} = \vec{u}\varphi \cdot v \quad \forall \vec{x} \in \partial\Omega : \vec{u} \cdot \vec{n}_\Omega < 0, \tag{7.6}$$

which is equivalent to not setting boundary conditions at all. This may lead to oscillations in the vicinity of such inflows. The algorithm presented here removes these

oscillations, by applying reinitialization to the level set which stabilizes the method, provided that the interface is far from such inflow boundaries. However, if the interface is perpendicular to inflow boundaries, it is possible to define meaningful boundary conditions for (7.2), which completely avoids the issue.

7.3 Test cases

The main goal of this work is to enable a cut-cell discontinuous Galerkin solver to simulate the dynamics of bubbles and droplets. To simulate the motion of a bubble, two features must be captured accurately: the motion of the bubble through the domain and the deformation of the bubble due to an external flow field. This is examined by two individual setups: The first test-case is the motion of a bubble in a velocity field, which is equivalent to a solid-body rotation. Since this vector-field is not irrotational, it is suited to test the motion of a bubble in viscous flow field.

The second test case covers the deformation of a circular bubble due to radial velocity components. This covers stronger local gradients in the velocity and level set field than the rotation test case and emulates a motion similar to that of an oscillating droplet under the influence of surface tension.

To measure the quality of the motion algorithm, we use the contour error E_c as in the chapter on reinitialization, see equation (4.21). If an analytical expression for the interface is known, we can integrate its value along the interface contour provided by the motion algorithm. This local value is the distance between the two contours.

$$E_c = \left(\int_{\Im(t)} \varphi_{\text{analytical}}(t)^2 \, \mathrm{d}S \right)^{\frac{1}{2}} \qquad \text{(see eq. (4.21))}$$

The last test case is the rotation of Zalesaks disk Zalesak (1979). This classical test case demonstrates the stability of the motion algorithm, since the kinks in the interface intrude oscillations in the whole level set field. Due to this behavior it is often used to demonstrate numerical dissipation introduced by a motion algorithm.

7.3.1 Rotating circle

The first test case is the motion of a circular contour in a flow field which is rotating around a point outside the circle as sketched in figure 7.1. While the contour is not deforming, this test case checks, whether translation and rotation is accurately computed.

The velocity field is chosen to have an angular velocity of $\omega = 1$, which means in Cartesian coordinates it can be expressed as

$$\vec{u} = \omega \vec{e_\theta} r = (-y, x)^T \tag{7.7}$$

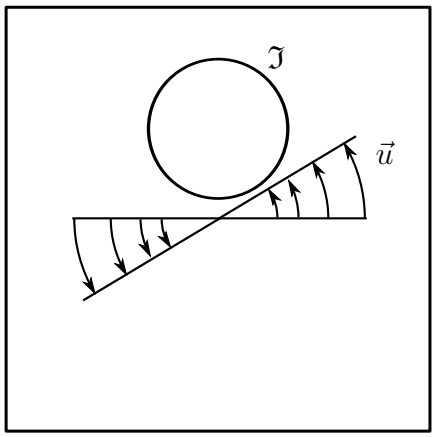


Figure 7.1: Testcase: A circular contour performing a solid-body rotation

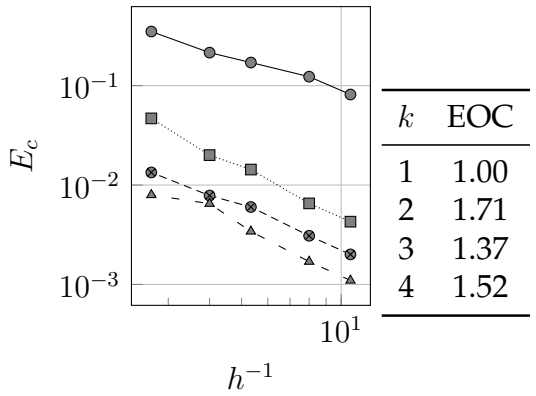
Then using the standard rotational transformation

$$(x^*, y^*)^T = \begin{pmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{pmatrix} (x, y)^T \tag{7.8}$$

the solution can be transformed into a rotating frame of reference, where it is constant

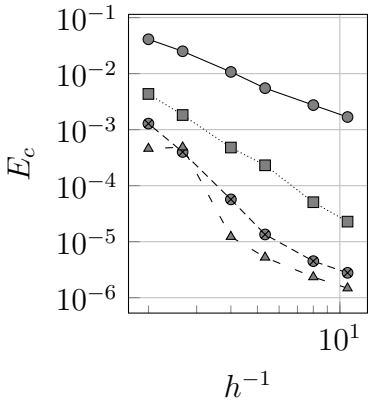
$$\varphi_{\text{exact}}(x^*, y^*) = \sqrt{(x^*)^2 + (y^* - y_0)^2} - 1. \tag{7.9}$$

For this test the vertical offset of the circle from the center of the domain is set to $y_0 = 1.25$. The problem is discretized on the domain $\Omega = (-3, 3)^2$ using computational grids of 12×12 up to 64×64 cells and polynomial degrees $k = 1 \cdots 4$. Time stepping is performed by a Crank-Nicholson scheme using a time step size of $\Delta t = 10^{-3}$. This time step size is chosen small enough to not limit the accuracy of the calculations. The figures 7.2 show the results of two different set ups for the contour error E_c, which measures the shift of the interface by the motion algorithm, evaluated after 300 time steps at $t = 0.3$. For both test cases, the convergence criterion is set to $\lambda = 10^{-8}$. The isotropic viscosity is chosen as $\epsilon = 10^-2$. The figure on the left shows the results for the algorithm including the reinitialization algorithm from chapter 4. Regardless of the polynomial degree k chosen for the calculation, the method gives first order accuracy. If reinitialization is only performed on the far-field, the method shows an accuracy of $E_c \sim h^{k+1}$ for calculations with polynomial degree up to $k \leq 3$ while being slightly suboptimal for $k = 4$. Possible reasons for this reduction might be either the time step size Δt or the choice of the isotropic viscosity ϵ which stabilizes the extension equation.



k	EOC
1	1.00
2	1.71
3	1.37
4	1.52

(a) with PDE-based reinitialization



k	EOC
1	1.95
2	3.16
3	3.82
4	3.84

(b) without PDE-based reinitialization

Figure 7.2: L2-Convergence Study, TimeStepper: Crank-Nicholson, $\Delta t = 0.001$, t=0.3

7.3.2 Oscillating circle

The second test case examines the periodic deformation of a circular contour by a temporally oscillating radial velocity component

$$\vec{u}(r, \theta, t) = \vec{e_r} \cos(n\theta) \cos(\pi t), \tag{7.10}$$

as depicted in figure 7.3. Where θ and r are the independent coordinates for a standard cylindrical coordinate as shown in figure 7.3, see e.g. Spurk (2013).

$$r(x, y) = \sqrt{x^2 + y^2} \tag{7.11a}$$

$$\theta(x, y) = \arctan\left(\frac{y}{x}\right) \tag{7.11b}$$

While the rotating circle test case 7.3.1 covers the rotational and translational motion of a contour, this test case examines the deformation of a contour, as might occur e.g.

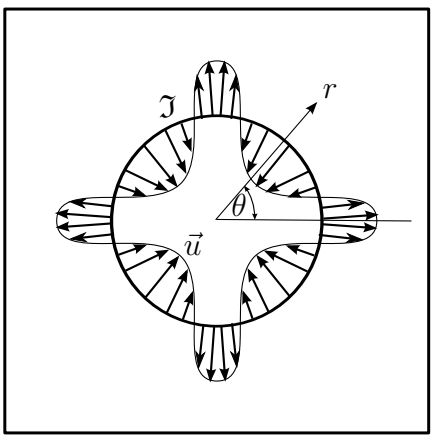


Figure 7.3: Testcase: A circular contour deforming by radial velocity components

in the simulation of an oscillating droplet. With this coordinate transformation an initially circular contour

$$\varphi(t=0) = \sqrt{x^2+y^2} = r - 1$$
$$r(t=0) = 1$$

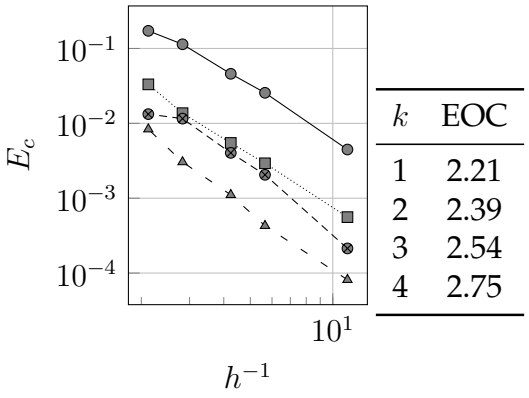
becomes

$$r_{\text{exact}}() = 1 + \frac{1}{\pi}\cos(n\theta)\sin(\pi t)\,. \tag{7.12}$$

One possible level-set for this geometry is

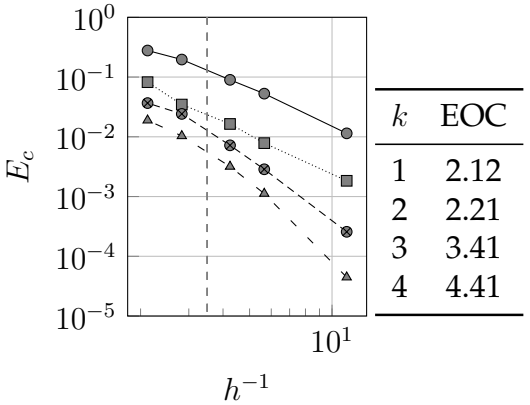
$$\varphi(x,y,t) = 1 + \frac{1}{\pi}\cos\left(n\arctan\left(\frac{y}{x}\right)\right)\sin(\pi t) - \sqrt{x^2+y^2}. \tag{7.13}$$

Note, that this level set does not fulfill the signed distance property, except for times $t \in \mathbb{Z}$. Due to this, the result of the numerical experiment is evaluated after 4 oscillation periods at $t = 4.0$. To measure the convergence rate of the scheme, the problem is discretized on the domain $\Omega = (-2,2)^2$ using polynomial degrees $k = 1 \dots 4$ and an isotropic viscosity of $\epsilon = 10^{-3}$ to stabilize the extension equation. Figure 7.4 shows the results for two different time step sizes: The results for the left figure, 7.4a are obtained using a time step size of $\Delta t = 10^{-2}$, which limits the accuracy of the algorithm to roughly $h^{2.5}$. The limiting factor in this case is the second order accuracy of the involved Crank-Nicholson time stepping scheme. Lowering the time step size to $\Delta t = 1e-3$ reduces the error induced by the time stepping algorithm. Then the computation is dominated by the spatial accuracy of the scheme, which is almost $E_c \sim h^{k+\frac{1}{2}}$.



k	EOC
1	2.21
2	2.39
3	2.54
4	2.75

(a) $\Delta t = 0.01$



k	EOC
1	2.12
2	2.21
3	3.41
4	4.41

(b) $\Delta t = 0.001$, The convergence rate has been calculated from the data points on the right of the dashed line

Figure 7.4: Results of the h-convergence for the oscillating circle test case using different time step sizes Δt.

7.3.3 Zalesaks disk

The last test case is Zalesaks famous disk (see Zalesak (1979)), that was already used to test the extension algorithm individually in section 5.4.3. To test the motion algorithm, this slotted disk is rotated about the origin by a unity angular velocity. The main motivation for this test case is to show that the algorithm presented here is stable even for cases which include sharp kinks in the interface, as they might occur in under resolved simulations. Both previous test cases consisted of interface contours, for which the curvature radius was much larger than the typical cell size. For both these cases kinks in the level set are located far from the level set, which allows to omit reinitialization and gives high order accuracy. The sharp kinks of the contour in Zalesaks disk however introduce large errors in the cells containing them. Since a polynomial cannot interpolate kinks accurately without oscillations, such a contour causes a loss of high-order accuracy, which has been reported for this test case and a discontinuous Galerkin method by Jibben and Herrmann (2012). These sources of errors require a stabilization of the scheme by applying the elliptic reinitialization in every time step.

The geometry for this testcase is shown in figure 7.5, where the geometry parameter of the circle are chosen as $r = 2$, $a = 0.3$ and $b = 1$.

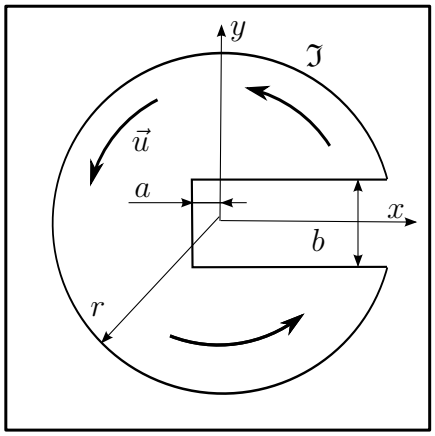


Figure 7.5: Testcase: A slotted disk performing a solid-body rotation

To demonstrate the stability of the algorithm presented here, the test case is discretized on the domain $\Omega = (-2.5, 2.5)^2$ with a very coarse grid of 16×16 cells using a polynomial degree of $k = 3$ and a rather large time step size of $\Delta t = 0.1$. Even for this greatly under resolved setting, the algorithm is stable and gives reasonable results see figure 7.6, although the result suffers from numerical dissipation, which has a smoothing effect on the edges of the contour.

Refining the grid by a factor of three, i.e. a mesh with 48×48 cells and reducing the time step size by a factor of 5 reduces numerical dissipation, which can be seen in figure 7.7. However, comparing figures 7.6 and 7.7b shows, that the dissipation introduced by reinitialization and the singularity at the kink is in the same length scale

as the grid size h, which confirms the findings of the rotating circle test case, section 7.3.1, that reinitialization introduces a first order error in the position of the interface contour.

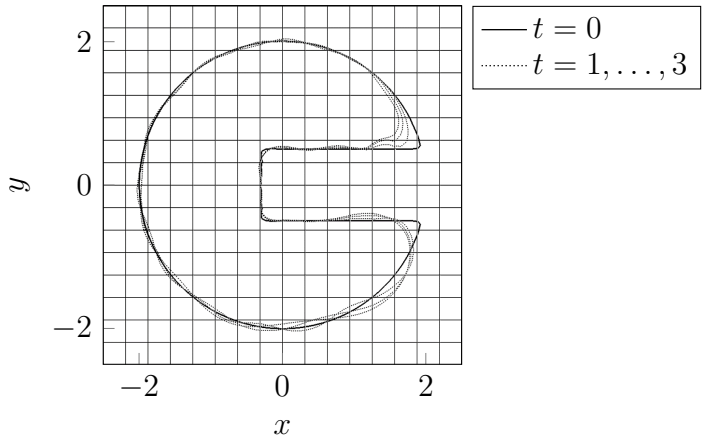
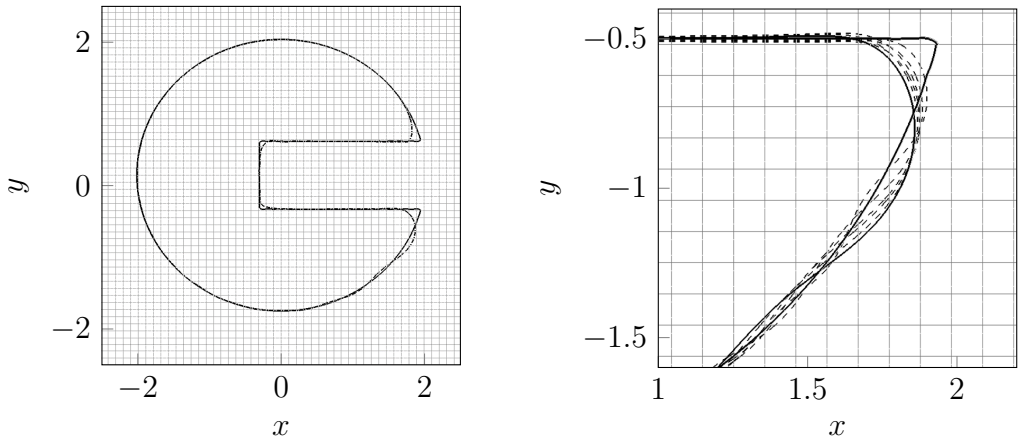


Figure 7.6: Result for Zalesaks disk on the coarse grid, numerical dissipation progressively smooths the interface contour



(a) View of the entire computational domain

(b) Detailed view of the bottom right kink

$t = 0$ - - - $t = 1, \ldots, 5$ $t = 6$

Figure 7.7: Result for Zalesaks disk on the fine grid showing progressive smoothing of the kinks in the interface contour

8 Multiphase flows

As a final step, this chapter applies the level set methods presented in the previous chapters to a solver for incompressible multiphase flows. To discretize the Navier-Stokes equations, we use the method by Kummer (2016) in combination with the time stepping algorithm by Kummer et al. (2018). The resulting velocity field is iteratively coupled to the motion algorithm from the previous section to determine the position of the interface.

8.1 Discretization of the Navier-Stokes equations

In shorthand notation, the discretization for the Navier-Stokes equations is

$$\begin{aligned} &\text{find} (\vec{u}, p) \in \mathbb{V}_{\text{DG}}^{X}(\mathfrak{K}) \text{s. t.} \\ &\int_{\mathfrak{A}(t)\cup\mathfrak{B}(t)} \vec{u}\vec{v}\,\mathrm{d}V\Big|_{t=t^{n+1}} - \int_{\mathfrak{A}(t)\cup\mathfrak{B}(t)} \vec{u}\vec{v}\,\mathrm{d}V\Big|_{t=t^{n}} \\ &+\frac{\Delta t}{2}\Big(Ns(\vec{u},(\vec{u},p),(\vec{v},\tau)) - q(\vec{v}) - r(\tau) - s(\vec{v}) - f(\vec{v})\Big)|_{t=t^n} \\ &+\frac{\Delta t}{2}\Big(Ns(\vec{u},(\vec{u},p),(\vec{v},\tau)) - q(\vec{v}) - r(\tau) - s(\vec{v}) - f(\vec{v})\Big)|_{t=t^{n+1}} = 0 \\ &\forall (\vec{v},\tau) \in \mathbb{V}_{\text{DG}}^{X}(\mathfrak{K}). \end{aligned} \tag{8.1}$$

In time, this is a Crank-Nicholson discretization, which takes the motion of the domain into account. The first two operators denote the mass matrices at the two time steps. The spatial operator consists of the Navier-Stokes operator $\text{NS}(-,-,-)$, terms for the boundary conditions q and r and source terms due surface tension s and external forces f. The Navier Stokes operator can be split into two parts

$$\text{Ns}(\vec{u},(\vec{u},p),(\vec{v},\tau)) = L((\vec{u},p),(\vec{v},\tau)) + t(\vec{u},\vec{u},\vec{v}), \tag{8.2}$$

Where the trilinear form $t(-,-,-)$ represents the nonlinear convection terms, and the bilinear Form $L(-,-)$ represents the linear operator discretizing the Stokes equation

$$L((\vec{u},p),(\vec{v},\tau)) = b(p,\vec{v}) - a(\vec{u},\vec{v}) - b(\tau,\vec{u}). \tag{8.3}$$

The Stokes operator itself consists of three individual operators, where $b(-,-)$ is the discrete form of the pressure gradient and the velocity divergence and $a(-,-)$ represent viscous terms.

The convective terms are discretized by a local Lax-Friedrichs flux,

$$t(\vec{w},\vec{u},\vec{v}) = -\int_{\Omega_h} \rho(\vec{u}\otimes\vec{w}) : \nabla\vec{v}\,\mathrm{d}V - \oint_{\Gamma\cup\partial\Omega_\mathrm{N}\cup\mathfrak{I}} \left(\{\vec{u}\otimes\vec{w}\}\,\vec{n}_\Gamma + (\lambda_\mathrm{LLF}/2)\,[\![\vec{u}]\!]\right)\cdot[\![\rho\vec{v}]\!]\;\mathrm{d}S, \tag{8.4}$$

as used in the work of Shahbazi et al. (2007). See Kummer (2016) for details on the choice of the local Lax-Friedrichs parameter λ_LLF.

The pressure gradient and velocity divergence are discretized by the standard central difference flux

$$b(p,\vec{v}) = -\int_{\Omega_h} p\,\nabla\cdot\vec{v}\,\mathrm{d}V - \oint_{(\Gamma\setminus\partial\Omega_\mathrm{N})\cup\mathfrak{I}} [\![\vec{v}]\!]\cdot\vec{n}_\Gamma\,\{p\}\;\mathrm{d}S. \tag{8.5}$$

Kummer (2016) slightly extends the original symmetric interior penalty (SIP) by Arnold (1982) to the transposed part of the stress trensor as

$$\begin{aligned} a(\vec{u},\vec{v}) = &-\int_{\Omega_h}\mu(\nabla\vec{u} : \nabla\vec{v} + (\nabla\vec{u})^T : \nabla\vec{v})\,\mathrm{d}V \\ &+\oint_{(\Gamma\setminus\partial\Omega_\mathrm{N})\cup\mathfrak{I}} \left(\left\{\mu(\nabla\vec{u}+\nabla\vec{u}^T)\right\}\vec{n}_\Gamma\right)\cdot[\![\vec{v}]\!] + \left(\left\{\mu(\nabla\vec{v}+\nabla\vec{v}^T)\right\}\vec{n}_\Gamma\right)\cdot[\![\vec{u}]\!]\;\mathrm{d}S \\ &-\oint_{(\Gamma\setminus\partial\Omega_\mathrm{N})\cup\mathfrak{I}} \eta\,[\![\vec{u}]\!]\cdot[\![\vec{v}]\!]\;\mathrm{d}S. \end{aligned} \tag{8.6}$$

Using a mixed order discretization, i.e. $k_p = k_{\vec{u}} - 1$, this discretization of the Stokes operator is stable without additional stabilization, see e.g.Shahbazi et al. (2007) and Klein et al. (2013b). For further details on the SIP discretization, refer to the original publication by Arnold (1982) or Arnold et al. (2002) for a detailed analysis. The choice of the penalty parameter must be adapted to the fact, that the method is defined on two arbitrarily small sub cells $K\cap\mathfrak{A}$ and $K\cap\mathfrak{B}$. For details, see Kummer (2016).

At the boundaries, the Dirichlet conditions are added the source terms in the continuity equation

$$r(\tau) = \oint_{\Gamma_\mathrm{D}} \tau\,\vec{u}_\mathrm{D}\cdot\vec{n}_\Omega\,\mathrm{d}S, \tag{8.7}$$

and source terms in the the convective and diffusive part of the momentum equation

$$\begin{aligned} q(\vec{v}) = &-\oint_{\Gamma_\mathrm{D}} (\vec{u}_\mathrm{D}\otimes\vec{u}_\mathrm{D})\,\vec{n}_\Omega\cdot[\![\rho\vec{v}]\!]\;\mathrm{d}S - \oint_{\partial\Omega_\mathrm{D}} \vec{u}_\mathrm{D}\cdot\left(\nabla\vec{v}\vec{n}_\Omega + \nabla\vec{v}^T\vec{n}_\Omega - \eta\vec{v}\right)\;\mathrm{d}S \\ &-\oint_{\partial\Omega_s} \left(-p + \mu\vec{n}_{\partial\Omega_s}\cdot(\nabla\vec{u}+\nabla\vec{u}^T)\cdot\vec{n}_{\partial\Omega_s}\right)(\vec{v}\cdot\vec{n}_{\partial\Omega_s})\,\mathrm{d}S. \end{aligned} \tag{8.8}$$

The boundary conditions used in this thesis are no slip boundary conditions

$$\vec{u}_\mathrm{D} = 0 \quad \text{on } \partial\Omega_0 \tag{8.9}$$

and slip walls (see Reusken et al. (2017))

$$\vec{u} \cdot \vec{n} = 0 \quad \text{on } \partial\Omega_s \tag{8.10}$$

$$\vec{t} \cdot (\nabla\vec{u} + \nabla\vec{u}^T) \cdot \vec{n} = 0 \quad \text{on } \partial\Omega_s, \tag{8.11}$$

where $\vec{t}$ is the tangential vector on $\partial\Omega_s$.

The term due to external forces is

$$f(\vec{v}) = -\int_\Omega \vec{F} \cdot \vec{v} \, \mathrm{d}V \tag{8.12}$$

with $\vec{F} = \vec{g}\rho$ if only gravity acts on the fluid.

The last term is the discretization of the surface tension, which follows the Laplace-Beltrami approach by Demlow and Dziuk (2007). Gross and Reusken (2011) give a broad overview on this kind of discretization.

$$s(\vec{v}) = -\oint_\mathfrak{I} \sigma \mathbf{P}_\mathfrak{I} : \{\nabla\vec{v}\} \, \mathrm{d}S + \int_{\mathfrak{I}\cap\Gamma} \sigma \left\{\tau_\mathfrak{I} \cdot [\![\vec{v}]\!]_\mathfrak{I}\right\}_\Gamma \, \mathrm{d}S \tag{8.13}$$

where $\tau_\mathfrak{I}$ is the tangential vector on the surface.

The resulting discretization (8.1) is nonlinear even for a prescribed interface position. Therefore it must be solved iteratively.

8.2 Coupling of the level set to the Navier Stokes equation

To couple the Navier-Stokes system to the motion algorithm, the algorithm 1 is expanded in the following way:

Algorithm 2 Motion Algorithm including the flow solver

procedure MULTIPHASEFLOWSOLVER(φ^t,Δt, $\vec{u}^t$)
  $\varphi^{t+1} \leftarrow \varphi^t$
  while $\|\varphi^{t+1} - \varphi_{\text{old}}\|_2 > \lambda$ **do** ▷ Perform until convergence
    $\varphi_{\text{old}} \leftarrow \varphi^{t+1}$
    $\vec{u}_{\text{ext}} = \text{Extension}(\vec{u}(t+1), \varphi^{t+1})$ ▷ $\nabla\vec{u}_{\text{ext},i} \cdot \nabla\varphi = 0$
    $\varphi^* \leftarrow \text{PerformAdvection}(\Delta t, \varphi^t, \vec{u}_{\text{ext}})$ ▷ $\partial_t\varphi^* + \vec{u}_{\text{ext}} \cdot \nabla\varphi^* = 0$
    $\varphi^{t+1} \leftarrow \text{ReInitialization}(\varphi^*)$ ▷ $|\nabla\varphi| = 1$
    $\mathfrak{A}^{t+1}, \mathfrak{B}^{t+1} \leftarrow \text{UpdateDomains}(\varphi^{t+1})$
    $\vec{u}^{t+1}, p^{t+1} \leftarrow \text{SolveNSE}(\vec{u}^t, p^t, \mathfrak{A}^t, \mathfrak{B}^t, \mathfrak{A}^{t+1}, \mathfrak{B}^{t+1})$
  end while
  return $\varphi^{t+1}, \vec{u}^{t+1}, p^{t+1}$
end procedure

This is a straight forward adaption of the original iterative scheme, algorithm 1, with the difference that the velocity for the level set motion is not preset, but computed from the Navier-Stokes system. Since the fluid interface moves in each iteration of the solver, the interface position and thus the two domains $\mathfrak{A}$ and $\mathfrak{B}$ must be recomputed in each time step.

Without mass transport, the jump condition for the velocity at the interface is $[\![\vec{u}]\!] = 0$ see (2.4b) . In the XDG method used for the discretization of the Navier-Stokes equations, this condition is enforced only weakly. Thus, the two velocity components must not necessarily be equal. Then, the interface velocity, which is the boundary condition for the extension velocity, must be some average of the values in the two phases. In flows with largely varying densities between the phases, e.g. water and air, the motion of the interface is dominated by the motion of the heavier fluid. Thus, the boundary condition for the extension equation is chosen as the density averaged velocity at the interface

$$\vec{u}_{\text{ext}} = \frac{\rho_{\mathfrak{A}}\vec{u}_{\mathfrak{A}} + \rho_{\mathfrak{B}}\vec{u}_{\mathfrak{B}}}{\rho_{\mathfrak{A}} + \rho_{\mathfrak{B}}} \quad \text{on } \mathfrak{I} \tag{8.14}$$

8.3 Rising bubble test case

To demonstrate the capabilities of the numerical methods presented in this thesis, we apply the Navier-Stokes solver to a bubble of light liquid surrounded by heavy liquid with surface tension between both phases. This test case is quite popular to validate such a solver and detailed results are available for the three finite element based solvers TP2D, FreeLIFE and MooNMD from the reference paper by Hysing et al. (2009). In the context of an unfitted discontinuous Galerkin method, this test case has been demonstrated by Heimann et al. (2013).

The set up as shown in figure 8.1: a circular bubble with a radius of $r = 0.25$ and its centerpoint at $(0.5, 0.5)$ is set into the domain $\Omega = (0, 1) \times (0, 2)$ with a no slip wall $\partial\Omega_0$ at the top and bottom and slip walls $\partial\Omega_s$ on the sides. The fluid of the bubble, $\mathfrak{A}$ is chosen to have a density of $\rho_{\mathfrak{A}} = 100$ and a viscosity of $\mu_{\mathfrak{A}} = 1$. The denser fluid, $\mathfrak{B}$ has a density of $\rho_{\mathfrak{B}} = 1000$ and a viscosity of $\mu_{\mathfrak{B}} = 10$. The surface tension of this material pair is $\sigma = 25$. The initial conditions are chosen as $\vec{u} = 0$ in the entire domain.

To compare the results to benchmark solutions by Hysing and Turek (2005) and Heimann et al. (2013), the following quantities are computed: The area A of the bubble

$$A = \int_{\mathfrak{A}} 1 \, \mathrm{d}V, \tag{8.15}$$

the center of Mass $\vec{x}_c$

$$\vec{x}_c = \frac{\int_{\mathfrak{A}} \vec{x} \, \mathrm{d}V}{A}, \tag{8.16}$$

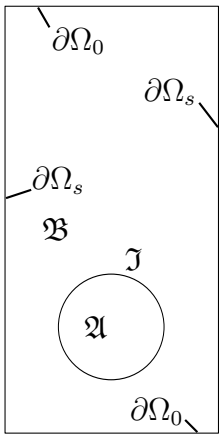


Figure 8.1: Initial and boundary condition for the rising bubbble

and the mean velocity of the bubble

$$\vec{u}_{\text{mean}} = \frac{\int_{\mathfrak{A}} \vec{u}\,\mathrm{d}V}{A}, \tag{8.17}$$

of which the vertical component is the rise velocity u_{rise}. Using the area of the bubble, one can define an equivalent radius

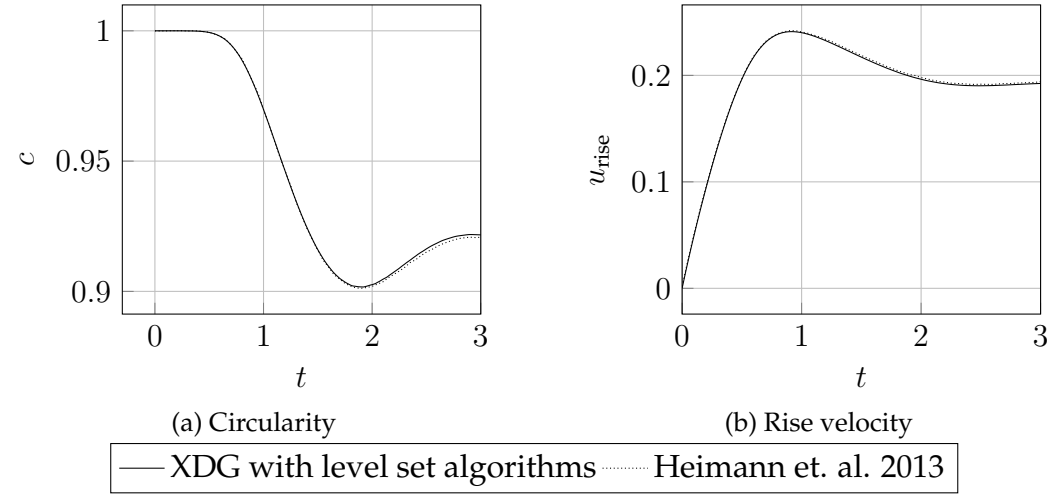
$$r_{\text{equiv.}} = \frac{\sqrt{A}}{4\pi}. \tag{8.18}$$

Without mass loss, this is equivalent to the initial radius of the bubble. The circularity of the bubble c is then defined as the ratio between the circumference of the equivalent circular bubble and the surface

$$c = \frac{2\pi r_{\text{equiv.}}}{\oint_{\mathfrak{I}} 1\,\mathrm{d}S} = \frac{\sqrt{A}}{2\oint_{\mathfrak{I}} 1\,\mathrm{d}S}. \tag{8.19}$$

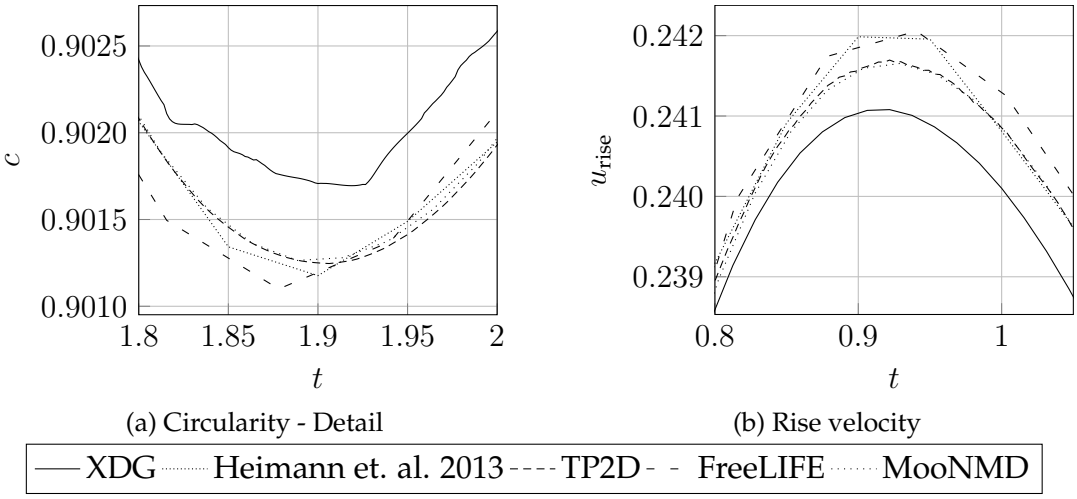
The results shown here are calculated using a grid of 40×80 cells and a polynomial degree of $k_\varphi = k_{\vec{u}} = 2$ for the level set and velocity field and a polynomial degree of $k_p = 1$ for the pressure field. The time step size is chosen as $\Delta t = 0.0015625$. Figure 8.2 and 8.3 compare the results from this computation with the quantities reported in Hysing et al. (2009) and Heimann et al. (2013). Both circularity and rise velocity show good agreement with the benchmark results, despite using twice as large time steps and elements as Heimann et al. (2013). The detailed view 8.3 shows, that the method presented here slightly overestimates the circularity of the bubble at the minimum and underestimates the maximum of the rise velocity by a factor of roughly 10^{-3}. Possible reasons might be the coarser resolution of the results presented here, differences in the handling of the level set or in the discretization. This might be the topic of further investigation. Figure 8.4 shows the magnitude of the local velocity. The simulation accurately covers the flow around the bubble as well as the dynamics in the wake and inside the bubble. As can be seen from this figure and the graph in figure 8.2, the

evolution of the bubbles shape due to the formation of the flow around it is represented well. To fully examine the capabilities of the solver presented here, further studies with different resolutions, material parameters and physical test cases are required.



(a) Circularity (b) Rise velocity

Figure 8.2: Comparison of the rising bubble test case with the benchmark data by Heimann et al. (2013)



(a) Circularity - Detail (b) Rise velocity

Figure 8.3: Comparison of the rising bubble test case with the benchmark data by Heimann et al. (2013) and Hysing et al. (2009) - detailed view

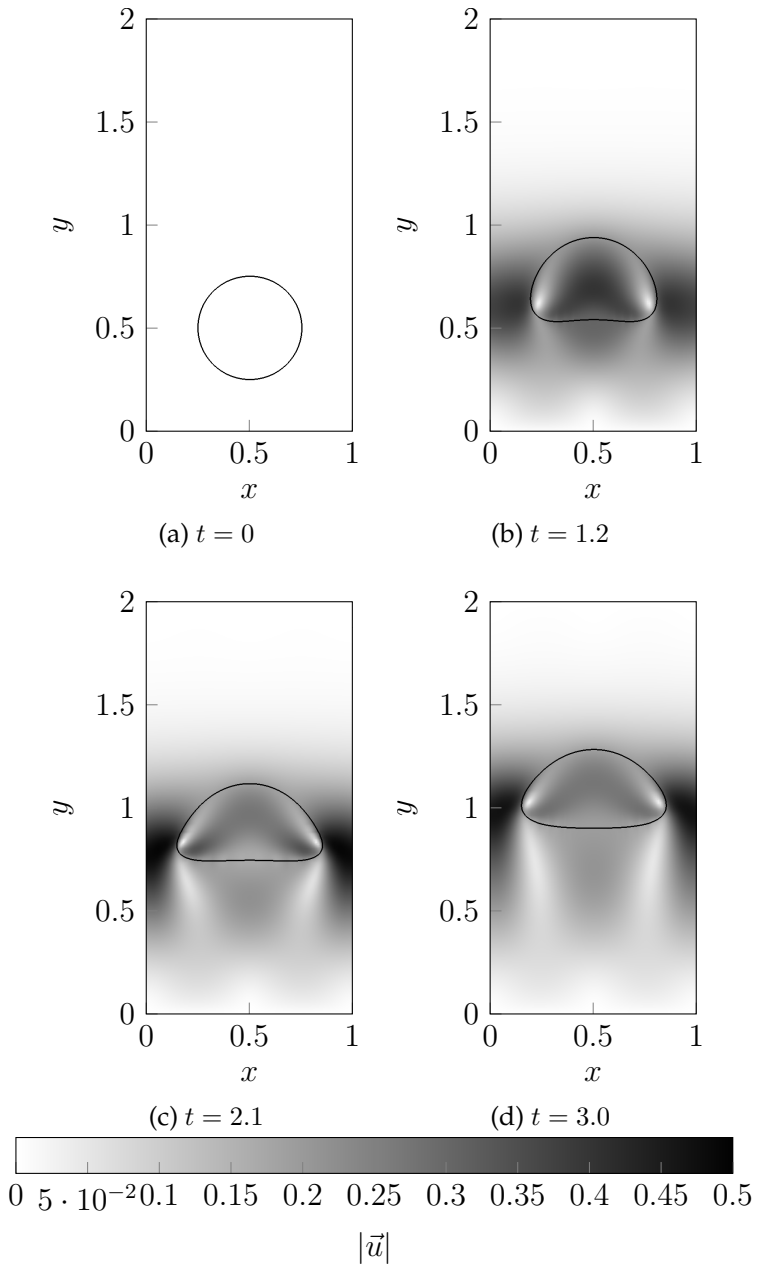


Figure 8.4: Snapshots of the interface and the velocity magnitude for multiple timesteps

9 Conclusion and outlook

This thesis covers several topics in the context of the simulation of multiphase flows using an extended discontinuous Galerkin method (XDG): A novel method for reinitialization, a method for the extension problem, the combination of both these algorithms with a level set advection equation and an application of this algorithm to the simulation of a rising bubble. In addition, an outlook is given, how the ideas for the extension equation can be used to simulate surface equations.

9.1 Reinitialization

This chapter is based on the publication by the author Utz et al. (2017b) and presents a procedure for level-set reinitialization, which accurately preserves the initial position of the interface and converges in global norms of the level-set field, but gives only first order accuracy in a norm measuring the shift of the interface. Both potential functions presented here exhibit individual drawbacks: The single-well potential ψ_1 may lead to oscillations in the vicinity of singularities, which may cause divergence of the scheme. When applying the double well potential, the method is stable even for interface features, which are about the same size as the grid cells and works on triangular and quadrilateral grids. However, using the double well potential in regions, where the initial conditions are overly flat, may lead to a constant solution, instead of the desired signed distance. In addition to the results published in Utz et al. (2017b), these drawbacks are avoided by combining the advantages of both approaches with a first order preconditioning based on a fast marching procedure.

9.2 Extension

This chapter is based on the second publication by the author Utz and Kummer (2017) and presents a novel technique for solving the extension problem, by reformulating the problem into an elliptic differential equation. The nature of the original problem allows a discretization using an upwind flux, which leads to a triangular matrix structure. Thus, the discretization of the extension problem can be solved using a marching algorithm. Numerical evidence indicates an accuracy of $\mathcal{O}(h^k)$ and stability on structured and unstructured grids, even for geometrical features in the same size as the grid cells. In addition to the original publication, this thesis includes the treatment of boundary conditions, i.e. cases, in which the interface intersects the boundary of the domain and cases, where the initial extension problem is ill-posed. This requires a modification of the original problem by adding artificial diffusion. This added

diffusion introduces a first order error, but stabilizes the method even for ill-posed conditions.

9.3 Trace-DG

As an outlook we briefly introduced a technique for solving surface PDEs called Trace DG. This technique shows promising results while relying on a stabilization approach which is similar to the method used for the extension problem. First results are promising for two dimensional and three dimensional test cases alike.

9.4 Combining the algorithms

This chapter combines the extension and the reinitialization algorithm with an upwind discretization to a motion algorithm for interfaces with arbitrary velocities. Since the extension and the reinitialization both require the solution of linear systems, the algorithm uses implicit time stepping to achieve large time steps, which requires an iterative coupling of the different steps. For smooth problems, level-set reinitialization can be limited to cells far from the interface, then the algorithm is high order accurate. Including the reinitialization procedure, the method is stable even for low-regularity problems such as Zalesaks disk.

9.5 Multiphase flows

In the final chapter, the level set algorithm is applied to a sharp interface discretization of the Navier-Stokes equations by Kummer (2016) which includes an adapted time stepping scheme by Kummer et al. (2018) and a Laplace-Beltrami formulation for the surface tension terms. In the context of a simulation of a rising bubble, the motion algorithm for the interface shows good agreement with benchmark results from literature even for rather coarse grids and time step sizes. It will be interesting to apply this solver to more complex flows, such as the collision of droplets or the breakup of jets.

Bibliography

Abbena, E., Salamon, S., and Gray, A. (2006). *Modern Differential Geometry of Curves and Surfaces with Mathematica, Third Edition*. CRC Press.

Adalsteinsson, D. and Sethian, J. A. (1995). A Fast Level Set Method for Propagating Interfaces. *Journal of Computational Physics*, 118(2):269–277.

Adalsteinsson, D. and Sethian, J. A. (1999). The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148(1):2–22.

Amestoy, P. R., Guermouche, A., L'Excellent, J.-Y., and Pralet, S. (2006). Hybrid scheduling for the parallel solution of linear systems. *Parallel computing*, 32(2):136–156.

Antonietti, P., Dedner, A., Madhavan, P., Stangalino, S., Stinner, B., and Verani, M. (2015). High Order Discontinuous Galerkin Methods for Elliptic Problems on Surfaces. *SIAM Journal on Numerical Analysis*, 53(2):1145–1171.

Arnold, D. N. (1982). An Interior Penalty Finite Element Method with Discontinuous Elements. *SIAM Journal on Numerical Analysis*, 19(4):742–760.

Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D. (2002). Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779.

Aslam, T., Luo, S., and Zhao, H. (2014). A static PDE Approach for MultiDimensional Extrapolation Using Fast Sweeping Methods. *SIAM Journal on Scientific Computing*, 36(6):A2907–A2928.

Aslam, T. D. (2004). A partial differential equation approach to multidimensional extrapolation. *Journal of Computational Physics*, 193(1):349–355.

Basting, C. and Kuzmin, D. (2012). A minimization-based finite element formulation for interface-preserving level set reinitialization. *Computing*, 95(1):13–25.

Basting, C. and Kuzmin, D. (2014). Optimal control for mass conservative level set methods. *Journal of Computational and Applied Mathematics*, 270:343–352.

Beck, A. D., Bolemann, T., Flad, D., Frank, H., Gassner, G. J., Hindenlang, F., and Munz, C.-D. (2014). High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations. *International Journal for Numerical Methods in Fluids*, 76(8):522–548.

Bernauer, M. K. and Herzog, R. (2011). Implementation of an X-FEM Solver for the Classical Two-Phase Stefan Problem. *Journal of Scientific Computing*, 52(2):271–293.

Burman, E., Hansbo, P., Larson, M. G., and Massing, A. (2016). Cut Finite Element Methods for Partial Differential Equations on Embedded Manifolds of Arbitrary Codimensions. *arXiv:1610.01660 [math]*. arXiv: 1610.01660.

Burman, E., Hansbo, P., Larson, M. G., and Massing, A. (2017). A cut discontinuous Galerkin method for the Laplace–Beltrami operator. *IMA Journal of Numerical Analysis*, 37(1):138–169.

Chen, S., Merriman, B., Osher, S., and Smereka, P. (1997). A Simple Level Set Method for Solving Stefan Problems. *Journal of Computational Physics*, 135(1):8–29.

Cheng, Y. and Shu, C.-W. (2007). A discontinuous Galerkin finite element method for directly solving the Hamilton–Jacobi equations. *Journal of Computational Physics*, 223(1):398–415.

Chessa, J., Smolinski, P., and Belytschko, T. (2002). The extended finite element method (XFEM) for solidification problems. *International Journal for Numerical Methods in Engineering*, 53(8):1959–1977.

Chéné, A. d., Min, C., and Gibou, F. (2007). Second-Order Accurate Computation of Curvatures in a Level Set Framework Using Novel High-Order Reinitialization Schemes. *Journal of Scientific Computing*, 35(2-3):114–131.

Chopp, D. L. (2000). A Level-Set Method for Simulating Island Coarsening. *Journal of Computational Physics*, 162(1):104–122.

Chopp, D. L. (2001). Some Improvements of the Fast Marching Method. *SIAM Journal on Scientific Computing*, 23(1):230–244.

Cockburn, B., Kanschat, G., and Schötzau, D. (2005). A locally conservative LDG method for the incompressible Navier-Stokes equations. *Mathematics of Computation*, 74(251):1067–1095.

Cockburn, B., Karniadakis, G. E., and Shu, C.-W. (2000). The Development of Discontinuous Galerkin Methods. In Cockburn, B., Karniadakis, G. E., and Shu, C.-W., editors, *Discontinuous Galerkin Methods*, number 11 in Lecture Notes in Computational Science and Engineering, pages 3–50. Springer Berlin Heidelberg.

Cockburn, B. and Shu, C.-W. (1991). The Runge-Kutta local projection $ P^ 1$-discontinuous-Galerkin finite element method for scalar conservation laws. *ESAIM: Mathematical Modelling and Numerical Analysis*, 25(3):337–361.

Cockburn, B. and Shu, C.-W. (1998). The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463.

Della Rocca, G. and Blanquart, G. (2014). Level set reinitialization at a contact line. *Journal of Computational Physics*, 265:34–49.

Demlow, A. and Dziuk, G. (2007). An Adaptive Finite Element Method for the Laplace-Beltrami Operator on Implicitly Defined Surfaces. *SIAM Journal on Numerical Analysis*, 45(1):421–442.

Desjardins, O. and Pitsch, H. (2009). A spectrally refined interface approach for simulating multiphase flows. *Journal of Computational Physics*, 228(5):1658–1677.

Detrixhe, M., Gibou, F., and Min, C. (2013). A parallel fast sweeping method for the Eikonal equation. *Journal of Computational Physics*, 237:46–55.

Di Pietro, D. A. and Ern, A. (2012). *Mathematical aspects of discontinuous galerkin methods*. Springer, Berlin; New York.

Dziuk, G. and Elliott, C. M. (2013). Finite element methods for surface PDEs. *Acta Numerica*, 22:289–396.

Elias, R. N., Martins, M. A. D., and Coutinho, A. L. G. A. (2007). Simple finite element-based computation of distance functions in unstructured grids. *International Journal for Numerical Methods in Engineering*, 72(9):1095–1110.

Engwer, C. (2009). *An unfitted discontinuous Galerkin scheme for micro-scale simulations and numerical upscaling*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, Heidelberg.

Ern, A., Stephansen, A. F., and Zunino, P. (2008). A discontinuous Galerkin method with weighted averages for advection-diffusion equations with locally small and anisotropic diffusivity. *IMA Journal of Numerical Analysis*, 29(2):235–256.

Fechter, S. and Munz, C.-D. (2015). A discontinuous Galerkin-based sharp-interface method to simulate three-dimensional compressible two-phase flow. *International Journal for Numerical Methods in Fluids*, pages n/a–n/a.

Fortmeier, O. and Bücker, M. H. (2011). Parallel re-initialization of level set functions on distributed unstructured tetrahedral grids. *Journal of Computational Physics*, 230(12):4437–4453.

Fries, T.-P. and Zilian, A. (2009). On time integration in the XFEM. *International Journal for Numerical Methods in Engineering*, 79(1):69–93.

Frolkovic, P., Mikula, K., and Urbán, J. (2015). Distance function and extension in normal direction for implicitly defined interfaces. *Discrete and continuous dynamical systems - series S*, 8(5):871–880.

Fu, Z., Jeong, W.-K., Pan, Y., Kirby, R. M., and Whitaker, R. T. (2011). A Fast Iterative Method For Solving The Eikonal Equation on Triangulated Surfaces. *SIAM journal on scientific computing : a publication of the Society for Industrial and Applied Mathematics*, 33(5):2468–2488.

Fu, Z., Kirby, R., and Whitaker, R. (2013). A Fast Iterative Method for Solving the Eikonal Equation on Tetrahedral Domains. *SIAM Journal on Scientific Computing*, 35(5):C473–C494.

Gassner, G., Lörcher, F., and Munz, C.-D. (2007). A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes. *Journal of Computational Physics*, 224(2):1049–1063.

Gibou, F. and Fedkiw, R. (2005). A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem. *Journal of Computational Physics*, 202(2):577–601.

Gibou, F., Fedkiw, R., Caflisch, R., and Osher, S. (2003). A Level Set Approach for the Numerical Simulation of Dendritic Growth. *Journal of Scientific Computing*, 19(1-3):183–199.

Gibou, F., Fedkiw, R. P., Cheng, L.-T., and Kang, M. (2002). A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. *Journal of Computational Physics*, 176(1):205–227.

Girault, V., Rivière, B., and Wheeler, M. (2005). A discontinuous Galerkin method with nonoverlapping domain decomposition for the Stokes and Navier-Stokes problems. *Mathematics of Computation*, 74(249):53–84.

Grande, J., Lehrenfeld, C., and Reusken, A. (2016). Analysis of a high order Trace Finite Element Method for PDEs on level set surfaces. *arXiv:1611.01100 [math]*. arXiv: 1611.01100.

Groß, S., Reichelt, V., and Reusken, A. (2006). A finite element based level set method for two-phase incompressible flows. *Computing and Visualization in Science*, 9(4):239–257.

Grooss, J. and Hesthaven, J. S. (2006). A level set discontinuous Galerkin method for free surface flows. *Computer Methods in Applied Mechanics and Engineering*, 195(25–28):3406–3429.

Gross, S., Olshanskii, M. A., and Reusken, A. (2015). A trace finite element method for a class of coupled bulk-interface transport problems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 49(5):1303–1330.

Gross, S. and Reusken, A. (2011). *Numerical Methods for Two-phase Incompressible Flows*. Springer Science & Business Media.

Harlow, F. H. and Welch, J. E. (1965). Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8(12):2182.

Hartmann, D., Meinke, M., and Schröder, W. (2008). Differential equation based constrained reinitialization for level set methods. *Journal of Computational Physics*, 227(14):6821–6845.

Hartmann, D., Meinke, M., and Schröder, W. (2010). The constrained reinitialization equation for level set methods. *Journal of Computational Physics*, 229(5):1514–1535.

Heimann, F., Engwer, C., Ippisch, O., and Bastian, P. (2013). An unfitted interior penalty discontinuous Galerkin method for incompressible Navier-Stokes two-phase flow: UDG FOR INCOMPRESSIBLE NAVIER-STOKES TWO-PHASE FLOW. *International Journal for Numerical Methods in Fluids*, 71(3):269–293.

Hesthaven, J. S. and Warburton, T. (2008). *Nodal discontinuous Galerkin methods algorithms, analysis, and applications*. Springer, New York.

Hillewaert, K. (2013). *Development of the discontinuous Galerkin method for high-resolution, large scale CFD and acoustics in industrial geometries*. PhD thesis, Ecole polytechnique de Louvain.

Hirt, C. W. and Nichols, B. D. (1981). Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of computational physics*, 39(1):201–225.

Holmgren, H. and Kreiss, G. (2015). Towards accurate modeling of moving contact lines. *arXiv preprint arXiv:1510.06639*.

Hou, T. Y., Li, Z., Osher, S., and Zhao, H. (1997). A Hybrid Method for Moving Interface Problems with Application to the Hele–Shaw Flow. *Journal of Computational Physics*, 134(2):236–252.

Hu, C. and Shu, C. (1999). A Discontinuous Galerkin Finite Element Method for Hamilton–Jacobi Equations. *SIAM Journal on Scientific Computing*, 21(2):666–690.

Hysing, S., Turek, S., Kuzmin, D., Parolini, N., Burman, E., Ganesan, S., and Tobiska, L. (2009). Quantitative benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids*, 60(11):1259–1288.

Hysing, S.-R. and Turek, S. (2005). The Eikonal equation: numerical efficiency vs. algorithmic complexity on quadrilateral grids. In *Proceedings of ALGORITMY*, volume 2005, pages 22–31.

Javierre, E., Vuik, C., Vermolen, F. J., and Segal, A. (2005). A level set method for particle dissolution in a binary alloy. *Reports of the Department of Applied Mathematical Analysis, 05-07*.

Jeong, W. and Whitaker, R. (2008). A Fast Iterative Method for Eikonal Equations. *SIAM Journal on Scientific Computing*, 30(5):2512–2534.

Jibben, Z. and Herrmann, M. (2012). A Runge–Kutta discontinuous Galerkin conservative level set method. In *Proceedings of the Summer Program*, page 305.

Kallendorf, C. (2017). *An Eulerian discontinuous Galerkin method for the numerical simulation of interfactial transport*. Dissertation, Cuvillier Verlag, Göttingen.

Karakus, A., Warburton, T., Aksel, H., and Sert, C. (2014). Level Set Reinitialization for A High Order h Adaptive Discontinuous Galerkin Method.

Karakus, A., Warburton, T., Aksel, M. H., and Sert, C. (2016). A GPU accelerated level set reinitialization for an adaptive discontinuous Galerkin method. *Computers & Mathematics with Applications*, 72(3):755–767.

Kimmel, R. and Sethian, J. A. (1998). Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*, 95(15):8431–8435.

Klöckner, A., Warburton, T., and Hesthaven, J. S. (2011). Viscous Shock Capturing in a Time-Explicit Discontinuous Galerkin Method. *Mathematical Modelling of Natural Phenomena*, 6(3):57–83.

Klein, B. (2015). *A high-order Discontinuous Galerkin solver for incompressible and low-Mach number flows*. PhD thesis, Technische Universität.

Klein, B., Kummer, F., Keil, M., and Oberlack, M. (2013a). An extension of the SIMPLE based discontinuous Galerkin solver to unsteady incompressible flows. *To be published*.

Klein, B., Kummer, F., and Oberlack, M. (2013b). A SIMPLE based discontinuous Galerkin solver for steady incompressible flows. *Journal of Computational Physics*, 237:235–250.

Krause, D. and Kummer, F. (2017). An incompressible immersed boundary solver for moving body flows using a cut cell discontinuous Galerkin method. *Computers & Fluids*, 153:118–129.

Kummer, F. (2012). *The BoSSS Discontinuous Galerkin solver for incompressible fluid dynamics and an extension to singular equations.* Dissertation, TU Darmstadt / Fachgebiet für Strömungsdynamik.

Kummer, F. (2016). Extended discontinuous Galerkin methods for two-phase flows: the spatial discretization. *International Journal for Numerical Methods in Engineering*, 109(2):259–289.

Kummer, F., Müller, B., and Utz, T. (2018). Time integration for extended discontinuous Galerkin methods with moving domains: Time integration for XDG methods with moving domains. *International Journal for Numerical Methods in Engineering*, 113(5):767–788.

Legrain, G., Chevaugeon, N., and Dréau, K. (2012). High order X-FEM and levelsets for complex microstructures: uncoupling geometry and approximation. *Computer Methods in Applied Mechanics and Engineering*, 241:172–189.

Lehrenfeld, C. (2014). The Nitsche XFEM-DG space-time method and its implementation in three space dimensions. *arXiv:1408.2941 [cs, math]*. arXiv: 1408.2941.

Lehrenfeld, C. (2015). *On a Space-Time Extended Finite Element Method for the Solution of a Class of Two-Phase Mass Transport Problems*. PhD thesis, RWTH Aachen.

Li, B. Q. (2006). *Discontinuous finite elements in fluid dynamics and heat transfer*. Springer, London.

Li, C., Xu, C., Gui, C., and Fox, M. (2005). Level set evolution without re-initialization: a new variational formulation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 1, pages 430–436 vol. 1.

Li, C., Xu, C., Gui, C., and Fox, M. (2010). Distance Regularized Level Set Evolution and Its Application to Image Segmentation. *IEEE Transactions on Image Processing*, 19(12):3243–3254.

Li, F., Shu, C.-W., Zhang, Y.-T., and Zhao, H. (2008). A second order discontinuous Galerkin fast sweeping method for Eikonal equations. *Journal of Computational Physics*, 227(17):8191–8208.

Liu, C., Pan, Z., and Duan, J. (2013). New Algorithm for Level Set Evolution without Re-initialization and Its Application to Variational Image Segmentation. *Journal of Software*, 8(9).

Loch, E. (2013). *The level set method for capturing interfaces with applications in two-phase flow problems*. PhD thesis, RWTH Aachen.

Luddens, F., Bergmann, M., and Weynans, L. (2015). Enablers for high-order level set methods in fluid mechanics. *International Journal for Numerical Methods in Fluids*, 79(12):654–675.

Luo, S. (2013). A uniformly second order fast sweeping method for eikonal equations. *Journal of Computational Physics*, 241:104–117.

Marchandise, E. (2006). *Simulation of three-dimensional two-phase flows : coupling of a stabilized finite element method with a discontinuous level set approach*. PhD thesis, Université Catholique de Louvain.

Marchandise, E., Geuzaine, P., Chevaugeon, N., and Remacle, J.-F. (2007). A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics. *Journal of Computational Physics*, 225(1):949–974.

Massjung, R. (2012). An Unfitted Discontinuous Galerkin Method Applied to Elliptic Interface Problems. *SIAM Journal on Numerical Analysis*, 50(6):3134–3162.

McCaslin, J. O. and Desjardins, O. (2014). A localized re-initialization equation for the conservative level set method. *Journal of Computational Physics*, 262:408–426.

Meister, A. (2011). *Numerik linearer Gleichungssysteme: eine Einführung in moderne Verfahren ; mit MATLAB-Implementierungen von C. Vömel*. Vieweg, Wiesbaden.

Min, C. and Gibou, F. (2007). A second order accurate level set method on non-graded adaptive cartesian grids. *Journal of Computational Physics*, 225(1):300–321.

Min, C. and Gibou, F. (2008). Robust second-order accurate discretizations of the multi-dimensional Heaviside and Dirac delta functions. *Journal of Computational Physics*, 227(22):9686–9695.

Müller, B. (2014). *Methods for higher order numerical simulations of complex inviscid fluids with immersed boundaries*. PhD Thesis, Technische Universität, Darmstadt.

Müller, B., Krämer-Eis, S., Kummer, F., and Oberlack, M. (2017). A high-order discontinuous Galerkin method for compressible flows with immersed boundaries. *International Journal for Numerical Methods in Engineering*, 110(1):3–30.

Müller, B., Kummer, F., and Oberlack, M. (2013). Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *International Journal for Numerical Methods in Engineering*, 96(8):512–528.

Müller, B., Kummer, F., Oberlack, M., and Wang, Y. (2012). Simple multidimensional integration of discontinuous functions with application to level set methods. *International Journal for Numerical Methods in Engineering*, 92(7):637–651.

Moon, P. and Spencer, D. E. (2012). *Field theory handbook: including coordinate systems, differential equations and their solutions*. Springer.

Moroney, T. J., Lusmore, D. R., McCue, S. W., and Sean McElwain, D. L. (2017). Extending fields in a level set method by solving a biharmonic equation. *Journal of Computational Physics*.

Mousavi, R. (2014). *Level Set Method for Simulating the Dynamics of the Fluid-Fluid Interfaces: Application of a Discontinuous Galerkin Method*. PhD thesis, TU Darmstadt, TU Darmstadt.

Mulder, W., Osher, S., and Sethian, J. A. (1992). Computing interface motion in compressible gas dynamics. *Journal of Computational Physics*, 100(2):209–228.

Olshanskii, M. A., Reusken, A., and Xu, X. (2014). A stabilized finite element method for advection-diffusion equations on surfaces. *IMA Journal of Numerical Analysis*, 34(2):732–758.

Osher, S. and Fedkiw, R. P. (2001). Level Set Methods: An Overview and Some Recent Results. *Journal of Computational Physics*, 169(2):463–502.

Osher, S. and Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49.

Owkes, M. and Desjardins, O. (2013). A discontinuous Galerkin conservative level set scheme for interface capturing in multiphase flows. *Journal of Computational Physics*, 249:275–302.

Pawel Stapór (2016). A two-dimensional simulation of solidification processes in materials with thermo-dependent properties using XFEM. *International Journal of Numerical Methods for Heat & Fluid Flow*, 26(6):1661–1683.

Peng, D., Merriman, B., Osher, S., Zhao, H., and Kang, M. (1999). A PDE-Based Fast Local Level Set Method. *Journal of Computational Physics*, 155(2):410–438.

Persson, P.-O. and Peraire, J. (2006). Sub-Cell Shock Capturing for Discontinuous Galerkin Methods. In *Aerospace Sciences Meetings*, Reno, Nevada, USA. American Institute of Aeronautics and Astronautics.

Pochet, F., Hillewaert, K., Geuzaine, P., Remacle, J.-F., and Marchandise, m. (2013). A 3d strongly coupled implicit discontinuous Galerkin level set-based method for modeling two-phase flows. *Computers & Fluids*, 87:144–155.

Qian, J., Zhang, Y., and Zhao, H. (2007). Fast Sweeping Methods for Eikonal Equations on Triangular Meshes. *SIAM Journal on Numerical Analysis*, 45(1):83–107.

Rasthofer, U. (2015). *Computational Multiscale Methods for Turbulent Single and Two-Phase Flows*. PhD thesis, München, Technische Universität München, Diss., 2015.

Reed, W. H. and Hill, T. R. (1973). Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Lab., Los Alamos, N. Mex.(USA).

Reusken, A., Xu, X., and Zhang, L. (2017). Finite element methods for a class of continuum models for immiscible flows with moving contact lines. *International Journal for Numerical Methods in Fluids*, 84(5):268–291.

Rivière, B. (2008). *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. Frontiers in applied mathematics. SIAM, Society for Industrial and Applied Mathematics, Philadelphia, PA. OCLC: ocn226292048.

Russo, G. and Smereka, P. (2000). A remark on computing distance functions. *Journal of Computational Physics*, 163(1):51–67.

Sauerland, H. (2013). *An XFEM Based Sharp Interface Approach for Two-Phase and Free-Surface Flows*. PhD thesis, RWTH Aachen, Aachen.

Saye, R. (2014). High-order methods for computing distances to implicitly defined surfaces. *Communications in Applied Mathematics and Computational Science*, 9(1):107–141.

Saye, R. (2017). Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid-structure interaction, and free surface flow: Part I. *Journal of Computational Physics*, Volume 344:647–682.

Schenk, O. and Gärtner, K. (2004). Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems*, 20(3):475–487.

Schulte, T. (2012). Development of Numerical Methods for the Simulation of a Class of Osmotic Problems in Moving Domains. Master's thesis, RWTH Aachen.

Schwarz, H.-R. and Köckler, N. (2013). *Numerische Mathematik*. Springer-Verlag.

Shahbazi, K. (2005). An explicit expression for the penalty parameter of the interior penalty method. *Journal of Computational Physics*, 205(2):401–407.

Shahbazi, K., Fischer, P. F., and Ethier, C. R. (2007). A high-order discontinuous Galerkin method for the unsteady incompressible Navier–Stokes equations. *Journal of Computational Physics*, 222(1):391–407.

Shakoor, M., Scholtes, B., Bouchard, P.-O., and Bernacki, M. (2015). An efficient and parallel level set reinitialization method – Application to micromechanics and microstructural evolutions. *Applied Mathematical Modelling*, 39(23–24):7291–7302.

Sonntag, M. and Munz, C.-D. (2017). Efficient Parallelization of a Shock Capturing for Discontinuous Galerkin Methods using Finite Volume Sub-cells. *Journal of Scientific Computing*, 70(3):1262–1289.

Spurk, J. (2013). *Strömungslehre: Einführung in die Theorie der Strömungen*. Springer-Verlag.

Sussman, M. and Hussaini, M. Y. (2003). A Discontinuous Spectral Element Method for the Level Set Equation. *Journal of Scientific Computing*, 19(1-3):479–500.

Sussman, M., Smereka, P., and Osher, S. (1994). A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, 114(1):146–159.

Thomée, V. (2006). *Galerkin finite element methods for parabolic problems*. Springer, Berlin.

Tsai, Y., Cheng, L., Osher, S., and Zhao, H. (2003). Fast Sweeping Algorithms for a Class of Hamilton–Jacobi Equations. *SIAM Journal on Numerical Analysis*, 41(2):673–694.

Tugurlan, M. C. (2008). *Fast marching methods-parallel implementation and analysis*. PhD thesis, Louisiana State University, Baton Rouge.

Utz, T., Kallendorf, C., Kummer, F., Müller, B., and Oberlack, M. (2017a). An Extended Discontinuous Galerkin Framework for Multiphase Flows. In *Transport Processes at Fluidic Interfaces*, Advances in Mathematical Fluid Mechanics, pages 65–91. Birkhäuser, Cham.

Utz, T. and Kummer, F. (2017). A high-order discontinuous Galerkin method for extension problems. *International Journal for Numerical Methods in Fluids*.

Utz, T., Kummer, F., and Oberlack, M. (2017b). Interface-preserving level-set reinitialization for DG-FEM. *International Journal for Numerical Methods in Fluids*, 84(4):183–198.

Wang, Y. and Oberlack, M. (2011). A thermodynamic model of multiphase flows with moving interfaces and contact line. *Continuum Mechanics and Thermodynamics*, 23(5):409–433.

Weller, S. and Bänsch, E. (2017). Time Discretization for Capillary Flow: Beyond Backward Euler. In *Transport Processes at Fluidic Interfaces*, pages 121–143. Springer.

Wu, L. and Zhang, Y.-T. (2014). A Third Order Fast Sweeping Method with Linear Computational Complexity for Eikonal Equations. *Journal of Scientific Computing*, 62(1):198–229.

Xia, Y., Wong, S. C., Zhang, M., Shu, C.-W., and Lam, W. H. K. (2008). An efficient discontinuous Galerkin method on triangular meshes for a pedestrian flow model. *International Journal for Numerical Methods in Engineering*, 76(3):337–350.

Zalesak, S. T. (1979). Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of computational physics*, 31(3):335–362.

Zhang, Y., Chen, S., Li, F., Zhao, H., and Shu, C. (2011). Uniformly Accurate Discontinuous Galerkin Fast Sweeping Methods for Eikonal Equations. *SIAM Journal on Scientific Computing*, 33(4):1873–1896.

Zhao, H. (2005). A fast sweeping method for Eikonal equations. *Mathematics of computation*, 74(250):603–627.

Zhao, H.-K., Chan, T., Merriman, B., and Osher, S. (1996). A Variational Level Set Approach to Multiphase Motion. *Journal of Computational Physics*, 127(1):179–195.

Zoby, M. R. G., Kronenburg, A., Navarro-Martinez, S., and Marquis, A. J. (2012). Assessment of Conventional Droplet Evaporation Models for Spray Flames. In

Nagel, W. E., Kröner, D. B., and Resch, M. M., editors, *High Performance Computing in Science and Engineering '11*, pages 209–227. Springer Berlin Heidelberg, Berlin, Heidelberg.

A Conservation of the signed distance property using an extension velocity

If the velocity field $\vec{u}$ for the level set advection (7.2) is chosen as the extension of a field (7.1b), the level set will retain its signed distance property for all time. This follows the same argument as Adalsteinsson and Sethian (1999) for a scalar extension velocity formulation. Let's revisit both equations:

$$\frac{\partial \varphi}{\partial t} + \vec{u} \cdot \nabla \varphi = 0 \qquad \text{(see (7.2))}$$

$$\nabla \vec{u} \cdot \nabla \varphi = 0 \qquad \text{(see (7.1b))}$$

By applying the gradient operator to equation (7.2), it becomes

$$\frac{\partial \nabla \varphi}{\partial t} + \nabla \vec{u} \cdot \nabla \varphi + \vec{u} \cdot \nabla \left(\nabla \varphi \right) = 0, \tag{A.1}$$

where the second term is zero due to the extension equation (7.1b).

Multiplying the result by the level set gradient $\nabla \varphi$ gives

$$\frac{\partial \nabla \varphi \cdot \nabla \varphi}{\partial t} + \vec{u} \cdot \nabla \left(\nabla \varphi \cdot \nabla \varphi \right) = 0, \tag{A.2}$$

or

$$\frac{\partial |\nabla \varphi|^2}{\partial t} + \vec{u} \cdot \nabla \left(|\nabla \varphi|^2 \right) = 0 \tag{A.3}$$

If this motion algorithm is initialized using a signed-distance field, i.e.

$$\| \nabla \varphi \| |_{t=0} = 1 \tag{A.4}$$

then the left hand side of (A.2) is zero for al times

$$\frac{\partial \nabla \varphi \cdot \nabla \varphi}{\partial t} = 0 \tag{A.5}$$

and thus the signed distance property is conserved.

Curriculum vitae

Thomas Utz

Birthday:	June 2nd, 1987
Birthplace:	Augsburg
Nationality:	German

Education:

1997 - 2006	Gymnasium bei St. Anna, Augsburg High school degree
2007 - 2010	Corporate studies, mechanical engineering DHBW Stuttgart in cooperation with MTU Aero Engines, Munich Degree: B.Eng.
2011 - 2013	Graduate studies mechanical and process engineering TU Darmstadt Degree: M.Sc.

Professional and Scientific Experience:

2007 - 2010	Trainee mechanical engineering MTU Aero Engines, Munich
2010 - 2011	Repair design engineer MTU Aero Engines, Munich
2014 - 2018	PhD researcher and teaching assistant Chair of Fluid Dynamics, TU Darmstadt Collegiate, Graduate School of Computational Engineering