Master Thesis

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Solving the Allen-Cahn Equation Using Discontinuous Galerkin Methods with a Domain Decomposition Approach

Simon Wenchel
Department of Mathematics
Chair of Mathematical Fluid Mechanics

Supervised by Prof. Dr. Christian Klingenberg

Submission
11. March 2025

www.uni-wuerzburg.de

Abstract

The discontinuous Galerkin (DG) methods have gained popularity in recent years for solving partial differential equations, due to their ability to handle discontinuities, local conservation and flexibility in enforcing boundary conditions. In the thesis, a DG-based domain decomposition approach is introduced and analyzed for solving the Poisson equation and the nonlinear Allen-Cahn equation.

Domain decomposition methods enable naturally efficient parallelization by solving smaller subproblems independently and obtaining the global solution iteratively by exchanging information between subdomains through interface conditions. This thesis focuses on the additive Schwarz method (ASM) and the optimized Schwarz method (OSM) when solving the subproblems with DG methods. In optimized Schwarz methods, Dirichlet interface conditions are replaced with Robin boundary conditions, which can be done naturally with the flux-based formulations of DG methods, improving the convergence of the iterative solution process. The Poisson equation is used to validate the combination of both methods in a linear setting, while the Allen-Cahn equation tests the performance and robustness in a more complex nonlinear scenario.

The thesis introduces a detailed theoretical derivation of the DG formulation of the Poisson equation and its implementation. Schwarz-based domain decomposition methods are then introduced, before combining them with DG methods. Numerical experiments are conducted to evaluate the convergence properties and accuracy of the introduced combination and compare it to pure DG methods. The discretization of the Allen-Cahn equation is then presented and numerical experiments also compare the performance of the combination to pure DG methods.

The results show that introducing the Robin boundary conditions, with the help of the DG formulation for the normal derivative improves the convergence of the Schwarz methods and makes it possible to solve the Poisson equation for non-overlapping subdomains. The overlapping additive Schwarz method shows good results for the application to the Allen-Cahn equation. The results make DG-based domain decomposition methods a promising approach for solving large-scale partial differential equations (PDEs).

Further research could focus on the application of interface conditions for the Allen-Cahn equation to improve the convergence with Robin boundary conditions and solve the problem for non-overlapping subdomains. Additionally, multilevel domain decomposition methods and local mesh refinement could be investigated to improve the DG-based domain decomposition approach further.

Zusammenfassung

Die Discontinuous Galerkin (DG) Methode hat in den letzten Jahren zunehmend an Bedeutung für die Lösung partieller Differentialgleichungen gewonnen, da sie den Umgang mit Unstetigkeiten ermöglicht, lokale Erhaltung gewährleistet und Flexibilität in der Wahl der Randbedingungen bietet. In dieser Arbeit wird eine DG-basierte Domain Decomposition Methode entwickelt und analysiert, die zur Lösung der Poisson Gleichung sowie der nichtlinearen Allen-Cahn Gleichung eingesetzt wird.

Domain Decomposition Methoden ermöglichen eine effiziente Parallelisierung, indem das globale Problem in kleinere Teilprobleme unterteilt wird, die unabhängig voneinander gelöst werden. Die globale Lösung wird iterativ durch den Informationsaustausch zwischen den Teilgebieten über Schnittstellenbedingungen bestimmt. Diese Arbeit untersucht die Additive Schwarz Methode und die Optimized Schwarz Methode in Kombination mit DG Methoden. Bei der Optimized Schwarz Methode werden Dirichlet Randbedingungen durch Robin Randbedingungen ersetzt, was sich durch die flussbasierte Formulierung der DG Methode auf natürliche Weise implementieren lässt und die Konvergenz des iterativen Lösungsprozesses verbessert. Die Poisson Gleichung dient als Validierungsbeispiel der Methode im linearen Kontext, während die Allen-Cahn Gleichung die Effizienz und Robustheit in einem komplexeren, nichtlinearen Problem untersucht.

Die Ergebnisse zeigen, dass die Einführung von Robin Randbedingungen unter Berücksichtigung der DG-Formulierung der Normalableitung die Konvergenz der Schwarz Methoden verbessert und es ermöglicht, die Poisson Gleichung für nicht überlappende Teilgebiete zu lösen. Die überlappende Additive Schwarz Methode zeigt vielversprechende Ergebnisse für die Anwendung auf die Allen-Cahn Gleichung. Diese Ergebnisse machen den DG-basierten Domain Decomposition Ansatz zu einer vielversprechenden Methode für großskalige Simulationen zur Lösung partieller Differentialgleichungen.

Zukünftige Arbeiten könnten sich auf die Optimierung der Schnittstellenbedingungen für die Allen-Cahn Gleichung konzentrieren, um die Konvergenz mit Robin Randbedingungen zu verbessern und das Problem für nicht-überlappende Teilgebiete zu lösen. Darüber hinaus könnten Multilevel Domain Decomposition Methoden und lokale Gitterverfeinerung untersucht werden, um den DG-basierten Domain Decomposition Ansatz weiter zu verbessern.

Acknowledgement

I would like to express my deepest gratitude to my supervisors, Prof. Dr. Klingenberg, Prof. Dr. Bolten, Dr. Muralikrishnan and M.Sc. Babamehdi for their guidance and support throughout the course of this thesis.
I am grateful to Prof. Dr. Klingenberg for kindly accepting my thesis topic and making this external thesis possible. His willingness to supervise my work and his encouragement during our meetings provided me with confidence and motivation.
I would like to thank Prof. Dr. Bolten for also making this project possible by serving as the second reviewer. His regular feedback and constructive suggestions contributed significantly to the progress of this thesis.
A special thanks to Dr. Sriram Muralikrishnan and M.Sc. Mehdi Babamehdi for their continuous support and guidance during the course of this thesis. Their willingness to take the time to assist whenever I needed help. Their patience in explaining complex concepts was invaluable for this work.
Finally, I would like to thank my family and friends for their support and encouragement, as well as all colleagues who made my time at JSC a memorable experience.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Related Work . . . 2
- 1.3 Outline . . . 2

2 Mathematical Preliminaries **5**
- 2.1 Vector Notations . . . 5
- 2.2 Important Vector Spaces . . . 5
- 2.3 Sobolev Spaces . . . 6
- 2.4 Broken Sobolev Spaces . . . 8
- 2.5 Trace Theorem . . . 8
- 2.6 Approximation Properties . . . 9
- 2.7 Green's Formula . . . 9

3 Discontinuous Galerkin Methods **11**
- 3.1 The DG Formulation of the Poisson Equation . . . 11
  - 3.1.1 Model Problem . . . 11
  - 3.1.2 Weak Formulation . . . 11
  - 3.1.3 Lax-Milgram Theorem . . . 12
  - 3.1.4 Jump and Average Operators . . . 13
  - 3.1.5 Discontinuous Galerkin Formulation . . . 13
- 3.2 Implementation . . . 14
  - 3.2.1 Mesh Triangulation . . . 14
  - 3.2.2 Discontinuous Finite Element Space . . . 16
  - 3.2.3 Basis Functions . . . 16
  - 3.2.4 Discretisation of the Weak Formulation Using Continuous FEM with Linear Basis Functions . . . 16
  - 3.2.5 Discretisation of the DG Formulation Using Discontinuous Basis Functions . . . 17
  - 3.2.6 Reference Element . . . 19
  - 3.2.7 Lagrangian Basis Functions . . . 23
  - 3.2.8 Numerical Quadratures . . . 24
  - 3.2.9 Local Matrices and Right-Hand Side . . . 25
  - 3.2.10 Global System . . . 30
  - 3.2.11 Matrix-free Implementation . . . 30
- 3.3 Error Estimates . . . 30
  - 3.3.1 Error Estimates in the Energy Norm . . . 31
  - 3.3.2 Error Estimates in the L^2 Norm . . . 32
- 3.4 Numerical Results . . . 32
  - 3.4.1 Error Estimates in the Energy Norm . . . 33
  - 3.4.2 Error Estimates in the L^2 Norm . . . 33

4 Domain Decomposition Methods **35**
4.1 Additive Schwarz Methods . . . 35
4.2 Optimized Schwarz Methods . . . 37

5 Combination of Discontinuous Galerkin and Domain Decomposition Methods **39**
5.1 Partitioning of the Domain . . . 39
5.2 Boundary Conditions of the Subdomains . . . 39
5.2.1 Boundary Value Approximation on the Interior Boundaries . . . 40
5.2.2 Robin Boundary Conditions (OSM) . . . 40
5.3 Implementation of the Schwarz Methods . . . 43
5.4 Numerical Results . . . 44
5.4.1 Progress of the Solution in the Optimized Schwarz Method . . . 45
5.4.2 Comparing the Convergence of the ASM and OSM for Overlapping Subdomains . . . 47
5.4.3 Convergence of the Optimized Schwarz Method with Varying Penalty Parameters . . . 47
5.4.4 Comparison of Interface Condition Implementation Approaches . . . 49
5.4.5 Optimizing the Convergence of the Optimized Schwarz Method with Varying Robin Boundary Condition and Penalty Parameter . . . 50

6 Allen-Cahn Equation **51**
6.1 Numerical Solution of the Allen-Cahn Equation . . . 52
6.1.1 Time Discretization . . . 52
6.1.2 Discretization of the Laplacian Using the Discontinuous Galerkin Method . . . 53
6.1.3 Implicit-Explicit Scheme (IMEX) Applied to the Allen-Cahn Equation 53
6.1.4 Fully Implicit Scheme . . . 55
6.1.5 Domain Decomposition . . . 56
6.2 Numerical Results . . . 56
6.2.1 Numerical Radius Determination of the Phase Separation . . . 57
6.2.2 Comparing the Numerical and Analytical Radii . . . 58

7 Conclusion **61**

List of Figures **63**

List of Tables **65**

Bibliography **67**

A Appendix **69**
A.1 Explanation of Code Structure . . . 69
A.1.1 Mesh . . . 69
A.1.2 SubMesh . . . 70
A.1.3 DG . . . 70
A.1.4 Domain Decomposition . . . 70

1 Introduction

1.1 Motivation

Numerical simulations of partial differential equations (PDEs) are essential in physics and engineering. Large-scale problems require dedicated, efficient computational methods, especially for complex geometries. Domain decomposition methods were first introduced by Schwarz in 1870, without the background of numerical simulations, but these methods have gained popularity in the last few decades with the increasing need for parallel computing. In domain decomposition methods, the domain is split into smaller subdomains and solved independently, which allows for parallelization. The solution with domain decomposition methods is obtained iteratively by exchanging information at the interfaces of the subdomains. When solving the subproblems with classical finite element methods (FEM), i.e., continuous Galerkin methods, continuity between the subdomains is enforced, making domain decomposition methods more challenging. Discontinuous Galerkin methods (DGM) naturally allow for discontinuities between the elements, making them well-suited for combination with domain decomposition methods. DG methods maintain high-order accuracy and stability, with local conservation properties, through the use of flux-based formulations at the element interfaces. Discontinuous Galerkin methods also provide flexibility in enforcing boundary conditions, as the interface conditions do not need to be strongly imposed and can be handled in a weak sense by the numerical fluxes. This is crucial for domain decomposition methods, as the interface conditions at the subdomain boundaries strongly influence the convergence of the iterative solution process. The combination of domain decomposition and discontinuous Galerkin methods is a promising approach for solving large-scale PDEs on complex geometries with high accuracy.
The Poisson equation is a simple elliptic PDE appearing in many applications and serves as a good test case for evaluating the combination of these methods. The Allen-Cahn equation is a reaction-diffusion equation used in separation models in material science. It is a nonlinear PDE with a nonlinear perturbation term, making it a challenging test case for numerical methods.
This thesis explores how the discontinuous Galerkin method can be effectively combined with domain decomposition methods to solve both linear and nonlinear PDEs, focusing on the Poisson and Allen-Cahn equations. In particular, the role of the interface conditions is investigated, and numerical experiments are performed to evaluate the accuracy and performance of the methods. The performance of this combination is compared to pure discontinuous Galerkin methods when solving the nonlinear Allen-Cahn equation.

1.2 Related Work

Since the introduction of domain decomposition methods by Schwarz in 1870 for solving partial differential equations on complex geometries in [1], a century passed until Lions introduced the optimized Schwarz method for handling non-overlapping subdomains in [2]. With the rise of parallel computing, domain decomposition methods were rediscovered, and their efficient application in large-scale simulations was investigated in [3]. At the same time, the discontinuous Galerkin method gained interest as an alternative to classical finite element methods, as it allows for discontinuities and flexibility for complex geometries while maintaining high-order accuracy and stability. More recently, research has focused on combining domain decomposition with DG methods, as their properties complement each other. In [4], Gander analyzed optimized Schwarz methods for symmetric positive problems, highlighting their improvements over additive Schwarz methods and showing their relation to finite element tearing and interconnecting (FETI) methods. While FETI methods enforce continuity between subdomains, transmission conditions to improve the convergence of the optimized Schwarz method were developed.
Building on these ideas, Antonetti et al. [5] proposed and introduced the optimized Schwarz method for DG methods, showing that optimized Schwarz methods are scalable and effective preconditioners for solving DG discretizations with GMRES.
A further step was taken by Gander and Hajian [6], who introduced a hybridizable DG method, which introduces additional degrees of freedom on the element interfaces to enforce continuity. They showed that the penalty parameter of the hybridizable DG method can lead to slow convergence in classical Schwarz methods. To resolve this, they introduced a modified Schwarz solver weakening the positive definiteness, resulting in faster convergence.
Recent research continues to refine these methods. In 2023, Lu et al. [7] published an article on two-level Schwarz methods for hybridizable DG methods, where a coarse grid correction is used to improve the scalability and robustness of DG methods.

The optimal choice of interface conditions in relation to the penalty parameter of the DG method is still an ongoing research topic. In this thesis, we investigate the interaction of these parameters numerically and show that DG-based domain decomposition methods can be effectively used to solve linear and nonlinear PDEs.

1.3 Outline

The outline of the thesis is as follows:
Chapter 2 provides the necessary mathematical background to introduce the discontinuous Galerkin methods. It introduces the fundamental concepts of Sobolev spaces, approximation properties, and Green's Formula.

Before deriving the DG formulation of the Poisson equation in Chapter 3, important concepts of the DG method are introduced, such as the weak formulation and the jump and average operators. A special focus on the implementational aspects of the DG method is given, and it is shown how to discretize the Poisson equation using the DG method. The theoretical convergence properties of the DG methods are then compared to the implementation in the numerical experiments.

Chapter 4 introduces the additive and optimized Schwarz methods for domain decomposition. The standard iterative scheme is presented, and the differences in the interface conditions for additive and optimized Schwarz methods are introduced.

Finally, the integration of DG methods with domain decomposition methods is discussed in Chapter 5. It is shown how the interface conditions can be implemented in the DG

method to be used in additive and optimized Schwarz methods. Numerical experiments are conducted to evaluate the performance of the combination of both methods in comparison to the pure DG method and to analyze how different parameters of the interface conditions and DG influence each other.

In Chapter 6, the extension to the nonlinear Allen-Cahn equation is presented. The Allen-Cahn equation is discretized by two time-stepping schemes, an implicit-explicit (IMEX) scheme and a fully implicit scheme. The DG formulation of the Laplacian from previous chapters is used to discretize the equation in space. Algorithmic details on how to apply the combination of both methods are given. Numerical experiments are performed on the approximation quality of these methods for evaluating the phase separation radius.

2 Mathematical Preliminaries

To effectively apply discontinuous Galerkin methods for solving partial differential equations, certain mathematical preliminaries must be introduced. In this section, we provide the necessary mathematical background to understand these methods. The following definitions are primarily based on the book *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations* by Rivière [8].

2.1 Vector Notations

The following notations will be used throughout. Let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ be column vectors of size n. The dot product of these two vectors is defined as $\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i$.
The gradient of a scalar function $u : \mathbb{R}^n \to \mathbb{R}$ is a vector, and the divergence of a vector function $\mathbf{v} : \mathbb{R}^n \to \mathbb{R}^n$ is a scalar.

$$\nabla u = \left(\frac{\partial u}{\partial x_i} \right)_{i=1,\ldots,n} \quad \text{and} \quad \nabla \cdot \mathbf{v} = \sum_{i=1}^{n} \frac{\partial v_i}{\partial x_i}$$

2.2 Important Vector Spaces

Throughout this thesis, the domain Ω is assumed to be a bounded quadrilateral domain in $\mathbb{R}^n$. To introduce relevant function spaces, we first define the space of square-integrable functions, $L^2(\Omega)$:

$$L^2(\Omega) = \{v : \int_\Omega v^2 < \infty\}.$$

Definition 2.1. *Let V be a vector space and K a field. A **symmetric bilinear form** is a mapping $a : V \times V \to K$, that is linear in both arguments and satisfies symmetry. The mapping $a : V \times V \to K$ is a bilinear form if and only if:*

$$\begin{aligned} a(u+v, w) &= a(u, w) + a(v, w) \quad , \\ a(\alpha u, v) &= \alpha a(u, v) \quad \textit{and} \\ a(u, v+w) &= a(u, v) + a(u, w) \quad , \\ a(u, \alpha v) &= \alpha a(u, v) \end{aligned}$$

for all $\alpha \in \mathbb{R}$,$u, v, w \in V$, $a \in K$. It is symmetric if:

$$a(u, v) = a(v, u) \quad \textit{for all } u, v \in V.$$

In this thesis, the field K will be the space of real numbers, $\mathbb{R}$.
Let V be a vector space, and let $a : V \times V \to \mathbb{R}$ be a symmetric bilinear form. The space V is a **normed vector space** with the norm $||\cdot||_V = (a(\cdot,\cdot))^{1/2}$.
Furthermore, if V is equipped with an inner product and is complete with respect to the norm $||\cdot||_V$, it is a **Hilbert space**. A space is said to be **complete** if every Cauchy sequence in the space converges to a limit within the space. The **dual space** of V, denoted by V', is the space of all continuous linear functionals on V.

With these definitions, we now state that the space $L^2(\Omega)$, equipped with the inner product $\langle u, v\rangle = \int_\Omega uv$, is a Hilbert space with the norm

$$||v||_{L^2(\Omega)} = \left(\int_\Omega v^2\right)^{1/2}.$$

Extending the functions u and v to vector functions $\mathbf{u} = (u_i)_{i=1,\dots,N}$ and $\mathbf{v} = (v_i)_{i=1,\dots,N}$, the inner product and norm become

$$\langle \mathbf{u}, \mathbf{v}\rangle = \sum_{i=1}^{N} \int_\Omega u_i v_i \quad \text{and} \quad ||\mathbf{v}||_{L^2(\Omega)} = \left(\sum_{i=1}^{N} \int_\Omega v_i^2\right)^{1/2}.$$

Definition 2.2. *The **support** of a continuous function $v : \Omega \to \mathbb{R}$ is the closure of the set of points where v is nonzero. If this support is bounded and contained within Ω, the function is said to have **compact support**.*

Let $\mathcal{D}(\Omega)$ denote the space of smooth functions with compact support in Ω. The dual space of $\mathcal{D}(\Omega)$ is the space of distributions, denoted $\mathcal{D}'(\Omega)$. Let $\alpha = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}^d$ be a multi-index with $|\alpha| = \sum_{i=1}^d \alpha_i$. The **distributional derivative** $D^\alpha v \in \mathcal{D}'(\Omega)$ is defined by

$$\forall \phi \in \mathcal{D}(\Omega), \qquad \langle D^\alpha v, \phi\rangle = (-1)^{|\alpha|}\langle v, D^\alpha \phi\rangle = (-1)^{|\alpha|} \int_\Omega v(x) \frac{\partial^{|\alpha|}\phi}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}\, dx.$$

For a single index $\alpha = (1)$, we have

$$\forall \phi \in \mathcal{D}(\Omega), \qquad \langle D^\alpha v, \phi\rangle = \frac{\partial v}{\partial x_1} = -\int_\Omega v(x) \frac{\partial \phi}{\partial x_1}\, dx.$$

These definitions are necessary to introduce Sobolev spaces in the next section. In the context of DG methods, Sobolev spaces, particularly broken Sobolev spaces, play a crucial role, which will be introduced later in this thesis. They provide the framework for the weak formulation of the partial differential equation, which is used to derive the method and compute a solution. The weak formulation represents the PDE in a variational form, which simplifies finding a solution. Later in the thesis, we will define the weak formulation and its solution and discuss its relation to the strong solution of the original PDE.

2.3 Sobolev Spaces

Before introducing a more general definition of Sobolev spaces, we first define the Sobolev space H^1 to better understand the structure of the general case. The **Sobolev space** $H^1(\Omega)$ consists of functions $v \in L^2(\Omega)$ whose partial derivatives $\frac{\partial v}{\partial x_i}$ exist and are also square-integrable. Specifically,

$$H^1(\Omega) = \{v \in L^2(\Omega) : \frac{\partial v}{\partial x_i} \in L^2(\Omega) \text{ for } i = 1, \dots, n\}.$$

It can be shown that if $v \in L^2(\Omega)$, then v can be identified with a distribution in the following sense:

$$\forall \phi \in \mathcal{D}(\Omega), \qquad v(\phi) = \int_\Omega v\phi \, dx.$$

Therefore, if $v \in H^1(\Omega)$, its derivative satisfies

$$\forall \phi \in \mathcal{D}(\Omega), \qquad \frac{\partial v}{\partial x_i}(\phi) = \int_\Omega \frac{\partial v}{\partial x_i}\phi \, dx = -\int_\Omega v\frac{\partial \phi}{\partial x_i} \, dx.$$

This allows us to formulate:

$$H^1(\Omega) = \{v \in L^2(\Omega) : \nabla v \in \left(L^2(\Omega)\right)^d\}.$$

Now we define the **Sobolev space** $H^s(\Omega)$ for $s \in \mathbb{N}$ using the distributional derivative:

$$H^s(\Omega) = \{v \in L^2(\Omega) : D^\alpha v \in L^2(\Omega) \quad \forall 0 \leq |\alpha| \leq s\}.$$

To better understand this formulation, consider the Sobolev space $H^2(\Omega)$ in two dimensions:

$$H^2(\Omega) = \left\{ v \in H^1(\Omega) : \frac{\partial^2 v}{\partial x_1^2}, \frac{\partial^2 v}{\partial x_1 \partial x_2}, \frac{\partial^2 v}{\partial x_2^2} \in L^2(\Omega) \right\}.$$

The **Sobolev norm** associated with the Sobolev space $H^s(\Omega)$ is defined as:

$$||v||_{H^s(\Omega)} = \left(\sum_{|\alpha| \leq s} ||D^\alpha v||^2_{L^2(\Omega)} \right)^{1/2}.$$

The associated **Sobolev seminorm** is defined as:

$$|v|_{H^s(\Omega)} = ||\nabla^s v||_{H^s(\Omega)} = \left(\sum_{|\alpha| = s} ||D^\alpha v||^2_{L^2(\Omega)} \right)^{1/2}.$$

The seminorm considers only the highest-order derivatives, which may lead to a loss of definiteness. For example, if v is a polynomial of degree less than s, its highest-order derivatives $D^\alpha v$ vanish, while v itself is not the zero function.

It is possible to extend Sobolev spaces beyond integer indices to fractional or even real values. The space $H^{s+1/2}(\Omega)$, where s is an integer, can be obtained by interpolating between $H^s(\Omega)$ and $H^{s+1}(\Omega)$. The K-interpolation method from [9] can be used to define Sobolev spaces with real indices. In this method, a function $v \in H^s(\Omega)$ is decomposed into $v = v_1 + v_2$, where $v_1 \in H^s(\Omega)$ and $v_2 \in H^{s+1}(\Omega)$. Then, for $t \in \mathbb{R}$, the Kernel is defined as

$$K(v,t) = \left(\inf_{v_1 + v_2 = v} (||v_1||^2_{H^s(\Omega)} + t^2 ||v_2||^2_{H^{s+1}(\Omega)}) \right)^{1/2}.$$

Definition 2.3. *A space V equipped with a norm $||\cdot||_V$ is the **completion** of a subset W if, for any $v \in V$ and any $\delta > 0$, there exists a $w \in W$ such that $||v - w||_V < \delta$.*

The space $H^{s+1/2}(\Omega)$ is the completion of $H^{s+1}(\Omega)$ with respect to the norm

$$||v||_{H^{s+1/2}(\Omega)} = \left(\int_0^\infty t^{-2} K^2(v,t) \, dt \right)^{1/2}.$$

2.4 Broken Sobolev Spaces

Broken Sobolev spaces are an extension of classical Sobolev spaces and are the key function spaces in the context of DG methods. In DG methods, we encounter functions that may not be continuous across the boundaries of their elements or subdomains. Therefore, we *break* the Sobolev spaces into individual elements while preserving their properties of the Sobolev spaces.
Let $\mathcal{T}_h$ be a partition of the polygonal domain Ω into elements E_i, with $i = 1, \ldots, N$. The elements in 2D are triangles or quadrilaterals. The intersection of two elements is either empty or a common edge or face e. This partition is called a conforming mesh, where h is the maximum diameter of the elements. For simplicity, we consider a uniform mesh where all elements have the same size h. The broken Sobolev space $H^s(\Omega)$ is then defined as:

$$H^s(\Omega) = \{v \in L^2(\Omega) : \forall E_i \in \mathcal{T}_h, v|_{E_i} \in H^s(E_i)\}.$$

This space is equipped with the **broken Sobolev norm**:

$$||v||_{H^s(\mathcal{T}_h)} = \left(\sum_{E_i \in \mathcal{T}_h} ||v||^2_{H^s(E_i)} \right)^{1/2}.$$

and the **broken gradient seminorm**, which we will use later:

$$||v||_{H^0(\mathcal{T}_h)} = \left(\sum_{E_i \in \mathcal{T}_h} ||\nabla v||^2_{L^2(E_i)} \right)^{1/2}. \tag{2.1}$$

With the help of these spaces, we can now define the weak formulation of PDEs. In the next section, we will explore how Sobolev spaces are used to handle the boundary conditions of a PDE.

2.5 Trace Theorem

To define boundary conditions for Sobolev functions, we introduce the so-called trace operators, which describe the behavior of these functions on the boundary of the domain. This is necessary because Sobolev functions may not be smooth everywhere, but they still have a well-defined behavior on the boundary in a weak or distributional sense. The trace operators provide a tool to extract these boundary values and impose boundary conditions in the weak formulation.

Theorem 2.1. *Let Ω be a bounded domain with a polygonal boundary $\partial\Omega$ and outward normal vector $\mathbf{n}$. There exist trace operators γ_0 and γ_1:*

$$\gamma_0 : H^s(\Omega) \to H^{s-1/2}(\partial\Omega) \text{ for } s > 1/2 \quad \text{and}$$
$$\gamma_1 : H^s(\Omega) \to H^{s-3/2}(\partial\Omega) \text{ for } s > 3/2,$$

which extend the boundary values and boundary normal derivatives of functions in $H^s(\Omega)$. The trace operators are surjective, and if $v \in \mathcal{C}^1(\overline{\Omega})$, where $\overline{\Omega}$ denotes the closure of the domain Ω,

$$\gamma_0 v = v|_{\partial\Omega},$$
$$\gamma_1 v = \nabla v \cdot \mathbf{n}.$$

The proof and more details on the Boundary Trace Theorem can be found in [10]. The subspace of $H^s(\Omega)$, with $s > 1/2$, consisting of functions whose traces are zero on the boundary is denoted by $H^s_0(\Omega)$:

$$H^s_0(\Omega) = \{v \in H^s(\Omega) : \gamma_0 v = 0, \text{ on } \partial\Omega\}.$$

2.6 Approximation Properties

In finite element methods, polynomials are typically used to approximate the solution of the PDE within each element. Their smoothness and ability to represent a wide range of functions make them a natural choice for approximating the solution of PDEs. In the following section, we examine approximation properties to ensure that the chosen polynomial function spaces provide sufficiently accurate approximation of the exact solution. We will later examine the function spaces of polynomials, which form the so-called shape functions of a finite element method. For a more detailed analysis of the approximation properties, see the book on finite elements by Larson and Bengzon [11] or the paper on optimal convergence rates of finite element methods by Babuška and Suri [12], which also provide proofs of the following theorems.

Theorem 2.2. *Let $\mathbb{P}_k$ be the space of polynomials of degree k in $\mathbb{R}^2$. Let E be a triangle or parallelogram in $\mathbb{R}^2$ and let h_E denote its diameter. Let $v \in H^s(E)$ for $s \geq 1$ and integer $k \geq 0$. Then there exists a constant C independent of v, h_E, and a function $\tilde{v} \in \mathbb{P}_k$ such that*

$$\forall 0 \leq q \leq s \quad ||v - \tilde{v}||_{H^q(E)} \leq C h_E^{\min(k+1,s)-q} |v|_{H^s(E)}.$$

This theorem states that a function v in a Sobolev space $H^s(E)$ over an element can be approximated by a polynomial of degree k, with an error that decreases with the size of the element h_E. The error also depends on the smoothness of v and the polynomial degree k.

From this theorem, we can construct a statement of convergence for the entire domain Ω. By subdividing Ω into N triangular or parallelogram elements E_i, for $i = 1, \ldots, N$, we construct a global approximation $\tilde{v}$ by combining the local approximations $\tilde{v}_{E_i}$on each element. Each local approximation $\tilde{v}_{E_i}$ satisfies the approximation property of Theorem 2.2 within its element.

The next theorem states that there exists an approximation that conserves the average of the normal flux on each edge e of an element E. This conservation property is crucial in FEM, as it ensures that physical conservation laws are satisfied and that the coupling between neighboring elements remains consistent.

Theorem 2.3. *Let E be a triangle or parallelogram in $\mathbb{R}^2$ and h_E its diameter. Let e be any edge of E and $\mathbf{n_e}$ the outward normal vector of edge e. Let $\mathbf{K}$ be a symmetric positive-definite matrix with constant entries and let $v \in H^s(E)$ for $s \geq 1$. Then there exists an approximation function $\tilde{v} \in \mathbb{P}_k$ satisfying*

$$\forall e \in \partial E \quad \int_e \mathbf{K}\nabla\tilde{v} \cdot \mathbf{n_e}\, ds = \int_e \mathbf{K}\nabla v \cdot \mathbf{n_e}\, ds$$

and the error estimate, where C is a constant independent of h_E:

$$\forall i = 0, 1, 2 \quad ||\nabla^i(v - \tilde{v})||_{L^2(E)} \leq C h_E^{\min(k+1,s)-i} |v|_{H^s(E)}.$$

2.7 Green's Formula

Green's formula is an essential tool for deriving the weak formulation of a PDE. Let Ω be a bounded domain in $\mathbb{R}^n$ with boundary $\partial\Omega$ and outward normal vector $\mathbf{n}$. It is a

generalization of integration by parts and follows from the divergence theorem. It relates the divergence of a vector field $\mathbf{f}$ to the flux of $\mathbf{f}$ through the boundary $\partial\Omega$:

$$\int_\Omega \nabla \cdot \mathbf{f}\, dx = \int_{\partial\Omega} \mathbf{f} \cdot \mathbf{n}\, ds.$$

Writing it component-wise, we obtain:

$$\sum_{i=1}^{n} \int_\Omega \frac{\partial f_i}{\partial x_i}\, dx = \sum_{i=1}^{n} \int_{\partial\Omega} f_i n_i\, ds.$$

Substituting $f_i = f_i g$ and applying the product rule, we obtain the partial integration formula:

$$\sum_{i=1}^{n} \int_\Omega \frac{\partial f_i}{\partial x_i} g\, dx = \sum_{i=1}^{n} \left(- \int_\Omega f_i \frac{\partial g}{\partial x_i}\, dx + \int_{\partial\Omega} f_i g n_i\, ds \right).$$

If we again substitute $f_i = w_i$ and $g = v$ and sum up the terms, we derived Green's formula:

$$\int_\Omega \nabla \cdot \mathbf{w} v\, dx = - \int_\Omega \mathbf{w} \cdot \nabla v\, dx + \int_{\partial\Omega} \mathbf{w} \cdot \mathbf{n} v\, ds.$$

With these mathematical tools introduced, we can now derive the DG methods for solving the Poisson equation.

3 Discontinuous Galerkin Methods

3.1 The DG Formulation of the Poisson Equation

3.1.1 Model Problem

Let Ω be a polygonal domain in $\mathbb{R}^2$ with boundary $\partial\Omega$. The boundary $\partial\Omega$ is divided into two disjoint sets Γ_D and Γ_N. For a given right-hand side function $f \in L^2(\Omega)$ and boundary value functions $g_D \in H^{1/2}(\Gamma_D)$ and $g_N \in L^2(\Gamma_N)$, we consider the Poisson equation:

$$-\Delta p = f \quad \text{in } \Omega, \tag{3.1}$$

$$p = g_D \quad \text{on } \Gamma_D, \tag{3.2}$$

$$\nabla p \cdot \mathbf{n} = g_N \quad \text{on } \Gamma_N. \tag{3.3}$$

Equation (3.2) represents a Dirichlet boundary condition, while equation (3.3) is a Neumann boundary condition. These conditions describe the behavior of the solution p and its normal derivative on the boundary $\partial\Omega$. The Poisson equation has a strong solution $p \in \mathcal{C}^2(\overline{\Omega})$, if f, g_D, and g_N are sufficiently smooth.

From the Poisson problem, we now derive the weak formulation using distributional derivatives.

3.1.2 Weak Formulation

For simplicity, we assume that the problem has purely Dirichlet boundary conditions, i.e., $\partial\Omega = \Gamma_D$. Using the trace theorem, we extend the boundary value function $g_D \in H^{1/2}(\partial\Omega)$ to the entire domain Ω. Let $p_D \in H^1(\Omega)$ be an extension of g_D such that $p_D = g_D$ on $\partial\Omega$. To better handle the boundary conditions, we decompose p as $p = p_D + w$, where $w \in H_0^1(\Omega)$ and $H_0^1(\Omega)$ is the subspace of $H^1(\Omega)$ with zero boundary values.

The goal is to find p, or equivalently w, such that it satisfies the weak formulation. To obtain the weak formulation (or variational formulation) of the Poisson equation, we multiply equation (3.1) by a test function $v \in H_0^1(\Omega)$ and integrate over the domain Ω:

$$\forall v \in H_0^1(\Omega) \quad -\int_\Omega \Delta w v \, dx = \int_\Omega f v \, dx.$$

Using the identity $\Delta p = \nabla \cdot \nabla p$, we apply Green's formula to the left-hand side integral and obtain:

$$\forall v \in H_0^1(\Omega) \quad \int_\Omega \nabla w \cdot \nabla v \, dx - \int_{\partial\Omega} \nabla w \cdot \mathbf{n} v \, ds = \int_\Omega f v \, dx.$$

By definition $v = 0$ on $\partial\Omega$, the boundary integral vanishes. This gives the weak formulation of the Poisson equation:

$$\forall v \in H_0^1(\Omega) \quad \int_\Omega \nabla w \cdot \nabla v \, dx = \int_\Omega f v \, dx. \tag{3.4}$$

The function p is then called the weak solution of the Poisson equation. The weak form in equation (3.4) can be written in an abstract form using a bilinear form $a(\cdot,\cdot)$ and a linear form $L(\cdot)$:

$$a(w, v) = L(v) \quad \forall v \in H_0^1(\Omega), \tag{3.5}$$

where

$$a(w, v) = \int_\Omega \nabla w \cdot \nabla v \, dx, \quad L(v) = \int_\Omega f v \, dx.$$

The bilinear form a is linear due to the linearity of the integral, the gradient operator, and the dot product.

The existence and uniqueness of w follow from the Lax-Milgram theorem.

3.1.3 Lax-Milgram Theorem

For the Lax-Milgram theorem to hold we need some properties for the bilinear and linear forms to be satisfied. First, we need continuity for both forms, ensuring that the forms are bounded.

Definition 3.1. *Let V be a Hilbert space. A bilinear form $a : V \times V \to \mathbb{R}$ and linear form $L : V \to \mathbb{R}$ are **continuous** if there exists a constant C_1 and C_2 such that*

$$\begin{aligned} \forall w, v \in V \quad & |a(w, v)| \leq C_1 ||w||_V ||v||_V \quad \text{and} \\ \forall v \in V \quad & |L(v)| \leq C_2 ||v||_V. \end{aligned}$$

The bilinear form $a(w, v) = \int_\Omega \nabla w \cdot \nabla v$, for $v, w \in H_0^1(\Omega)$, can be shown to be continuous by the Cauchy-Schwarz inequality:

$$\begin{aligned} |a(w, v)|^2 = \left| \int_\Omega \nabla w \cdot \nabla v \, dx \right|^2 &\leq \int_\Omega |\nabla w|^2 \, dx \int_\Omega |\nabla v|^2 \, dx \\ &= ||w||^2_{L^2(\Omega)} ||v||^2_{L^2(\Omega)} = ||w||^2_{H^1(\Omega)} ||v||^2_{H^1(\Omega)}. \end{aligned}$$

Additionally the bilinear form needs to be coercive. This gives a lower bound for the bilinear form dependent on $||v||_V$. This ensures that the bilinear form is nonzero unless v is zero.

Definition 3.2. *A bilinear form $a : V \times V \to \mathbb{R}$ is **coercive** if there exists a constant $C > 0$ such that*

$$\forall v \in V \quad a(v, v) \geq C ||v||^2_V.$$

For the bilinear form of the Poisson equation, coercivity follows straight from the definition of the Sobolev H^1 norm:

$$a(v, v) = \int_\Omega \nabla v \cdot \nabla v \, dx = ||\nabla v||^2_{L^2(\Omega)} = ||v||^2_{H^1(\Omega)}.$$

Theorem 3.1. *Let V be a Hilbert space and $a : V \times V \to \mathbb{R}$ a continuous and coercive bilinear form and $L : V \to \mathbb{R}$ a continuous linear form. Then there exists a unique solution $u \in V$ to the abstract weak form*

$$a(u, v) = L(v) \quad \forall v \in V.$$

The proof of the Lax-Milgram theorem can be found in the original paper by Lax and Milgram [13] or in the book by Larson and Bengzon [11].

3.1.4 Jump and Average Operators

To get from the classical variational formulation to the discontinuous Galerkin variational formulation, jump and average operators are introduced. They provide a tool to handle the discontinuous behavior of the functions along their element boundaries.
As defined before, an interior edge or face e is defined as the intersection of two distinct elements: $e = \partial E_i \cap \partial E_j \quad \text{where } i \neq j$. An exterior edge or face has only one neighboring element and is the intersection of an element boundary with the domain boundary: $e = \partial E_i \cap \partial\Omega$. For simplicity, we consider an interior edge e located between two elements E_1 and E_2 and its unit normal vector $\mathbf{n}_e$ points from E_1 to E_2. The unit normal vector of an exterior face is always outwards pointing.
The trace of a function $v \in H^1(\mathcal{T}_h)$ is well-defined on any edge e and is denoted by $v|_{E_i^e}$. For an interior edge e there exist two traces: $v|_{E_1^e}$ and $v|_{E_2^e}$.
By adding the values of the traces we can define the **average** operator, denoted by:

$$\forall e \in \partial E_1 \cap \partial E_2 \qquad \{v\} = \frac{1}{2}(v|_{E_1^e} + v|_{E_2^e}).$$

The **jump** operator is the difference between the two traces and gives a measure of discontinuity on one edge:

$$\forall e \in \partial E_1 \cap \partial E_2 \qquad [v] = v|_{E_1^e} - v|_{E_2^e}.$$

The extension of these operators to boundary faces is straightforward:

$$\forall e \in \partial E_1 \cap \partial\Omega \qquad \{v\} = v|_{E_1^e} \quad \text{and} \quad [v] = v|_{E_1^e}.$$

With these notations we can now derive the DG formulation of the Poisson equation.

3.1.5 Discontinuous Galerkin Formulation

In the following, Γ_h denotes the set of all interior faces, and we assume $s > 3/2$. For the DG formulation, we introduce the so-called penalty term $J^{\beta,\sigma} : H^s(\mathcal{T}_h) \times H^s(\mathcal{T}_h) \to \mathbb{R}$, a bilinear form that measures discontinuities along element faces over the domain:

$$J^{\beta,\sigma}(v, w) = \frac{\sigma}{|e|^\beta} \sum_{e \in \Gamma_h \cup \Gamma_D} \int_e [v][w]\, ds.$$

The penalty parameter σ is a positive constant that determines the strength of the penalty term. With $|e|$ the length of edge e is denoted and β is the superpenalization parameter dependent on the dimension of the problem, which in our case is always 2. We will state later that superpenalisation is needed in some cases to reach optimal convergence rates.

The DG bilinear form $a_\epsilon(v, w) : H^s(\mathcal{T}_h) \times H^s(\mathcal{T}_h) \to \mathbb{R}$ is then defined as:

$$a_\epsilon(v, w) = \sum_{E \in \mathcal{T}_h} \int_E \nabla v \cdot \nabla w\, dx \tag{3.6}$$

$$- \sum_{e \in \Gamma_h \cup \Gamma_D} \int_e \{\nabla v \cdot \mathbf{n}_e\}[w]\, ds \tag{3.7}$$

$$+ \epsilon \sum_{e \in \Gamma_h \cup \Gamma_D} \int_e \{\nabla w \cdot \mathbf{n}_e\}[v]\, ds \tag{3.8}$$

$$+ J^{\beta,\sigma}(v, w). \tag{3.9}$$

The DG bilinear form is derived from the weak form of the Poisson equation (3.5), with additional terms accounting for discontinuities and stabilizing the method. The first term

(3.6) is the classical bilinear form but instead of taking the integral over the whole domain, we sum up the integrals over each element. The second term (3.7) is the application of Green's formula and the average and jump operators come into play to handle the discontinuities. The term (3.8) is introducing the so-called symmetry parameter ϵ to the method. It can take the values $-1, 0, 1$ and controls the symmetry of the method. For $\epsilon = -1$ the method is symmetric. The last term (3.9) is the penalty term, which penalizes the discontinuity of the function over the domain.

Additionally, we define the DG linear form $L_\epsilon(v) : H^s(\mathcal{T}_h) \to \mathbb{R}$:

$$L(v) = \sum_{E \in \mathcal{T}_h} \int_E fv \, dx + \sum_{e \in \Gamma_D} \int_e \left(\epsilon \nabla v \cdot \mathbf{n}_e + \frac{\sigma}{|e|^\beta} v \right) g_D \, ds + \sum_{e \in \Gamma_N} \int_e v g_N \, ds.$$

According to Rivière [8], by applying Schwarz's inequality and trace inequalities, the integrals in the DG bilinear and linear forms are well-defined and bounded for any $s > 3/2$. The DG formulation of the Poisson problem 3.1.1 is then defined as follows: Find $p \in H^s(\mathcal{T}_h)$, with $s > 3/2$, such that:

$$a_\epsilon(p, v) = L(v) \quad \forall v \in H^s(\mathcal{T}_h). \tag{3.10}$$

Interior Penalty Galerkin (IPG) Methods

The parameter ϵ of the DG bilinear form determines the symmetry of the method and influences its properties. If $\epsilon = -1$ the method is symmetric and therefore called the **Symmetric Interior Penalty Galerkin** (SIPG) method. For $\epsilon = 1$ the method is called the **Non Symmetric Interior Penalty Galerkin** (NIPG) method. A special case worth mentioning is the **NIPG0** method, which is the same as the NIPG method but with penalty parameter $\sigma = 0$. And with $\epsilon = 0$ the method is called the **Incomplete Interior Penalty Galerkin** (IIPG) method, because the term (3.8) vanishes. These are the most common IPG methods and will be analyzed in the numerical experiments of this thesis.

3.2 Implementation

In the following section, we will introduce the necessary aspects for implementing the DG method and obtaining a numerical solution. The implementation aspects of the DG method are based on the book by Rivière [8] and the lecture notes by Dolejší [14]. To solve a PDE on a given domain Ω using finite elements, the domain needs to be discretized into a set of non-overlapping elements, denoted by $\mathcal{T}_h$.

3.2.1 Mesh Triangulation

The elements of $\mathcal{T}_h$ can have any polygonal shape, but triangles or quadrilaterals in 2D and tetrahedra or hexahedra in 3D are the most common geometries. Triangular meshes allow for greater flexibility when creating meshes for complex geometries. However, since this thesis focuses on simple square or rectangular domains, we will use uniform quadrilateral meshes, where all elements are squares of the same size. This simplifies the implementation of the DG method, as it is straightforward to identify neighboring elements and their shared faces. The structured nature of the mesh also makes it easier to handle boundary conditions for domain decomposition methods, as will be shown in the next chapter.

A mesh $\mathcal{T}_h$ for the domain $\Omega = [x_1, y_1] \times [x_2, y_2]$ is created by dividing the domain into $N_x \times N_y$ square elements of size $h_x = (x_2 - x_1)/N_x$ and $h_y = (y_2 - y_1)/N_y$. The total

number of elements is denoted by $N_E = N_x \cdot N_y$.

The structure of such a mesh, as used in continuous finite elements, is shown in Figure 3.1. The nodes $\mathbf{x}_i$, where $i = 1, \ldots, N_x \cdot N_y$, represent the coordinates of the points in the mesh. Each node $\mathbf{x}_i$ is associated with a function value v_i.

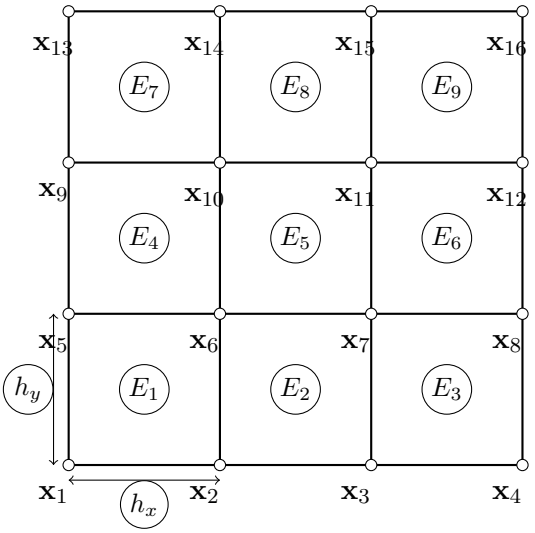


Figure 3.1: Example of a structured mesh used for continuous finite elements with $N_E = 9$ elements.

In discontinuous Galerkin methods, continuity across elements is not enforced. For example, in Figure 3.2, the node $\mathbf{x}_i$ shared by two elements, E_1 and E_2, has two different values: $v_i^{E_1}$ and $v_i^{E_2}$.

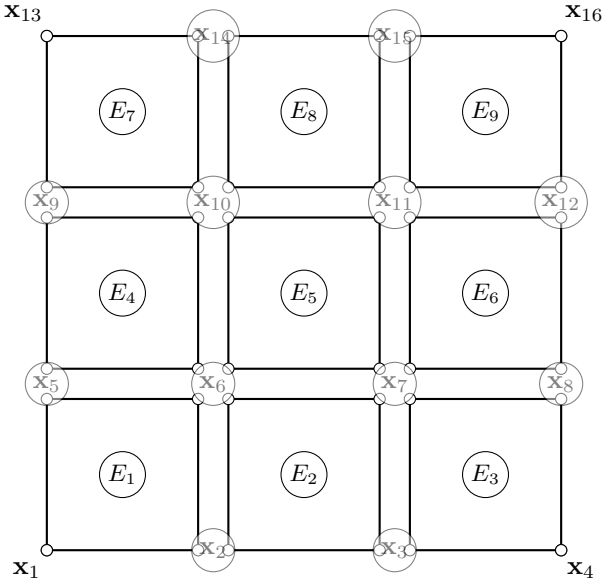


Figure 3.2: Example of a structured mesh used for discontinuous finite elements with $N_E = 9$ elements.

3.2.2 Discontinuous Finite Element Space

To approximate the solution of the Poisson equation, we first define the function space where the approximate solution p_h resides.
The finite element subspace $\mathcal{D}^k(\mathcal{T}_h)$ of piecewise discontinuous polynomials in the broken Sobolev space $H^s(\mathcal{T}_h)$, $s \geq 3/2$, is defined as:

$$\mathcal{D}^k(\mathcal{T}_h) = \{v \in L^2(\Omega) : v|_E \in \mathbb{P}_k(E), \quad \forall E \in \mathcal{T}_h\},$$

where $\mathbb{P}_k(E)$ denotes the space of polynomials of degree at most k restricted to each element E.
The mesh in Figure 3.2 displays a mesh for a discontinuous finite element space $\mathcal{D}^1(\mathcal{T}_h)$. This means that $v \in \mathcal{D}^1(\mathcal{T}_h)$ are piecewise linear functions on each element $E_i \in \mathcal{T}_h$, uniquely determined by their values $v_j^{E_i}$ at the nodes $\mathbf{x}_j$.
To uniquely determine a function in $\mathcal{D}^k(\mathcal{T}_h)$, we need $N_{\text{loc}} = (k+1)^2$ nodes for each element in 2D.

3.2.3 Basis Functions

The set of linearly independent functions $\{\phi_i^E\}_{i=1}^{N_{\text{loc}}}$ forms a basis for the finite element space on a single element E, $\mathcal{D}^k(E)$. The basis functions are defined over the whole domain, $\phi_i^E \in \mathcal{D}^k(\mathcal{T}_h)$, but unlike in continuous finite elements, they only have local support on a single element E. The union of all basis functions $\{\phi_i^{E_k}\}_{i=1}^{N_{\text{loc}}}$ for all elements $E_k \in \mathcal{T}_h$ forms a basis for the entire mesh.
In Section 3.2.7, Lagrangian basis functions will be introduced, and their shape will be illustrated.

3.2.4 Discretisation of the Weak Formulation Using Continuous FEM with Linear Basis Functions

To better understand the steps needed in formulating the DG scheme, we first examine how to solve the Poisson equation with the weak form (3.1.2) using linear continuous finite elements.
Contrary to before, the continuous finite element space $V^1(\mathcal{T}_h)$ enforces continuity over the closure of the domain $\bar{\Omega}$:

$$V^1(\mathcal{T}_h) = \{v, \quad v \in C(\bar{\Omega}) \cap H_0^1(\Omega), \quad v|_E \in \mathbb{P}^1(E) \quad \forall E \in \mathcal{T}_h\}.$$

The total number of nodes for continuous linear finite elements needed to uniquely determine a function $v \in V^1(\mathcal{T}_h)$ in one mesh is: $N_{\text{Nodes}} = (N_x + 1) \cdot (N_y + 1)$.
The functions that form the basis $\{\phi_i\}_{i=1}^{N_{\text{Nodes}}}$ of $V^1(\mathcal{T}_h)$, are given by $\phi_i \in V^1(\mathcal{T}_h)$ with the property $\phi_i(\mathbf{x}_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta.
We recall the structure of a mesh for continuous finite elements in Figure 3.1, where the support of ϕ_i consists of the elements that share the node $\mathbf{x}_i$.
With the basis functions, we can now reformulate the numerical solution p_h of the weak form (3.1.2) as a linear combination:

$$p_h(\mathbf{x}) = \sum_{j=1}^{N_{\text{Nodes}}} p_j \phi_j(\mathbf{x}),$$

where p_j is the value of the function at the node $\mathbf{x}_j$, and it obviously follows:

$$\nabla p_h(\mathbf{x}) = \sum_{j=1}^{N_{\text{Nodes}}} p_j \nabla \phi_j(\mathbf{x}).$$

Since $\{\phi_i\}_{i=1}^{N_{\text{Nodes}}}$ forms a basis of $V^1(\mathcal{T}_h)$, it is sufficient that the weak formulation (3.1.2) is satisfied for each basis function ϕ_i. Using the linear combination for p_h, we need to find $\{p_j\}_{j=1}^{N_{\text{Nodes}}} \in \mathbb{R}^{N_{\text{Nodes}}}$ such that:

$$\forall i = 1, \ldots, N_{\text{Nodes}} \quad \sum_{j=1}^{N_{\text{Nodes}}} p_j(\nabla\phi_j, \nabla\phi_i) = (f, \phi_i).$$

From this formulation, we can define the system matrix $\mathbf{A}$, which in this case is the so-called stiffness matrix $\mathbf{S}$:

$$\mathbf{A} = \{A_{ij}\}_{i,j=1}^{N_{\text{Nodes}}} = ((\nabla\phi_i, \nabla\phi_j))_{i,j=1}^{N_{\text{Nodes}}} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, d\mathbf{x}.$$

Because the basis functions are zero outside of their support, the integral over the domain reduces to the integral over their support:

$$A_{ij} = \begin{cases} \int_{\text{supp}(\phi_i)} \nabla\phi_i \cdot \nabla\phi_i \, d\mathbf{x} & \text{if } i = j, \\ \int_{\text{supp}(\phi_i) \cap \text{supp}(\phi_j)} \nabla\phi_i \cdot \nabla\phi_j \, d\mathbf{x} & \text{if } i \neq j \text{ and } \mathbf{x}_i, \mathbf{x}_j \text{ share an edge}, \\ 0 & \text{otherwise.} \end{cases}$$

The right-hand side vector $\mathbf{b}$ is:

$$\mathbf{b} = \{b_i\}_{i=1}^{N_{\text{Nodes}}} = ((f, \phi_i))_{i=1}^{N_{\text{Nodes}}} = \int_\Omega f\phi_i \, d\mathbf{x} = \int_{\text{supp}(\phi_i)} f\phi_i \, d\mathbf{x}.$$

Thus, we obtain the linear system $\mathbf{Ap} = \mathbf{b}$.

In the continuous case, the basis functions are defined over the entire domain, with their support being the elements that share the node $\mathbf{x}_i$. Solving the integrals over different elements is computationally expensive, which is why we exploit the fact that the basis functions within each element $E \in \mathcal{T}_h$ can be projected onto a reference element $\hat{E}$. Using a reference element standardizes the evaluation of the basis functions, their gradients, and the integrals over all elements without considering the actual position of each element.

3.2.5 Discretisation of the DG Formulation Using Discontinuous Basis Functions

With the same approach, we can perform the discretization for the DG formulation (3.10) using discontinuous basis functions in

$$\mathcal{D}^k(\mathcal{T}_h) = \{v \in L^2(\Omega) : v|_E \in \mathbb{P}_k(E), \quad \forall E \in \mathcal{T}_h\}.$$

We now define the set of linearly independent basis functions for each element E_l, with $l = 1, \ldots, N_E$:

$$B_l := \{\phi_j^{E_l} \in \mathcal{D}^k(\mathcal{T}_h) : \phi_j^{E_l}(\mathbf{x}_i) = \delta_{ij}, \quad \text{supp}(\phi_j^{E_l}) \subseteq E_l, \quad i, j = 1, \ldots, N_{\text{loc}}\}.$$

The union of all local bases $B = \bigcup_{l=1}^{N_E} B_l$ forms a basis for the whole space:

$$\mathcal{D}^k(\mathcal{T}_h) = \text{span}\,(B)\,.$$

With the help of the basis functions in B, we can now approximate the numerical solution p_h of the DG formulation (3.10) as a linear combination of the basis functions over all elements:

$$p_h(\mathbf{x}) = \sum_{l=1}^{N_E} \sum_{i=1}^{N_{\text{loc}}} p_i^{E_l} \phi_i^{E_l}(\mathbf{x}), \quad \text{and } \nabla p_h(\mathbf{x}) = \sum_{l=1}^{N_E} \sum_{i=1}^{N_{\text{loc}}} p_i^{E_l} \nabla\phi_i^{E_l}(\mathbf{x}),$$

where $p_i^{E_l}$ are the values of the local nodes of element E_l.

The DG formulation contains integrals over the element faces. We need to define the linear combination of basis functions for the function restricted to the face $p_h|_e$. As introduced in the Chapter 2, there are two traces for a function $v \in \mathcal{D}^k(\mathcal{T}_h)$ on the face e, one being $v|_{E_1^e}$ and the other $v|_{E_2^e}$. To better handle the different traces, we introduce a global-to-local mapping of the nodes of the mesh. In the global numbering, even if two or more elements share a node, the node is counted only once. In Figure 3.2, this numbering is displayed.

Let I_e be the global index set of all nodes contained in the face e, shared by two elements E_1 and E_2, and $N_{\text{loc}}^e = |I_e| = k+1$ the number of nodes on the face e. It is defined as:

$$I_e = \{i \in \{1, \dots, kN_xN_y\} : \mathbf{x}_i \text{ lies on } e\}.$$

To relate the global indices to the local indices of each element, we define the mappings for the neighboring elements E_1 and E_2:

$$gl_1 : I_e \to I_{E_1}^e, \quad \text{and} \quad gl_2 : I_e \to I_{E_2}^e.$$

With these mappings, we can now access the local basis functions of each element sharing a face e:

$$I_e^{E_1} = \{gl_1(i) : i \in I_e\} \quad \text{and} \quad I_e^{E_2} = \{gl_2(i) : i \in I_e\}.$$

Using these two sets, we define the jump and average operators along the face e:

$$[p_h(\mathbf{x})|_e] = \sum_{i \in \mathcal{I}_e} [p_i\phi_i(x)|_e] = \sum_{i \in \mathcal{I}_e} \left(p_{\text{gl}_1(i)}^{E_1}\phi_{\text{gl}_1(i)}^{E_1}(\mathbf{x})|_e - p_{\text{gl}_2(i)}^{E_2}\phi_{\text{gl}_2(i)}^{E_2}(\mathbf{x})|_e\right),$$
$$\{p_h(\mathbf{x})|_e\} = \sum_{i \in \mathcal{I}_e} \{p_i\phi_i(x)|_e\} = \frac{1}{2}\sum_{i \in \mathcal{I}_e} \left(p_{\text{gl}_1(i)}^{E_1}\phi_{\text{gl}_1(i)}^{E_1}(\mathbf{x})|_e + p_{\text{gl}_2(i)}^{E_2}\phi_{\text{gl}_2(i)}^{E_2}(\mathbf{x})|_e\right).$$

Similarly, we define the jump and average operators for the gradient of the function p_h along the face e:

$$[\nabla p_h(\mathbf{x})|_e] = \sum_{i \in \mathcal{I}_e} [\nabla p_i\phi_i(x)|_e] = \sum_{i \in \mathcal{I}_e} \left(p_{\text{gl}_1(i)}^{E_1}\nabla\phi_{\text{gl}_1(i)}^{E_1}(\mathbf{x})|_e - p_{\text{gl}_2(i)}^{E_2}\nabla\phi_{\text{gl}_2(i)}^{E_2}(\mathbf{x})|_e\right),$$
$$\{\nabla p_h(\mathbf{x})|_e\} = \sum_{i \in \mathcal{I}_e} \{\nabla p_i\phi_i(x)|_e\} = \frac{1}{2}\sum_{i \in \mathcal{I}_e} \left(p_{\text{gl}_1(i)}^{E_1}\nabla\phi_{\text{gl}_1(i)}^{E_1}(\mathbf{x})|_e + p_{\text{gl}_2(i)}^{E_2}\nabla\phi_{\text{gl}_2(i)}^{E_2}(\mathbf{x})|_e\right).$$

As in the previous section, it is sufficient that the DG formulation (3.10) is satisfied for each basis function ϕ_i:

$$a_\epsilon(p_h, \phi_i) = l(\phi_i), \quad \forall \phi_i \in B.$$

Inserting the linear combinations of the basis functions into the bilinear and linear forms results in a linear system of equations for the unknowns $p \in \mathbb{R}^{N_E \cdot N_{\text{loc}}}$:

$$\sum_{j=1}^{N_{\text{loc}}}\sum_{l=1}^{N_E} a_\epsilon(\phi_j^{E_l}, \phi_i^{E_k})p_j^{E_l} = l(\phi_i^{E_k}), \quad \forall i = 1, \dots, N_{\text{loc}}, \forall k = 1, \dots, N_E.$$

In Section 3.2.9, we will derive how the linear system is finally assembled after introducing the concept of the reference element.

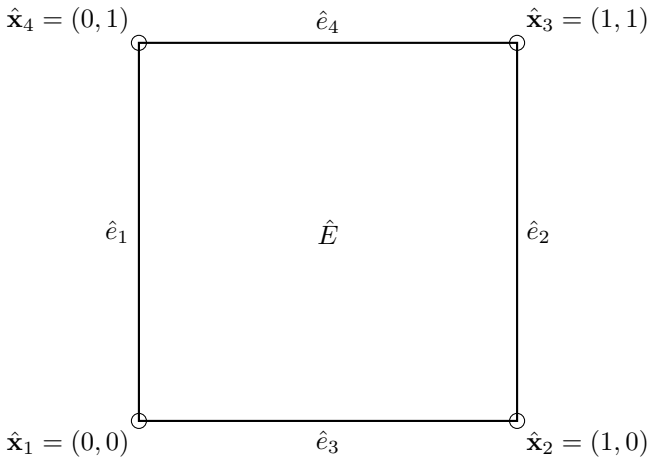


Figure 3.3: The unit square reference element $\hat{E}$.

3.2.6 Reference Element

The reference square element $\hat{E}$, displayed in Figure 3.3, is defined as:

$$\hat{E} = \{(\hat{x}_1, \hat{x}_2) \,|\, 0 \leq \hat{x}_i \leq 1,\, i = 1, 2\}.$$

An element $E \in \mathcal{T}_h$ is referred to as a physical element. We assume that each physical element is associated with a mapping $F_E(\hat{\mathbf{x}}) : \hat{E} \to \mathbb{R}^2$, which transforms the reference element $\hat{E}$ to the physical element E.
This mapping is simple for square elements, as the reference element just needs to be scaled and translated onto the physical element. This simplicity is due to the alignment of the coordinate axes of the reference element with the physical element.
For general quadrilateral elements, the mapping is given by the function:

$$F_E(\hat{\mathbf{x}}) = F_E(\hat{x}_1, \hat{x}_2) = \mathbf{x}_1 + \hat{x}_1 (\mathbf{x}_2 - \mathbf{x}_1) + \hat{x}_2 (\mathbf{x}_3 - \mathbf{x}_1) + \hat{x}_1 \hat{x}_2 (\mathbf{x}_4 - \mathbf{x}_2 - \mathbf{x}_3 + \mathbf{x}_1),$$

where $\mathbf{x}_i$ are the coordinates of the nodes of the physical element.

The mapping is assumed to be invertible and continuously differentiable. The existence of an inverse mapping F_E^{-1} follows from the assumption that the Jacobian matrix

$$J_{F_E} := \frac{D}{D\hat{\mathbf{x}}} F_E = \begin{pmatrix} \frac{\partial F_E}{\partial \hat{x}_1} & \frac{\partial F_E}{\partial \hat{x}_2} \end{pmatrix}$$

does not change sign on $\hat{E}$, a condition guaranteed by the inverse function theorem.

The mapping can also be extended to more complex elements.

The Jacobian matrix for the square elements in the mesh used in this thesis is given as follows: For quadrilateral elements with edge length h, we differentiate the mapping F_E, obtaining:

$$\frac{D}{D\hat{x}_1} = \begin{pmatrix} h \\ 0 \end{pmatrix} \quad \text{and} \quad \frac{D}{D\hat{x}_2} = \begin{pmatrix} 0 \\ h \end{pmatrix},$$

and thus the Jacobian matrix is:

$$J_{F_E} = h \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The Jacobian appears when transforming integrals over physical elements to the reference element.

3.2.6.1 Integral Evaluation on the Reference Element

The Fubini theorem provides the foundation for transforming an integral over a physical element into an equivalent integral over the reference element.

Theorem 3.2. *Fubini Theorem Let $E \in \mathcal{T}_h$ be a physical element and $F_E : \hat{E} \to E$ be the mapping from the reference element to the physical element. Then for any integrable function $f : E \to \mathbb{R}$, the following holds:*

$$\int_E f(\mathbf{x})\, d\mathbf{x} = \int_{\hat{E}} f(F_E(\hat{\mathbf{x}}))|\det J_{F_E}(\hat{\mathbf{x}})|\, d\hat{\mathbf{x}}.$$

If F_E is a linear mapping, the Jacobian is constant, and the determinant is the area of the physical element: $|\det J_{F_E}(\hat{\mathbf{x}})| = |E|$.
Using this theorem, we now derive the transformation rules for integrals involving derivatives over a physical element, so that we can apply them in the DG formulation.

Let $E \in \mathcal{T}_h$ be a physical element and $F_E : \hat{E} \to E$ the mapping from the reference element to the physical element. For the inverse mapping, we have $\hat{\mathbf{x}} = F_E^{-1}(\mathbf{x}) = (F_{E,1}^{-1}, \ldots, F_{E,d}^{-1})$, where d is the dimension of the elements, which is 2 for the square elements used in this thesis.
We define the function $\hat{f} : \hat{E} \to \mathbb{R}$ as $\hat{f}(\hat{\mathbf{x}}) = f(F_E(\hat{\mathbf{x}})) = f \circ F_E(\hat{\mathbf{x}})$.

Applying Fubini's theorem and the chain rule, we obtain:

$$\begin{aligned}
\int_E \frac{\partial}{\partial x_i} f(\mathbf{x}) &= \int_{\hat{E}} \frac{\partial}{\partial x_i} \hat{f}(F_E(\mathbf{x}))|\det J_{F_E}(\hat{\mathbf{x}})| \\
&= \int_{\hat{E}} \sum_{j=1}^{d} \frac{\partial \hat{f}(\hat{\mathbf{x}})}{\partial \hat{x}_j} \frac{\partial \hat{x}_j}{\partial x_i} |\det J_{F_E}(\hat{\mathbf{x}})|\, d\hat{\mathbf{x}} \\
&= \int_{\hat{E}} \sum_{j=1}^{d} \frac{\partial \hat{f}(\hat{\mathbf{x}})}{\partial \hat{x}_j} \frac{\partial F_{E,j}^{-1}}{\partial x_i} |\det J_{F_E}(\hat{\mathbf{x}})|\, d\hat{\mathbf{x}}.
\end{aligned}$$

To solve the integral, we need the partial derivatives of the inverse mapping F_E^{-1}. We derive an identity showing that the Jacobian of the inverse mapping is the inverse of the Jacobian of the mapping:

$$J_{F_E^{-1}} = J_{F_E}^{-1}.$$

Let $i, k = 1, \ldots, d$, then:

$$\begin{aligned}
x_k &= F_{E,k}(F_E^{-1}(\mathbf{x})) \qquad |\frac{\partial}{\partial x_i} \\
\frac{\partial x_k}{\partial x_i} &= \sum_{j=1}^{d} \frac{\partial F_{E,k}}{\partial \hat{x}_j} \frac{\partial F_{E,j}^{-1}}{\partial x_i} \\
\delta_{ik} &= \sum_{j=1}^{d} \frac{\partial F_{E,k}}{\partial \hat{x}_j} \frac{\partial F_{E,j}^{-1}}{\partial x_i} \\
\delta_{ik} &= \sum_{j=1}^{d} (J_{F_E})_{kj} \left(J_{F_E^{-1}}\right)_{ji}.
\end{aligned}$$

From the last equation, we derived the elementwise identity for the inverse Jacobians:

$$\mathbb{I} = J_{F_E^{-1}} \left(J_{F_E}\right)^{-1} \qquad \Rightarrow \qquad J_{F_E^{-1}} = \left(J_{F_E}\right)^{-1}_{ij}.$$

Inverting the Jacobian J_{F_E} is significantly simpler than finding the inverse mapping for each element, as the Jacobian is just a 2×2 or 3×3 matrix. Using this identity, the integral becomes:

$$\begin{aligned}
\int_E \frac{\partial}{\partial x_i} f(\mathbf{x}) &= \int_{\hat{E}} \sum_{j=1}^{d} \frac{\partial \hat{f}(\hat{\mathbf{x}})}{\partial \hat{x}_j} \left(J_{F_E}(\hat{\mathbf{x}})^{-1}\right)_{ji} |\det J_{F_E}(\hat{\mathbf{x}})| \, d\hat{\mathbf{x}} \\
&= \int_{\hat{E}} \sum_{j=1}^{d} \frac{\partial \hat{f}(\hat{\mathbf{x}})}{\partial \hat{x}_j} \left(J_{F_E}(\hat{\mathbf{x}})^{-T}\right)_{ij} |\det J_{F_E}(\hat{\mathbf{x}})| \, d\hat{\mathbf{x}} \\
&= \int_{\hat{E}} \left(J_{F_E}(\hat{\mathbf{x}})^{-T} \hat{\nabla} \hat{\mathbf{f}}(\hat{\mathbf{x}})\right)_i |\det J_{F_E}(\hat{\mathbf{x}})| \, d\hat{\mathbf{x}}.
\end{aligned}$$

To utilize these calculations for the stiffness matrix, we evaluate integrals involving the product of derivatives, which corresponds to the stiffness matrix integrals:

$$\int_E \frac{\partial \phi_a(\mathbf{x})}{\partial x_i} \frac{\partial \phi_b(\mathbf{x})}{\partial x_j} \, d\mathbf{x} = \int_{\hat{E}} \left(J_{F_E}(\hat{\mathbf{x}})^{-T} \hat{\nabla} \hat{\phi}_a(\hat{\mathbf{x}})\right)_i \left(J_{F_E}(\hat{\mathbf{x}})^{-T} \hat{\nabla} \hat{\phi}_b(\hat{\mathbf{x}})\right)_j |\det J_{F_E}(\hat{\mathbf{x}})| \, d\hat{\mathbf{x}}.$$

Recalling that the Jacobian for square elements is $J_{F_E} = h \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, which leads to the simplified integral:

$$\int_E \frac{\partial \phi_a(\mathbf{x})}{\partial x_i} \frac{\partial \phi_b(\mathbf{x})}{\partial x_j} \, d\mathbf{x} = \int_{\hat{E}} \left(\frac{1}{h} \hat{\nabla} \hat{\phi}_a(\hat{\mathbf{x}})\right)_i \left(\frac{1}{h} \hat{\nabla} \hat{\phi}_b(\hat{\mathbf{x}})\right)_j h^2 \, d\hat{\mathbf{x}}. \tag{3.11}$$

3.2.6.2 Integral Evaluation on the Face of the Reference Element

In addition to the integrals over the elements $E \in \mathcal{T}_h$, we need to evaluate face integrals in the DG formulation (3.10) along the physical faces $e \in \Gamma_h \cup \Gamma_D \cup \Gamma_N$, of the form:

$$\int_e f(\mathbf{x}) \, ds, \tag{3.12}$$

$$\int_e \mathbf{f}(\mathbf{x}) \cdot \mathbf{n}_e \, ds, \tag{3.13}$$

where $\mathbf{n}_e$ is the normal vector of the face e and the given functions $f(\mathbf{x}) : e \to \mathbb{R}$ and $\mathbf{f}(\mathbf{x}) : e \to \mathbb{R}^2$ are defined on the face e.
To utilize the concept of reference elements, we find a parametrization that maps the unit interval to the reference face $\hat{\mathbf{x}}_{\hat{e}_i}(t) : [0,1] \to \hat{e}_i$, where $\hat{e}_i$ denotes the i-th face of the reference element (see fig. 3.3).
For all elements E the mapping $E = F_E(\hat{E})$ exists, and for reference faces $\hat{e}_i$ the mapping is $e_i = F_E(\hat{e}_i)$.
The parametrizations $\hat{\mathbf{x}}_{\hat{e}_i}(t)$, their tangential vectors $d\hat{\mathbf{x}}_{\hat{e}_i} := \frac{d}{dt}\hat{\mathbf{x}}_{\hat{e}_i}(t)$, and their outward-

pointing normal unit vectors $\mathbf{n}_{\hat{e}_i}$ for square elements are:

$$\hat{\mathbf{x}}_{\hat{e}_1}(t) = \begin{pmatrix} 0 \\ t \end{pmatrix}, \qquad d\hat{\mathbf{x}}_{\hat{e}_1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad \mathbf{n}_{\hat{e}_1} = \begin{pmatrix} -1 \\ 0 \end{pmatrix},$$
$$\hat{\mathbf{x}}_{\hat{e}_2}(t) = \begin{pmatrix} 1 \\ t \end{pmatrix}, \qquad d\hat{\mathbf{x}}_{\hat{e}_2} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad \mathbf{n}_{\hat{e}_2} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$
$$\hat{\mathbf{x}}_{\hat{e}_3}(t) = \begin{pmatrix} t \\ 0 \end{pmatrix}, \qquad d\hat{\mathbf{x}}_{\hat{e}_3} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \mathbf{n}_{\hat{e}_3} = \begin{pmatrix} 0 \\ -1 \end{pmatrix},$$
$$\hat{\mathbf{x}}_{\hat{e}_4}(t) = \begin{pmatrix} t \\ 1 \end{pmatrix}, \qquad d\hat{\mathbf{x}}_{\hat{e}_4} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \mathbf{n}_{\hat{e}_4} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

where $t \in [0, 1]$.
To transform the domain of the integral using Fubini's Theorem, we need to calculate the Jacobian of the mapping from the reference face to the physical face. The map $\hat{\mathbf{x}}_{\hat{e}_i}(t)$ lifts the reference face, respectively the unit interval $[0, 1]$, to the face of the reference element $\hat{e}_i$ in two-dimensional space. We then map the reference face to the physical face e_i using $F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t))$.
To evaluate the integral over the physical face, we utilize the definition of scalar line integrals in the plane from a vector calculus book [15]:

$$\int_{e_i} f(\mathbf{x})\, ds = \int_0^1 f(F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t))) \left\| \frac{d}{dt} F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t)) \right\| dt,$$

where $\|\cdot\|$ denotes the Euclidean norm of the vector.

Using the chain rule, the derivative of the mapping function from the reference face to the physical face is:

$$\frac{d}{dt} F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t)) = \frac{D}{D\hat{\mathbf{x}}} F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t)) \cdot \frac{d}{dt} \hat{\mathbf{x}}_{\hat{e}_i}(t) = J_{F_E} \cdot d\hat{\mathbf{x}}_{\hat{e}_i}.$$

The integral (3.12) becomes:

$$\int_{e_i} f(\mathbf{x})\, ds = \int_0^1 f\left(F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t))\right) \left\| J_{F_E} \cdot d\hat{\mathbf{x}}_{\hat{e}_i} \right\| dt. \tag{3.14}$$

Since the mesh aligns with the coordinate axis and it consists of square elements, with edge lenght h, the norm of the product of the Jacobean and the tangential vector is constant for any face

$$\|J_{F_E} \cdot d\hat{\mathbf{x}}_{\hat{e}_i}\| = \left\| \begin{pmatrix} h & 0 \\ 0 & h \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\| = \left\| \begin{pmatrix} h & 0 \\ 0 & h \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\| = h \quad \text{for } i = 1, \ldots, 4.$$

Thus, the integral (3.14) simplifies to:

$$\int_{e_i} f(\mathbf{x})\, ds = h \int_0^1 f\left(F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t))\right) dt. \tag{3.15}$$

Similarly, the integral (3.13) is calculated using the normal vector of the physical face. Since the face is a straight line, the normal vector $\mathbf{n}_e$ of the physical face is constant and coincides with the normal vector of the reference face $\mathbf{n}_{\hat{e}_i}$. Therefore, the integral simplifies to:

$$\int_{e_i} \mathbf{f}(\mathbf{x}) \cdot \mathbf{n}_{e_i}\, ds = h \int_0^1 \mathbf{f}\left(F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t))\right) \cdot \mathbf{n}_{\hat{e}_i}\, dt. \tag{3.16}$$

This result allows for efficient computation of face integrals in the DG formulation.

3.2.7 Lagrangian Basis Functions

As introduced in Section 3.2.3, the basis functions can be any polynomial basis that spans the finite element space. Lagrange polynomials are a common choice for basis functions because they are easy to implement for any polynomial degree, and the distribution of nodes within the elements is straightforward, which is why they are used in this thesis. More advanced FEM libraries utilize hierarchical basis functions to improve performance when the polynomial degree varies across elements, or orthogonal polynomials to improve numerical stability.

We will now examine the Lagrangian basis functions on the reference element, as we have demonstrated how to evaluate any integral over a physical element by transforming it to the reference element.
For the Lagrangian basis functions of degree k, we need $N_{\mathrm{DoF}} = (k+1)^d$ nodes per element, which are uniformly distributed in each direction. The distribution of the nodes in lexicographical order $\hat{\mathbf{x}}_i$ for $i = 1, \ldots, N_{\mathrm{DoF}}$ is displayed in figure 3.4 and can be calculated by:

$$\hat{x}^{(k)}_{j(k+1)+i+1,1} = \frac{i}{k}, \quad \hat{x}^{(k)}_{j(k+1)+i+1,2} = \frac{j}{k}, \quad i,j = 0, \ldots, k. \tag{3.17}$$

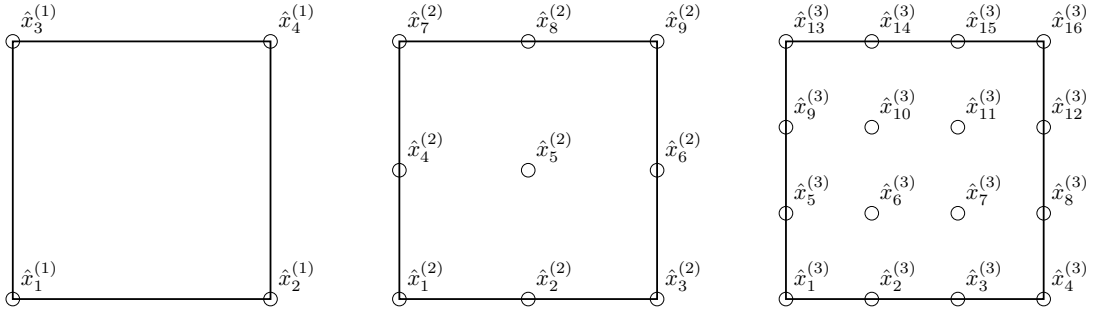


Figure 3.4: Distribution of Lagrange nodes for different polynomial degrees k.

1D Lagrange Polynomials

To construct the later-used Lagrangian basis functions, we start with 1D Lagrange polynomials and extend them to 2D. Let $\{x_i\}_{i=0}^k$ be a set of $k+1$ distinct nodes, where k is the polynomial degree of the Lagrangian functions. The Lagrange polynomials $\ell_i(x)$ are defined as:

$$\ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{k} \frac{x - x_j}{x_i - x_j}.$$

The Lagrange polynomials $\{\ell_i(x)\}_{i=0}^k$ form a basis of the space of polynomials of degree at most k in the interval $[x_0, x_k]$ and satisfy the property $\ell_i(x_j) = \delta_{ij}$. By constructing Lagrange polynomials with equidistant nodes in the interval $[0, 1]$, we can use the tensor product to construct the 2D Lagrangian basis functions for the reference element.

2D Reference Lagrangian Basis Functions

The Lagrangian nodes in 2D are displayed in figure 3.4 and can be computed for higher orders using (3.17). For the node $\hat{\mathbf{x}}_{j(k+1)+i+1} = \left(\frac{i}{k}, \frac{j}{k}\right)$, we construct the 2D Lagrangian

basis function:

$$\hat{\phi}^{(k)}_{j(k+1)+i+1}(x) = \ell_i(x) \cdot \ell_j(x). \tag{3.18}$$

The Lagrange basis functions $\{\hat{\phi}^{(k)}_i\}_{i=1}^{N_{\mathrm{DoF}}}$ form a basis of the space of polynomials of degree at most k on the reference element $\hat{E}$. These basis functions also satisfy the Kronecker delta property $\hat{\phi}^{(k)}_i(\hat{\mathbf{x}}_j) = \delta_{ij}$ for $i, j = 1, \ldots, N_{\mathrm{DoF}}$.

Gradient of the Lagrangian Basis Functions

In the DG formulation, the gradient of the test and trial functions is often required. To approximate these gradients, we also need to compute the gradient of the basis functions on the reference element. Because of the structure of the basis functions given in (3.18), the gradient can be easily computed:

$$\nabla\hat{\phi}^{(k)}_{j(k+1)+i+1}(x) = \big(\ell_i'(x) \cdot \ell_j(x), \quad \ell_i(x) \cdot \ell_j'(x)\big)^T,$$

where the derivatives of the 1D Lagrange polynomials are:

$$\ell_i'(x) = \sum_{\substack{j=0 \\ j\neq i}}^{k} \frac{1}{x_i - x_j} \prod_{\substack{l=0 \\ l\neq i \\ l\neq j}}^{k} \frac{x - x_l}{x_i - x_l}.$$

3.2.8 Numerical Quadratures

The last ingredient for the evaluation of the integrals in the DG formulation is numerical integration. Solving the nonlinear integrals analytically is too complex or not possible; therefore, we approximate the integrals with numerical quadrature rules. There exist many quadrature rules with different properties, but their general idea is to approximate the integral of a function over a domain by a weighted sum of function values at certain points in the domain.

$$\int_\Omega f(\mathbf{x})\, d\mathbf{x} \approx \sum_{i=1}^{N_q} w_i f(\mathbf{x}_i),$$

where $\mathbf{x}_i$ are the quadrature points in the domain Ω and w_i are the quadrature weights. The accuracy of a quadrature rule depends on its order, which indicates the highest polynomial degree that the rule can integrate exactly. For example, the midpoint rule is of order 1, as it can integrate linear functions exactly. The Gauss quadrature rule can exactly approximate the integral of polynomials of degree $2N_q - 1$, where N_q is the number of quadrature points [16].
For this reason, we adopt the widely used Gauss quadrature rule for numerical integration over the reference element and its faces.

Face Quadratures

As shown in the previous section, with the help of parametrization, the integral over a face can be transformed to the unit interval. While the Gauss quadrature rule is typically defined on the interval $[-1, 1]$, we need to solve the integral in $[0, 1]$. Therefore, the quadrature points and weights are shown in Table 3.1. For higher orders, the values

can be looked up in [14] or [17]. The face integrals of (3.15) and (3.16) can then be approximated by:

$$h \int_0^1 f\left(F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t))\right) dt \approx h \sum_{i=1}^{N_q} w_i \hat{f}\left(x_i^e\right), \tag{3.19}$$

$$h \int_0^1 f\left(F_E(\hat{\mathbf{x}}_{\hat{e}_i}(t))\right) \cdot \mathbf{n}_{\hat{e}_i} dt \approx h \sum_{i=1}^{N_q} w_i \hat{f}\left(x_i^e\right) \cdot \mathbf{n}_{\hat{e}_i}. \tag{3.20}$$

Table 3.1: Gauss Quadrature Points and Weights for the Interval $[0, 1]$

# Points (n)	x_1	x_2	x_3	Weights
1	$\frac{1}{2}$	—	—	1
2	$\frac{1}{2} - \frac{1}{2\sqrt{3}}$	$\frac{1}{2} + \frac{1}{2\sqrt{3}}$	—	$\frac{1}{2}, \frac{1}{2}$
3	$\frac{1}{2} - \frac{\sqrt{3/5}}{2}$	$\frac{1}{2}$	$\frac{1}{2} + \frac{\sqrt{3/5}}{2}$	$\frac{5}{18}, \frac{4}{9}, \frac{5}{18}$

Element Quadratures

Approximating integrals in higher dimensions for complex geometries can be challenging, but for quadrilateral and especially square elements, the Gauss quadrature rule is straightforward. The concept of the Gauss quadrature rule is extended to 2D by taking the tensor product of the 1D rule. The placement of the quadrature points is shown in Figure 3.5. The integral (3.11) is then approximated by:

$$\int_{\hat{E}} \hat{\nabla}\hat{\phi}_a(\hat{\mathbf{x}}) \hat{\nabla}\hat{\phi}_b(\hat{\mathbf{x}})\, d\hat{\mathbf{x}} \approx \sum_{i=1}^{N_q} \sum_{j=1}^{N_q} w_i w_j \hat{\nabla}\hat{\phi}_a(\mathbf{x}_{ij}) \cdot \hat{\nabla}\hat{\phi}_b(\mathbf{x}_{ij}), \tag{3.21}$$

where w_i and w_j are the weights of the 1D Gauss quadrature rule and $\mathbf{x}_{ij}$ are the $N_q^{(2D)} = (N_q)^2$ quadrature points in the reference element, with $\mathbf{x}_{ij} = (x_i, x_j)$ for $i, j = 1, \ldots, N_q$. We can therefore exchange the double sum for a single sum over the quadrature points, which simplifies the implementation to:

$$\int_{\hat{E}} \hat{\nabla}\hat{\phi}_a(\hat{\mathbf{x}}) \hat{\nabla}\hat{\phi}_b(\hat{\mathbf{x}})\, d\hat{\mathbf{x}} \approx \sum_{k=1}^{N_q^{(2D)}} w_k \hat{\nabla}\hat{\phi}_a(\mathbf{x}_k) \cdot \hat{\nabla}\hat{\phi}_b(\mathbf{x}_k). \tag{3.22}$$

3.2.9 Local Matrices and Right-Hand Side

In this section, we will derive the local matrices created by the element and face integrals, which will be assembled into the global system matrix in the next section. Each integral in the DG formulation needs to be derived only once, as we utilize the concept of the reference element.

3.2.9.1 Volume Contribution

There are two element integrals in the DG formulation: the stiffness matrix on the left-hand side and the integral $\int_E fv\, dx$ on the right-hand side.
The local stiffness matrix for $i, j = 1, \ldots, k+1$, where k is the polynomial degree of the basis functions, is given by:

$$\forall E \in \mathcal{T}_h \qquad (\boldsymbol{A}_E)_{ij} = \int_E \nabla\phi_j \cdot \nabla\phi_i\, dx.$$

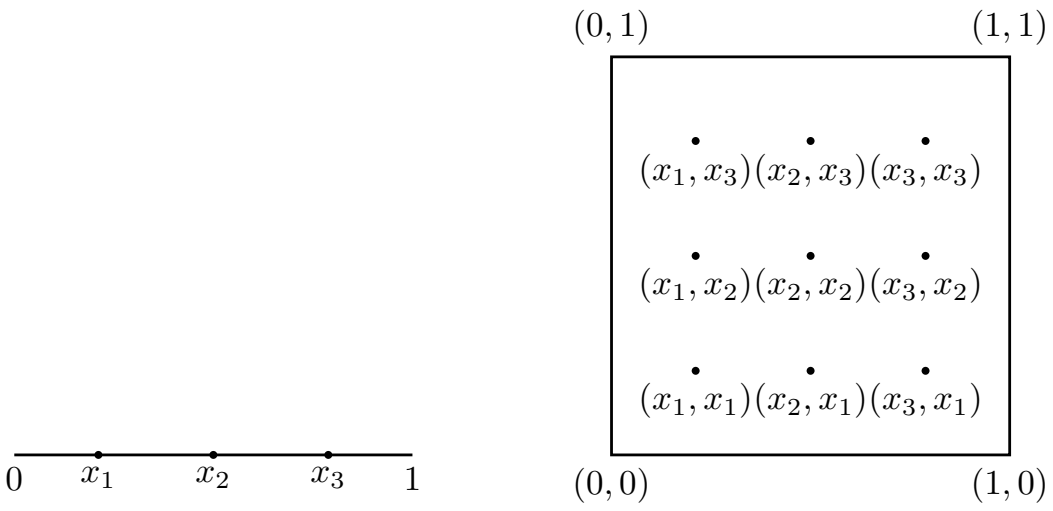


Figure 3.5: Gauss Quadrature Points for the face quadrature (left) and element quadrature (right).

We already showed how to transform this integral in (3.11), and it holds for all elements $E \in \mathcal{T}_h$ because the Jacobian is constant for all elements:

$$\forall E \in \mathcal{T}_h \qquad (\boldsymbol{A}_E)_{ij} = \int_{\hat{E}} \hat{\nabla}\hat{\phi}_j \cdot \hat{\nabla}\hat{\phi}_i \, d\hat{\mathbf{x}}.$$

Approximating it with the Gauss quadrature rule (3.21), we obtain the local stiffness matrix:

$$\forall E \in \mathcal{T}_h \qquad (\boldsymbol{A}_E)_{ij} \approx \sum_{i=1}^{N_q} \sum_{j=1}^{N_q} w_i w_j \hat{\nabla}\hat{\phi}_j(\mathbf{x}_{ij}) \cdot \hat{\nabla}\hat{\phi}_i(\mathbf{x}_{ij}). \tag{3.23}$$

The local right-hand side, transformed onto the reference element, can be approximated by:

$$\begin{aligned} \forall E \in \mathcal{T}_h \qquad (\boldsymbol{b}_E)_i &= \int_E f\phi_i \, dx = \int_{\hat{E}} f\hat{\phi}_i \, d\hat{\mathbf{x}} \\ &\approx \sum_{i=1}^{N_q^{(2D)}} w_i f\left(F_E^{-1}(\mathbf{x}_i)\right) \hat{\phi}_i\left(\mathbf{x}_i\right), \end{aligned} \tag{3.24}$$

where $f(F_E^{-1}(x_i))$ is the function value at the quadrature point x_i in the physical element.

These two local matrices also appear in continuous Galerkin methods, whereas the face integrals are unique to the DG method.

3.2.9.2 Face Contribution

We now compute the face integrals for an interior face e between two elements E^1 and E^2, with the normal vector $\mathbf{n}_e$ pointing from E^1 to E^2.

Interior Faces

In the bilinear form, there are three face integrals $I_1 + I_2 + I_3$ for the interior face e:

$$I_1 = -\int_e \{\nabla v \cdot \mathbf{n}_e\}[w]\, ds, \quad I_2 = \epsilon \int_e \{\nabla w \cdot \mathbf{n}_e\}[v]\, ds, \quad I_3 = \int_e [v][w]\, ds.$$

Recalling the definitions of the jump and average operators, where $v^{E_e^1}$ and $v^{E_e^2}$ are the traces of the function v on the face e from the elements E^1 and E^2, respectively, we can rewrite the integrals:

$$\begin{aligned}
I_1 &= \int_e \frac{1}{2}\left(\nabla v^{E_e^1} \cdot \mathbf{n}_e + \nabla v^{E_e^2} \cdot \mathbf{n}_e\right)\left(w^{E_e^1} - w^{E_e^2}\right) ds, \\
I_2 &= \epsilon \int_e \frac{1}{2}\left(\nabla w^{E_e^1} \cdot \mathbf{n}_e + \nabla w^{E_e^2} \cdot \mathbf{n}_e\right)\left(v^{E_e^1} - v^{E_e^2}\right) ds, \\
I_3 &= \int_e \frac{1}{2}\left(v^{E_e^1} - v^{E_e^2}\right)\frac{1}{2}\left(w^{E_e^1} - w^{E_e^2}\right) ds.
\end{aligned}$$

Expanding the brackets of the jump and average, the face integrals are expanded into four terms for each integral, accounting for all possible combinations of the traces from E^1 and E^2:

$$\begin{aligned}
I_1 = -\frac{1}{2} \quad & \int_e \left[\nabla v^{E_e^1} \cdot \mathbf{n}_e\, w^{E_e^1} - \nabla v^{E_e^1} \cdot \mathbf{n}_e\, w^{E_e^2} + \nabla v^{E_e^2} \cdot \mathbf{n}_e\, w^{E_e^1} - \nabla v^{E_e^2} \cdot \mathbf{n}_e\, w^{E_e^2}\right] ds, \\
I_2 = \frac{\epsilon}{2} \quad & \int_e \left[\nabla w^{E_e^1} \cdot \mathbf{n}_e\, v^{E_e^1} - \nabla w^{E_e^1} \cdot \mathbf{n}_e\, v^{E_e^2} + \nabla w^{E_e^2} \cdot \mathbf{n}_e\, v^{E_e^1} - \nabla w^{E_e^2} \cdot \mathbf{n}_e\, v^{E_e^2}\right] ds, \\
I_3 = \frac{\sigma}{4h^\beta} \quad & \int_e \left[v^{E_e^1} w^{E_e^1} \quad - v^{E_e^1} w^{E_e^2} \quad - v^{E_e^2} w^{E_e^1} \quad + v^{E_e^2} w^{E_e^2}\right] ds.
\end{aligned}$$

Taking the sum of these integrals, we can reorder the integrals so that we have the same interactions between the traces in each term:

$$I_1 + I_2 + I_3 = m_{11} + m_{12} + m_{21} + m_{22}.$$

We obtain the integrals m_{11}, m_{12}, m_{21} and m_{22} for an interior face e:

$$\begin{aligned}
m_{11} &= \frac{1}{2}\left[-\int_e \nabla v^{E_e^1} \cdot \mathbf{n}_e\, w^{E_e^1}\, ds + \epsilon \int_e \nabla w^{E_e^1} \cdot \mathbf{n}_e\, v^{E_e^1}\, ds + \frac{\sigma}{2h^\beta}\int_e v^{E_e^1} w^{E_e^1}\, ds\right], \\
m_{12} &= \frac{1}{2}\left[-\int_e \nabla v^{E_e^1} \cdot \mathbf{n}_e\, w^{E_e^2}\, ds + \epsilon \int_e \nabla w^{E_e^1} \cdot \mathbf{n}_e\, v^{E_e^2}\, ds - \frac{\sigma}{2h^\beta}\int_e v^{E_e^1} w^{E_e^2}\, ds\right], \\
m_{21} &= \frac{1}{2}\left[\int_e \nabla v^{E_e^2} \cdot \mathbf{n}_e\, w^{E_e^1}\, ds - \epsilon \int_e \nabla w^{E_e^2} \cdot \mathbf{n}_e\, v^{E_e^1}\, ds - \frac{\sigma}{2h^\beta}\int_e v^{E_e^2} w^{E_e^1}\, ds\right], \\
m_{22} &= \frac{1}{2}\left[\int_e \nabla v^{E_e^2} \cdot \mathbf{n}_e\, w^{E_e^2}\, ds - \epsilon \int_e \nabla w^{E_e^2} \cdot \mathbf{n}_e\, v^{E_e^2}\, ds + \frac{\sigma}{2h^\beta}\int_e v^{E_e^2} w^{E_e^2}\, ds\right].
\end{aligned}$$

If we now insert the test and trial functions into the bilinear form, we get the local matrices $\mathbf{M}_{11}^e$, $\mathbf{M}_{12}^e$, $\mathbf{M}_{21}^e$ and $\mathbf{M}_{22}^e$ for the interior face e:

$$\begin{aligned}
(\mathbf{M}_{11}^e)_{ij} &= \frac{1}{2}\left[-\int_e \nabla \phi_j^{E_e^1} \cdot \mathbf{n}_e \phi_i^{E_e^1}\, ds + \epsilon \int_e \nabla \phi_i^{E_e^1} \cdot \mathbf{n}_e \phi_j^{E_e^1}\, ds + \frac{\sigma}{2h^\beta}\int_e \phi_j^{E_e^1} \phi_i^{E_e^1}\, ds\right], labeleq : M11 \\
(\mathbf{M}_{12}^e)_{ij} &= \frac{1}{2}\left[-\int_e \nabla \phi_j^{E_e^1} \cdot \mathbf{n}_e \phi_i^{E_e^2}\, ds + \epsilon \int_e \nabla \phi_i^{E_e^1} \cdot \mathbf{n}_e \phi_j^{E_e^2}\, ds - \frac{\sigma}{2h^\beta}\int_e \phi_j^{E_e^1} \phi_i^{E_e^2}\, ds\right], labeleq : M12 \\
(\mathbf{M}_{21}^e)_{ij} &= \frac{1}{2}\left[\int_e \nabla \phi_j^{E_e^2} \cdot \mathbf{n}_e \phi_i^{E_e^1}\, ds - \epsilon \int_e \nabla \phi_i^{E_e^2} \cdot \mathbf{n}_e \phi_j^{E_e^1}\, ds - \frac{\sigma}{2h^\beta}\int_e \phi_j^{E_e^2} \phi_i^{E_e^1}\, ds\right], labeleq : M21 \\
(\mathbf{M}_{22}^e)_{ij} &= \frac{1}{2}\left[\int_e \nabla \phi_j^{E_e^2} \cdot \mathbf{n}_e \phi_i^{E_e^2}\, ds - \epsilon \int_e \nabla \phi_i^{E_e^2} \cdot \mathbf{n}_e \phi_j^{E_e^2}\, ds + \frac{\sigma}{2h^\beta}\int_e \phi_j^{E_e^2} \phi_i^{E_e^2}\, ds\right]. labeleq : M22
\end{aligned}$$

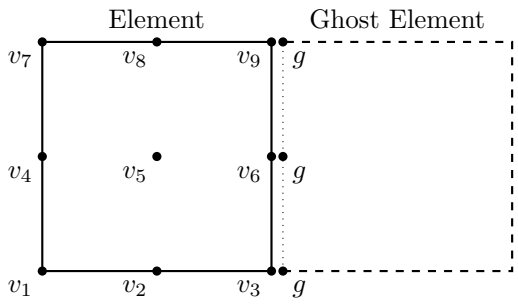


Figure 3.6: Display of Ghost Element on the domain boundary.

As shown in Section 3.2.6.2, the face integrals appearing in the local matrices can be transformed like the equations (3.15) and (3.16). For the reference face integral, we note that the basis functions $\hat{\phi}_j^{E^1_{e,k}}$ and their gradients $\nabla\hat{\phi}_j^{E^1_{e,k}}$ are evaluated at the parametrized reference face $\hat{\mathbf{x}}_{\hat{e}_k}(t)$, where $k = 1, 2, 3, 4$ represents the face index. To simplify the notation, we omit $(\hat{\mathbf{x}}_{\hat{e}_k}(t))$ in the expressions below.

$$
\begin{aligned}
(\mathbf{M}^{e_k}_{11})_{ij} &= \frac{h}{2}\int_0^1 \left[-\nabla\hat{\phi}_j^{E^1_{e,k}} \cdot \mathbf{n}_{\hat{e}_k}\hat{\phi}_i^{E^1_{e,k}} + \epsilon\nabla\hat{\phi}_i^{E^1_{e,k}} \cdot \mathbf{n}_{\hat{e}_k}\hat{\phi}_j^{E^1_{e,k}} + \frac{\sigma}{2h^\beta}\hat{\phi}_j^{E^1_{e,k}}\hat{\phi}_i^{E^1_{e,k}} \right] dt,\\
(\mathbf{M}^{e_k}_{12})_{ij} &= \frac{h}{2}\int_0^1 \left[-\nabla\hat{\phi}_j^{E^1_{e,k}} \cdot \mathbf{n}_{\hat{e}_k}\hat{\phi}_i^{E^2_{e,k}} + \epsilon\nabla\hat{\phi}_i^{E^1_{e,k}} \cdot \mathbf{n}_{\hat{e}_k}\hat{\phi}_j^{E^2_{e,k}} - \frac{\sigma}{2h^\beta}\hat{\phi}_j^{E^1_{e,k}}\hat{\phi}_i^{E^2_{e,k}} \right] dt,\\
(\mathbf{M}^{e_k}_{21})_{ij} &= \frac{h}{2}\int_0^1 \left[\nabla\hat{\phi}_j^{E^2_{e,k}} \cdot \mathbf{n}_{\hat{e}_k}\hat{\phi}_i^{E^1_{e,k}} - \epsilon\nabla\hat{\phi}_i^{E^2_{e,k}} \cdot \mathbf{n}_{\hat{e}_k}\hat{\phi}_j^{E^1_{e,k}} - \frac{\sigma}{2h^\beta}\hat{\phi}_j^{E^2_{e,k}}\hat{\phi}_i^{E^1_{e,k}} \right] dt,\\
(\mathbf{M}^{e_k}_{22})_{ij} &= \frac{h}{2}\int_0^1 \left[\nabla\hat{\phi}_j^{E^2_{e,k}} \cdot \mathbf{n}_{\hat{e}_k}\hat{\phi}_i^{E^2_{e,k}} - \epsilon\nabla\hat{\phi}_i^{E^2_{e,k}} \cdot \mathbf{n}_{\hat{e}_k}\hat{\phi}_j^{E^2_{e,k}} + \frac{\sigma}{2h^\beta}\hat{\phi}_j^{E^2_{e,k}}\hat{\phi}_i^{E^2_{e,k}} \right] dt.
\end{aligned}
$$

By applying the Gauss quadrature rule for the reference elements, see (3.19) and (3.20), we can compute the discretised local matrix on the reference element.
In the next section we will see how to assemble the global matrix from the local matrices over the reference element.

Exterior Boundary Faces

For boundary faces, the local matrices differ because a full neighbouring element is missing and the boundary conditions need to be enforced. To address this, we introduce the concept of ghost elements 3.6. A ghost element is not represented in the physical domain and has no local basis functions. The nodal values of the ghost cell are determined by the boundary conditions on the interface to the physical domain and zero for all other nodes. Test and trial functions behave differently on the ghost element, the trial function v represents the solution and takes the values of the boundary condition on the interface, while the test function w is zero on the ghost element.
We will now examine the average and jump operators especially for the boundary interface to the ghost element appearing in the face integrals. We recall the Dirichlet and Neumann boundary conditions in the continuous form:

$$v = g_D \quad \text{on} \quad \Gamma_D, \quad \nabla v \cdot \mathbf{n} = g_N \quad \text{on} \quad \Gamma_N.$$

Thus, enforcing the boundary conditions in the discrete formulation, we have for Dirichlet boundary faces:

$$v^{E_{\text{ghost}}} = g_D, \quad \nabla v^{E_{\text{ghost}}} \cdot \mathbf{n}_e = 0,$$
$$w^{E_{\text{ghost}}} = 0, \quad \nabla w^{E_{\text{ghost}}} \cdot \mathbf{n}_e = 0.$$

While for Neumann boundary faces:

$$v^{E_{\text{ghost}}} = 0, \quad \nabla v^{E_{\text{ghost}}} \cdot \mathbf{n}_e = g_N,$$
$$w^{E_{\text{ghost}}} = 0, \quad \nabla w^{E_{\text{ghost}}} \cdot \mathbf{n}_e = 0.$$

Considering a **Dirichlet boundary face** $e \in \Gamma_D$, the jump and average operators are computed by:

$$[v] = v^{E_e} - v^{E_{\text{ghost}}} = v^{E_e} - g_D, \qquad \{\nabla v\} = \nabla v^{E_e} \cdot \mathbf{n}_e,$$
$$[w] = w^{E_e}, \qquad \{\nabla w\} = \nabla w^{E_e} \cdot \mathbf{n}_e.$$

Inserting these definitions into the face integrals I_1, I_2 and I_3 for the Dirichlet boundary face, we get:

$$I_1 = \int_e \{\nabla v \cdot \mathbf{n}_e\}[w]\, ds = \int_e \nabla v^{E_e} \cdot \mathbf{n}_e w^{E_e}\, ds,$$
$$I_2 = \epsilon \int_e \{\nabla w \cdot \mathbf{n}_e\}[v]\, ds = \epsilon \int_e \nabla w^{E_e} \cdot \mathbf{n}_e v^{E_e}\, ds - \epsilon \int_e \nabla w^{E_e} \cdot \mathbf{n}_e g_D\, ds,$$
$$I_3 = \frac{\sigma}{h^\beta} \int_e [v][w]\, ds = \frac{\sigma}{h^\beta} \int_e v^{E_e} w^{E_e}\, ds - \frac{\sigma}{h^\beta} \int_e v^{E_e} g_D\, ds.$$

We see that only the local matrix $\mathbf{M}_{11}^e$ is created, as there are no interactions between different elements and we have two integrals enforcing the boundary condition, which are transferred to the right-hand side when assembling the system matrix.
The local matrix on the reference face e_k is:

$$(\mathbf{M}_{11}^{e_k})_{ij} = h \int_0^1 \left[-\nabla \hat{\phi}_j^{E_{e,k}} \cdot \mathbf{n}_{\hat{e}} \hat{\phi}_i^{E_{e,k}} + \epsilon \nabla \hat{\phi}_i^{E_{e,k}} \cdot \mathbf{n}_{\hat{e}} \hat{\phi}_j^{E_{e,k}} + \frac{\sigma}{h^\beta} \hat{\phi}_j^{E_{e,k}} \hat{\phi}_i^{E_{e,k}} \right] dt. \tag{3.25}$$

The contribution to the right-hand side is:

$$\left(\mathbf{b}_{e_k}^D\right)_i = \int_e \left(\epsilon \nabla \phi_i^{E_e} \cdot \mathbf{n}_e + \frac{\sigma}{h^\beta} \phi_i^{E_e} \right) g_D\, ds \tag{3.26}$$
$$= h \int_0^1 \left(\epsilon \hat{\nabla} \hat{\phi}_i^{E_{e,k}} \cdot \mathbf{n}_{\hat{e}_k} + \frac{\sigma}{h^\beta} \hat{\phi}_i^{E_{e,k}} \right) g_D\, dt. \tag{3.27}$$

For **Neumann boundary faces** $e \in \Gamma_N$, the integrals I_2 and I_3 disappear, while we enforce the boundary condition strongly by assuming:

$$\nabla v^{E_e} \cdot \mathbf{n}_e = \nabla v^{E_{\text{ghost}}} \cdot \mathbf{n}_e = g_N.$$

To ensure that the derivatives on the interface are equal, there must be no jump in the function values on the interface, meaning the jump $[v] = 0$. The average formulation simplifies to:

$$I_1 = \int_e \{\nabla v \cdot \mathbf{n}_e\}[w]\, ds = \int_e \frac{1}{2} (\nabla v^{E_e} \cdot \mathbf{n}_e + \nabla v^{E_{\text{ghost}}}) w^{E_e}\, ds = \int_e g_N w^{E_e}\, ds.$$

No local matrix is created because $[v] = 0$ and we only have the right-hand side contribution:

$$\left(\mathbf{b}_{e_k}^N\right)_i = \int_e \phi_i^{E_e} g_N\, ds = h \int_0^1 \hat{\phi}_i^{E_{e,k}} g_N\, dt.$$

3.2.10 Global System

We now have every ingredient to assemble the global system matrix $\mathbf{A}$ and the right-hand side vector $\mathbf{b}$. With the global-to-local mapping, we can assemble the global system matrix and right-hand side, by looping over the elements and faces, obtaining the global system

$$\mathbf{A}\mathbf{u} = \mathbf{b}.$$

The assembly of the right-hand side vector $\mathbf{b}$ is shown in Algorithm 1.

Algorithm 1 Assembly of $\mathbf{b}$ (Right-hand side contribution)

1: **Output:** $\mathbf{b}$ (global right-hand side vector)
2: Initialize $\mathbf{b} = 0$ (global vector)
3: **(1) Loop over elements: Compute volume contribution**
4: **for** each element E **do**
5: Compute local source term: $\mathbf{b} = \texttt{compute\_volume\_rhs}(E, f)$
6: Store local contribution in global vector: $\mathbf{b}^E \mathrel{+}= \mathbf{b}$
7: **end for**
8: **(2) Loop over exterior faces: Compute Dirichlet boundary contribution**
9: **for** each exterior face e **do**
10: Compute boundary integral: $\mathbf{b}_e^D = \texttt{compute\_dirichlet\_rhs}(e, g_D)$
11: Store local contribution in global vector: $\mathbf{b}^E \mathrel{+}= \mathbf{b}_e^D$
12: **end for**

The global system can then be solved with a direct or an iterative solver, by minimizing the norm of the residual

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{u}.$$

Because of the local support of the basis functions, the global matrix is sparse and can be efficiently solved with a Krylov subspace method like the generalized minimal residual method (GMRES). GMRES only requires the evaluation of the matrix-vector product $\mathbf{A}\mathbf{u}$ at each iteration, which can be done without the need of assembling and storing the global matrix. For large problems, the memory consumption of the global matrix can become a bottleneck and a matrix-free implementation with an iterative solver is more efficient.

3.2.11 Matrix-free Implementation

Using a matrix-free approach enables parallelization and reduces memory requirements. Another advantage is that it allows solving nonlinear problems, which we will see in later chapters.

The main idea of the matrix-free approach is to evaluate the matrix-vector product by looping over the local elements and faces and storing the values in a global vector. The mathematical formulation of the cell and face contributions was discussed in the previous section. An algorithmic description of how the matrix-vector product is evaluated is given in Algorithm 2.

3.3 Error Estimates

In this section, we review important results on the convergence of DG methods. The convergence behavior of finite element methods, including DG methods, is well studied in the literature. [11][8] In particular, we focus on the convergence rates for the Poisson equation.

Algorithm 2 Compute $\mathbf{Au}$ (Matrix-free DG method for Poisson)

Input: $\mathbf{u}$ (global solution vector)
Output: $\mathbf{v} = \mathbf{Au}$ (global result)
Compute and store stiffness matrix: $S =$ `compute_stiffness_matrix`()
Initialize $\mathbf{v} = 0$ (global vector)
(1) Loop over elements: Compute volume contribution
for each element E **do**
    Gather local solution $\mathbf{u}_E$
    Store local contribution in global vector: $\mathbf{v}^E \mathrel{+}= \mathbf{S}\mathbf{u}_E$
end for
(2) Loop over interior faces: Compute numerical interface terms
for each interior face e between elements E^1 and E^2 **do**
    Gather local solutions $\mathbf{u}^{E^1}$ and $\mathbf{u}^{E^2}$
    Compute interface contribution: $\mathbf{M}_{11}, \mathbf{M}_{12}, \mathbf{M}_{21}, \mathbf{M}_{22}$
    Store local contribution in global vector:
        $\mathbf{v}^{E^1} \mathrel{+}= \mathbf{M}_{11}\mathbf{u}^{E^1} + \mathbf{M}_{12}\mathbf{u}^{E^2}$
        $\mathbf{v}^{E^2} \mathrel{+}= \mathbf{M}_{21}\mathbf{u}^{E^1} + \mathbf{M}_{22}\mathbf{u}^{E^2}$
end for
(3) Loop over exterior faces: Apply Dirichlet boundary conditions
for each exterior face e **do**
    Gather local solution $\mathbf{u}^E$
    Compute boundary contribution: $\mathbf{M}_{11}$
    Store local contribution in global vector: $\mathbf{v}^E \mathrel{+}= \mathbf{M}_{11}\mathbf{u}^E$
end for

The most important aspect of convergence analysis is bounding the error between the exact solution and the numerical approximation. The error vector is defined as

$$\mathbf{e} = \mathbf{u} - \mathbf{u}_h.$$

We consider error estimates in the L^2 norm and the energy norm. The L^2 norm provides a measure of the difference between the exact and numerical solutions, while the H^1 energy norm additionally accounts for the error in the gradient of the solution and the jumps on the interfaces.
The precise derivations and proofs of the following estimates can be found in [8]. We use these statements to verify the correctness of our implementation in Section 3.4. To formulate the statements, we introduce a condition under which certain estimates hold.
In the derivation, it is shown that under a certain condition, it is required to characterize the superpenalization parameter β more precisely. That condition is the following:

Definition 3.3. *Condition 1 The approximation $\mathbf{u}_h$ of the exact solution $\mathbf{u}$ can be chosen to be continuous. Additionally the Dirichlet boundary condition is given by a continuous piecewise polynomial of degree k or the whole domain has a Neumann boundary.*

3.3.1 Error Estimates in the Energy Norm

It is possible to derive an error estimate for the energy norm, which provides an upper bound for the broken gradient seminorm, defined in (2.1).
The energy norm on $\mathcal{D}^k(\mathcal{T}_h)$ is defined as:

$$\|v\|_{\mathcal{T}} = \sum_{E \in \mathcal{E}_h} \int_E \nabla v \cdot \nabla v + J^{\sigma,\beta}(v, v).$$

Theorem 3.3. *Let $\mathbf{u} \in H^{k+1}(\mathcal{T}_h)$ be the exact solution of the Poisson equation (3.1), with $s \geq 3/2$. Assume that:*

- *The penalty parameter σ is large enough for SIPG and IIPG.*
- *For NIPG$_0$, the polynomial degree satisfies $k \geq 2$.*

Then, there exists a constant C, independent of the mesh size h, such that the error in the H^1 energy norm is bounded by:

$$\|\mathbf{u} - \mathbf{u}_h\|_{\mathcal{T}} \leq Ch^{\min(k,s)}\|\mathbf{u}\|_{H^s(\mathcal{T}_h)}.$$

This holds if

- *Condition 3.4 is satisfied and $\beta \geq \frac{1}{d-1}$, where d is the dimension of the domain, which is 2 for the considered problems.*
- *Otherwise, $\beta = \frac{1}{d-1}$.*

3.3.2 Error Estimates in the L^2 Norm

Theorem 3.4. *Assume that the error estimate for the energy norm in Theorem 3.5 holds. Then there exists a constant C, independent of the mesh size h, such that the error in the L^2 norm is bounded by:*

$$\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)} \leq Ch^{\min(k+1,s)}||p||_{H^s(\mathcal{E}_h)}.$$

This holds for

- *SIPG always*
- *NIPG and IIPG, if Condition 1 3.4 is satisfied and $\beta \geq \frac{1}{3(d-1)}$ for IIPG*

If Condition 3.4 or $\beta \geq \frac{1}{3(d-1)}$ is not satisfied, the error changes to a suboptimal rate of $h^{\min(k+1,s)-1}$.

$$\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)} \leq Ch^{\min(k+1,s)-1}||p||_{H^s(\mathcal{E}_h)}.$$

For NIPG and IIPG in the suboptimal case a special behaviour of convergence rates can be observed. In uniform meshes, the convergence rates for the suboptimal case with $\beta = \frac{1}{d-1}$ turn optimal for odd polynomial degrees. For even polynomial degrees, the convergence rates remain suboptimal. This behavior is purely a numerical observation, and the theoretical explanation is still an open question.

With these error estimates, we can now verify the correctness of the implementation before combining the DG method with domain decomposition methods in the following chapters.

3.4 Numerical Results

In this section, we review important results on the convergence of DG methods. The convergence behavior of finite element methods, including DG methods, is well studied in the literature. In particular, we focus on the convergence rates for the Poisson equation.

The most important aspect of convergence analysis is bounding the error between the exact solution and the numerical approximation. The error vector is defined as

$$\mathbf{e} = \mathbf{u} - \mathbf{u}_h.$$

We consider error estimates in the L^2 norm and the energy norm. The L^2 norm provides a measure of the difference between the exact and numerical solutions, while the H^1 energy norm additionally accounts for the error in the gradient of the solution and the jumps on the interfaces.
The precise derivations and proofs of the following estimates can be found in [8]. We use these statements to verify the correctness of our implementation in Section 3.4. To formulate the statements, we introduce a condition under which certain estimates hold.

In the derivation, it is shown that under a certain condition, it is required to characterize the superpenalization parameter β more precisely. That condition is the following:

Definition 3.4. *Condition 1 The approximation $\mathbf{u}_h$ of the exact solution $\mathbf{u}$ can be chosen to be continuous. Additionally, the Dirichlet boundary condition is given by a continuous piecewise polynomial of degree k, or the whole domain has a Neumann boundary.*

3.4.1 Error Estimates in the Energy Norm

It is possible to derive an error estimate for the energy norm, which provides an upper bound for the broken gradient seminorm, defined in (2.1).
The energy norm on $\mathcal{D}^k(\mathcal{T}_h)$ is defined as:

$$\|v\|_{\mathcal{T}} = \sum_{E\in\mathcal{E}_h} \int_E \nabla v \cdot \nabla v + J^{\sigma,\beta}(v,v).$$

Theorem 3.5. *Let $\mathbf{u} \in H^{k+1}(\mathcal{T}_h)$ be the exact solution of the Poisson equation (3.1), with $s \geq 3/2$. Assume that:*

- *The penalty parameter σ is large enough for SIPG and IIPG.*
- *For NIPG$_0$, the polynomial degree satisfies $k \geq 2$.*

Then, there exists a constant C, independent of the mesh size h, such that the error in the H^1 energy norm is bounded by:

$$\|\mathbf{u} - \mathbf{u}_h\|_{\mathcal{T}} \leq Ch^{\min(k,s)}\|\mathbf{u}\|_{H^s(\mathcal{T}_h)}.$$

This holds if

- *Condition 3.4 is satisfied and $\beta \geq \frac{1}{d-1}$, where d is the dimension of the domain, which is 2 for the considered problems.*
- *Otherwise, $\beta = \frac{1}{d-1}$.*

3.4.2 Error Estimates in the L^2 Norm

Theorem 3.6. *Assume that the error estimate for the energy norm in Theorem 3.5 holds. Then, there exists a constant C, independent of the mesh size h, such that the error in the L^2 norm is bounded by:*

$$\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)} \leq Ch^{\min(k+1,s)}||p||_{H^s(\mathcal{E}_h)}.$$

This holds for

- *SIPG always.*
- *NIPG and IIPG if Condition 3.4 is satisfied and $\beta \geq \frac{1}{3(d-1)}$ for IIPG.*

If Condition 3.4 or $\beta \geq \frac{1}{3(d-1)}$ *is not satisfied, the error changes to a suboptimal rate of* $h^{\min(k+1,s)-1}$*:*

$$\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)} \leq Ch^{\min(k+1,s)-1}||p||_{H^s(\mathcal{E}_h)}.$$

For NIPG and IIPG in the suboptimal case, a special behavior of convergence rates can be observed. In uniform meshes, the convergence rates for the suboptimal case with $\beta = \frac{1}{d-1}$ turn optimal for odd polynomial degrees. For even polynomial degrees, the convergence rates remain suboptimal.
This behavior is purely a numerical observation, and the theoretical explanation is still an open question.

With these error estimates, we can now verify the correctness of the implementation before combining the DG method with domain decomposition methods in the following chapters.

4 Domain Decomposition Methods

Domain decomposition methods are a class of algorithms for solving partial differential equations. Their main advantage lies in their naturally parallel structure, which makes them well-suited for large-scale computations. To improve performance or handle larger problems, distributing the computation across multiple processors is essential, as processor clock speeds no longer increase significantly and storage capacity remains a limiting factor. Domain decomposition methods are therefore used to solve linear and nonlinear systems of equations arising from the discretization of PDEs.

In this chapter, we introduce the formulation of a decomposed problem and examine its behavior at the interfaces between subdomains. The algorithms and statements in this chapter are based on the book on domain decomposition methods by Dolean et al. [18], a deeper theoretical analysis is given in the paper by Gander [4]. We look at the continuous formulation of the Poisson equation in 2D with two subdomains and extend it to multiple subdomains before combining it with the discontinuous Galerkin method in the next chapter.

The main difference between additive and optimized Schwarz methods lies in how gradient information is exchanged between subdomains. In the additive Schwarz method, the gradient information is transferred implicitly through the overlap, as subdomains share the overlap where the curvature of the solution is naturally captured. Contrary to OSM, the normal derivative information is shared explicitly, allowing for zero overlap between subdomains.

4.1 Additive Schwarz Methods

The Schwarz method was first introduced in the 19th century when there was no interest in parallel computing. At that time, the mathematical tools were limited to studying PDEs using only Fourier transforms, which were applicable only to simple geometries. Schwarz was interested in solving the Poisson equation on more general domains and therefore created an algorithm to iteratively solve the problem on a union of simple subdomains. [1]

Original Schwarz Algorithm

The original Schwarz method, applied to the decomposition of the domain Ω into Ω_1 and Ω_2, is as follows: Update the solutions of the subdomains, $u_1^{(k+1)}$ and $u_2^{(k+1)}$, iteratively by solving the subproblems:

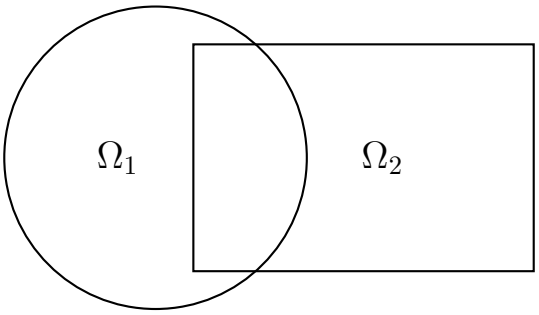


Figure 4.1: Domain decomposition of a 2D domain Ω into Ω_1 and Ω_2.

$$
\begin{aligned}
-\Delta u_1^{(k+1)} &= f && \text{in } \Omega_1, \\
u_1^{(k+1)} &= 0 && \text{on } \partial\Omega_1 \cap \partial\Omega, \\
u_1^{(k+1)} &= u_2^{(k)} && \text{on } \partial\Omega_1 \cap \partial\Omega_2,
\end{aligned}
\qquad
\begin{aligned}
-\Delta u_2^{(k+1)} &= f && \text{in } \Omega_2, \\
u_2^{(k+1)} &= 0 && \text{on } \partial\Omega_2 \cap \partial\Omega, \\
u_2^{(k+1)} &= u_1^{(k+1)} && \text{on } \partial\Omega_1 \cap \partial\Omega_2.
\end{aligned}
$$

Schwarz showed that this algorithm converges and, by doing so, proved the well-posedness of the Poisson equation for complex domains.
This algorithm is not parallel, as it applies the $k+1$ solution of the first subdomain as the boundary condition for the second subdomain in the $k+1$ iteration.
It can be easily parallelized by changing $u_2^{(k+1)} = u_1^{(k+1)}$ to $u_2^{(k+1)} = u_1^{(k)}$.

Additive Schwarz Method

To generalize this idea, we introduce the extension and restriction operators, $\mathcal{E}_i$ and $\mathcal{R}_i$, which map the solution from the subdomains Ω_i to the global domain Ω and vice versa. With the help of these operators, it is possible to define the additive Schwarz method for multiple subdomains.

Algorithm 3 Continuous Additive Schwarz Method (ASM) for the Poisson Problem

Require: Initial guess u^0, tolerance ϵ

1: **repeat**

2: **Restrict global solution:** $u_i^{(k)} = R_i u^{(k)}$ for all subdomains Ω_i.

3: **Solve local subproblems in parallel:** For each subdomain Ω_i, solve

$$
\begin{aligned}
-\Delta u_i^{(k+1)} &= f \quad \text{in } \Omega_i, \\
u_i^{(k+1)} &= 0 \quad \text{on } \partial\Omega_i \cap \partial\Omega, \\
u_i^{(k+1)} &= u^{(k)} \quad \text{on } \partial\Omega_i \cap (\Omega \setminus \Omega_i).
\end{aligned}
$$

4: **Combine subdomain solutions:** Update the global solution using

$$u^{(k+1)} = \sum_i E_i u_i^{(k+1)}.$$

5: **Check for convergence:** If $\|u^{(k+1)} - u^k\| < \epsilon$, then stop.

6: Otherwise, set $k = k + 1$ and continue.

7: **until** convergence is reached

Algorithm 3 is parallel, as the local subproblems can be solved simultaneously. The extension operator can be applied in different ways for the overlapping parts of the subdomains. One possibility is to take the average of the solutions of all subdomains contributing to that overlap. The other possibility is to cut off the overlap of the solution from all subdomains, which gives the restricted additive Schwarz method used in the next sections. For elliptic problems, non-overlapping partitions of Ω do not converge. The convergence rate improves with increasing overlap, as the ratio of shared information between the subdomains increases.

4.2 Optimized Schwarz Methods

To enhance the convergence properties of overlapping or non-overlapping domain decomposition methods, a new class of Schwarz methods has been introduced in the past decades. These optimized Schwarz methods achieve much better convergence rates by using modified transmission conditions, which enhance the exchange of information between the subdomains. The boundary conditions are changed from Dirichlet conditions to a mix of Dirichlet and Neumann conditions, known as Robin boundary conditions. By doing so, not only is the interchange of function values between the subdomains in each iteration considered, but also the exchange of the normal derivative of the solution on the boundary. Contrary to classical Schwarz methods, optimized Schwarz methods guarantee convergence for elliptic problems even for non-overlapping subdomains. The general Robin boundary conditions are given by:

$$(\nabla u \cdot \mathbf{n} + \alpha u) = g_R \quad \text{on } \partial\Omega.$$

The subproblems for the two subdomains Ω_1 and Ω_2 are then:

$$\begin{aligned} -\Delta u_1^{(k+1)} &= f && \text{in } \Omega_1, \\ u_1^{(k+1)} &= 0 && \text{on } \partial\Omega_1 \cap \partial\Omega, \\ \nabla u_1^{(k+1)} \cdot \mathbf{n}_1 + \alpha u_1^{(k+1)} &= \nabla u_2^{(k)} \cdot \mathbf{n}_1 + \alpha u_2^{(k)} && \text{on } \partial\Omega_1 \cap \Omega_2, \end{aligned}$$

$$\begin{aligned} -\Delta u_2^{(k+1)} &= f && \text{in } \Omega_2, \\ u_2^{(k+1)} &= 0 && \text{on } \partial\Omega_2 \cap \partial\Omega, \\ \nabla u_2^{(k+1)} \cdot \mathbf{n}_2 + \alpha u_2^{(k+1)} &= \nabla u_1^{(k)} \cdot \mathbf{n}_2 + \alpha u_1^{(k)} && \text{on } \partial\Omega_2 \cap \Omega_1. \end{aligned}$$

The normal vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ are the normal vectors of the subdomains Ω_1 and Ω_2, respectively. Their orientation depends on whether the subdomains overlap or not and is displayed in Figure 4.2.

The theoretical analysis of the influence of the value of α on the convergence rate of the optimized Schwarz method for the Helmholtz equation, as well as a proof of general convergence, is provided in [18].

The optimized Schwarz method can be extended to multiple subdomains, as in Algorithm 3, with the modified transmission conditions. In the next chapter, we will see how Schwarz methods can be combined with discontinuous Galerkin methods. Implementation details and numerical results will be presented.

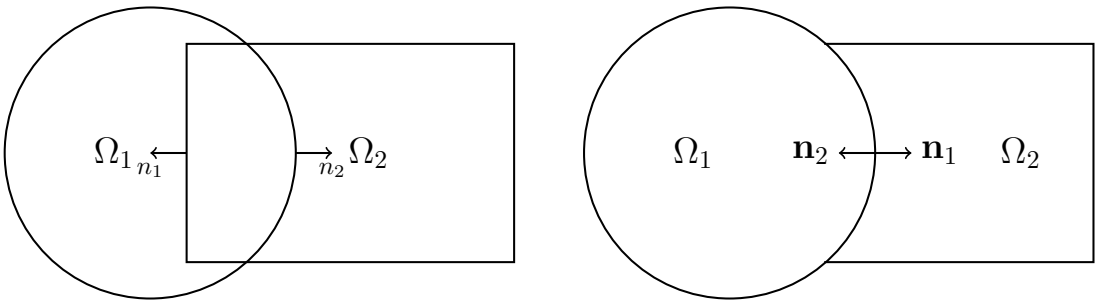


Figure 4.2: Orientation of the normal vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ for overlapping and non-overlapping subdomains.

5 Combination of Discontinuous Galerkin and Domain Decomposition Methods

The numerical experiments for solving the Poisson equation with discontinuous Galerkin methods and domain decomposition methods are conducted for the combination of discontinuous Galerkin methods with domain decomposition approaches. This combination is a natural choice, as the element-wise discontinuity of DG aligns well with the partitioning into subdomains. It enables efficient parallelization and allows for different refinement levels in different subdomains while preserving the high-order accuracy and stability of the DG method across subdomains.

5.1 Partitioning of the Domain

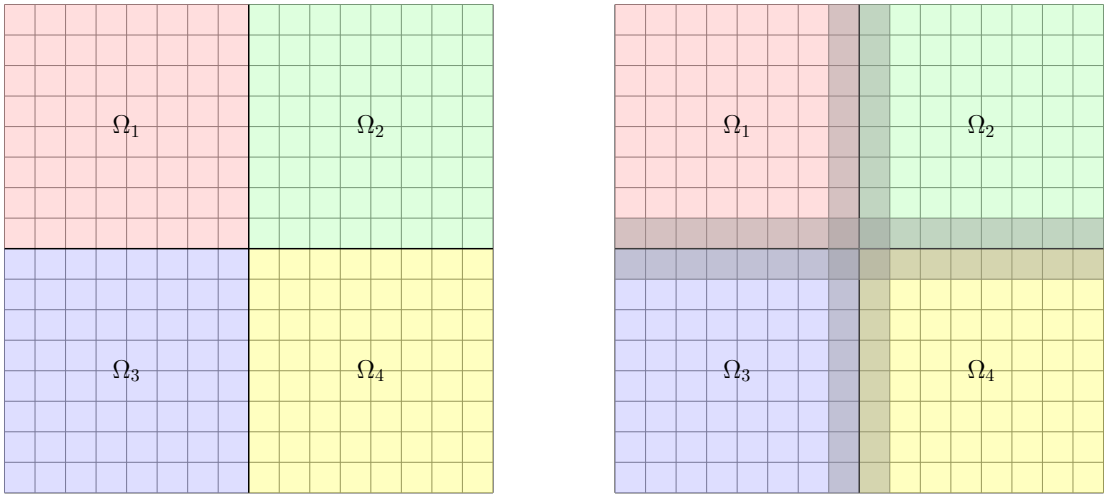
In this thesis, we consider square uniform meshes, which can easily be partitioned into subdomains. We divide the domain into $N_{\text{sub}} \times N_{\text{sub}}$ subdomains, where each subdomain is also a square. This setup allows for a simple formulation of the boundary conditions on the interfaces of the subdomains. A boundary element of one subdomain always has exactly one neighboring element in the adjacent subdomain, with the same number of degrees of freedom. In Figure 5.1, the domain is partitioned into four subdomains, showing how the overlap on the interior boundaries of the subdomains is applied, while on the exterior boundaries, there is no overlap.

5.2 Boundary Conditions of the Subdomains

Looking at the boundaries of the subdomains, we consider two types of boundaries:

- Exterior boundaries, where the subdomain boundaries coincide with the boundaries of the global domain: $\partial\Omega_i \cap \partial\Omega$.
- Interior boundaries, introduced by partitioning the domain, where boundary values are taken from the neighboring subdomain: $\partial\Omega_i \cap \Omega_j$.

To obtain a unique solution for the global domain, the boundary conditions on the exterior boundaries are set according to the Dirichlet boundary conditions of the global problem. The treatment of the boundary conditions on the interior boundaries is not as straightforward. Therefore, we examine the implementation of different types of transmission conditions on the interior boundaries in the following sections.



No Overlap Overlap = 1

Figure 5.1: Partitioning of the domain into subdomains.

5.2.1 Boundary Value Approximation on the Interior Boundaries

In Section 3.2.9.2, we introduced the concept of ghost cells and how boundary conditions are applied with a given boundary function. For interior boundaries, the boundary function $g_D(x)$ is not explicitly given, and thus, the boundary values at the quadrature points for the face integral are unknown.
For two subdomains Ω_1 and Ω_2 sharing the interior boundary $\partial\Omega_1 \cap \Omega_2$, we approximate the boundary values at the quadrature points on the interface using the values and basis functions of the neighboring element in Ω_2. The boundary element $E^1 \in \Omega_1$ has the neighboring element $E^2 \in \Omega_2$ on the interface. We use the basis functions and nodal values of E^2 to approximate the function values at the quadrature points on the interface:

$$g_D(x_q) \approx \sum_{i=1}^{N} u_i^{E^2} \phi_i^{E^2}(x_q), \quad g_N(x_q) \approx \sum_{i=1}^{N} u_i^{E^2} \nabla\phi_i^{E^2}(x_q) \cdot \mathbf{n}_e, \tag{5.1}$$

Using this, we apply Dirichlet boundary conditions on the interior boundaries of the subdomains and solve the problem with the additive Schwarz method for overlapping subdomains.

5.2.2 Robin Boundary Conditions (OSM)

The Robin boundary condition takes the form:

$$\alpha u + \nabla u \cdot \mathbf{n} = \alpha g_D + g_N = g_R \quad \text{on } \partial\Omega. \tag{5.2}$$

For the interior boundaries of Ω_1 between two subdomains Ω_1 and Ω_2, we can formulate the Robin boundary condition as:

$$\alpha v^{E^1} + \nabla v^{E^1} \cdot \mathbf{n} = \alpha v^{E^2} + \nabla v^{E^2} \cdot \mathbf{n} \quad \text{on } \partial\Omega_1 \cap \Omega_2. \tag{5.3}$$

We already explored how to enforce Dirichlet and Neumann boundary conditions when solving a DG problem. As before the boundary conditions are enforced through the face integrals on the boundary faces.

$$I_1 = \int_e \{\nabla v \cdot \mathbf{n}_e\}[w]\, ds \qquad I_2 = \epsilon \int_e \{\nabla w \cdot \mathbf{n}_e\}[v]\, ds \qquad I_3 = \frac{\sigma}{h^\beta} \int_e [v][w]\, ds$$

Because the global problem is given with Dirichlet boundary conditions, we only need to consider interior boundary faces between subdomains. The literature does not provide an approach on how to enforce Robin boundary conditions in interior penalty DG methods. Therefore, we will derive three different approaches to apply the Robin boundary conditions. The first two ideas are modifications of the Dirichlet and Neumann boundary formulations from Section 3.2.9.2. The third approach follows the implementation of Robin boundary conditions in continuous finite element methods, where the Robin boundary condition is transformed into a Neumann boundary condition.

Approach 1: Combination of Dirichlet and Neumann Boundary Conditions

Looking at the derivation of the face integrals for the Dirichlet and Neumann boundary conditions, we see that for the Neumann boundary condition, only I_1 contributes to the right-hand side, while for the Dirichlet boundary condition, I_2 and I_3 contribute to the local matrix and right-hand side. When combining these approaches, we assume

$$\nabla v^{E^1} \cdot \mathbf{n} = \nabla v^{E^2} \cdot \mathbf{n} = g_N,$$

and by that the average operator of I_1 is:

$$\{\nabla v \mathbf{n}_e\} = \frac{1}{2}\left(\nabla v^{E^1} \cdot \mathbf{n}_e + \nabla v^{E^2} \cdot \mathbf{n}_e\right) = \frac{1}{2}\left(g_N + g_N\right) = g_N.$$

For the test function values, we follow the ghost cell approach, where the test and trial function values on the ghost cell interface are:

$$v^{E_e^2} = g_D \quad \text{and} \quad w^{E_e^2} = 0,$$

The jump of the test function is determined with the factor of the Robin boundary condition α being applied:

$$[v] = \alpha v^{E^1} - \alpha v^{E^2} = \alpha v^{E^1} - \alpha g_D.$$

This leads to the following face integrals:

$$I_1 = \int_e \{\nabla v \cdot \mathbf{n}_e\}[w]\, ds = \int_e \nabla v_e^{E^2} \cdot \mathbf{n}_e w_e^{E^1}\, ds = \int_e g_N w_e^{E^1}\, ds, \tag{5.4}$$

$$I_2 = \epsilon \int_e \{\nabla w \cdot \mathbf{n}_e\}[v]\, ds = \epsilon \int_e \nabla w_e^{E^1} \cdot \mathbf{n}_e \alpha v_e^{E^1}\, ds - \epsilon \int_e \nabla w_e^{E^1} \cdot \mathbf{n}_e \alpha g_D\, ds, \tag{5.5}$$

$$I_3 = \frac{\sigma}{h^\beta} \int_e [v][w]\, ds = \frac{\sigma}{h^\beta} \int_e \alpha v_e^{E^1} w_e^{E^1}\, ds - \frac{\sigma}{h^\beta} \int_e \alpha g_D w_e^{E^1}\, ds. \tag{5.6}$$

Comparing these terms with the face integrals for the Dirichlet and Neumann boundary conditions, we see that I_1 corresponds to the Neumann part and does not contribute to the local matrix M_e^{11}. I_2 and I_3 are mixed terms, contributing to both the local matrix and the right-hand side, with each term multiplied by the coefficient α.
For interior boundary faces with Robin boundary conditions, the local matrix is:

$$\left(M_e^{11}\right)_{ij} = \alpha\Big(\epsilon \int_e \nabla \phi_{i,E_e^1} \cdot \mathbf{n}_e \phi_{j,E_e^1}\, ds + \frac{\sigma}{h^\beta} \int_e \phi_{j,E_e^1} \phi_{i,E_e^1}\, ds\Big), \tag{5.7}$$

and the local right-hand side is:

$$\left(b_e^R\right)_i = \alpha \int_e \left(\epsilon \nabla \phi_{i,E_e^1} \cdot \mathbf{n}_e + \frac{\sigma}{h^\beta} \phi_{i,E_e^1}\right) g_D\, ds. \tag{5.8}$$

Approach 2: Modification of the Dirichlet and Neumann Boundary Conditions

Like before we use a ghost element to apply the boundary values. The values on the ghost cell for the test function v are:

$$v^{E^2} = g_D, \quad \nabla v^{E^2} \cdot \mathbf{n} = g_N, \tag{5.9}$$

while the trial function w remains zero on the ghost cell. Contrary to the Neumann boundary condition we do not assume that the normal derivative on the boundary element is equal to the Neuman boundary condition. This leads to a change in the average operator of I_1 because there the average there is:

$$\{\nabla v \mathbf{n}_e\} = \frac{1}{2}\left(\nabla v^{E^1} \cdot \mathbf{n}_e + \nabla v^{E^2} \cdot \mathbf{n}_e\right) = \frac{1}{2}\left(\nabla v^{E^1} \cdot \mathbf{n}_e + g_N\right),$$

while the jump of the test function remains:

$$[v] = \alpha v^{E^1} - \alpha g_D.$$

The face integrals for the Robin boundary conditions are:

$$\begin{aligned}
I_1 &= \int_e \{\nabla v \cdot \mathbf{n}_e\}[w]\, ds = \frac{1}{2}\int_e \nabla v^{E^1} \cdot \mathbf{n}_e w_e^{E^1}\, ds + \frac{1}{2}\int_e g_N w_e^{E^1}\, ds, \\
I_2 &= \epsilon \int_e \{\nabla w \cdot \mathbf{n}_e\}[v]\, ds = \epsilon \int_e \nabla w_e^{E^1} \cdot \mathbf{n}_e \alpha v_e^{E^1}\, ds - \epsilon \int_e \nabla w_e^{E^1} \cdot \mathbf{n}_e \alpha g_D\, ds, \\
I_3 &= \frac{\sigma}{h^\beta}\int_e [v][w]\, ds = \frac{\sigma}{h^\beta}\int_e \alpha v_e^{E^1} w_e^{E^1}\, ds - \frac{\sigma}{h^\beta}\int_e \alpha g_D w_e^{E^1}\, ds.
\end{aligned}$$

Comparing the face integrals to the face integrals when enforcing Dirichlet boundary conditions, we see that for I_1 a right-hand side term is added. If we sort the terms in their contribution to the local matrix M_e^{11} and the local right-hand side b_e^R, we get:

$$\left(M_e^{11}\right)_{ij} = \Big(\frac{1}{2}\int_e \nabla v^{E^1} \cdot \mathbf{n}_e w_e^{E^1}\, ds + \alpha\epsilon \int_e \nabla \phi_{i,E_e^1} \cdot \mathbf{n}_e \phi_{j,E_e^1}\, ds + \frac{\alpha\sigma}{h^\beta}\int_e \phi_{j,E_e^1}\phi_{i,E_e^1}\, ds\Big), \tag{5.10}$$

$$\left(b_e^R\right)_i = \left(-\frac{1}{2}\int_e g_N w_e^{E^1}\, ds, +\alpha\epsilon \int_e \nabla \phi_{i,E_e^1} \cdot \mathbf{n}_e g_D\, ds + \frac{\alpha\sigma}{h^\beta}\phi_{i,E_e^1} g_D\right) ds. \tag{5.11}$$

Approach 3: Transformation of Robin Boundary Conditions to Neumann Boundary Conditions

For the third approach we take a more general formulation of the Robin boundary condition and introduce a factor β in front of the Neumann part of the boundary condition. This can easily be translated to the previous formulation of the Robin boundary condition, by dividing the Boundary condition by β.
The main idea is to transform the Robin boundary condition to a Neumann boundary condition through the following steps: We have the Robin boundary condition:

$$\alpha v^{E^1} + \beta \nabla v^{E^1} \cdot \mathbf{n} = \alpha v^{E^2} + \beta \nabla v^{E^2} \cdot \mathbf{n} \quad \text{on } \partial\Omega_1 \cap \Omega_2,$$

substract αv^{E^1} from both sides we obtain the following Neumann boundary condition:

$$\nabla v^{E^1} \cdot \mathbf{n} = \frac{\alpha}{\beta}(v^{E^2} - v^{E^1}) + \nabla v^{E^2} \cdot \mathbf{n}. \tag{5.12}$$

Where the function values on the ghost cell interface are:

$$v^{E^2} = g_D, \quad \nabla v^{E^2} \cdot \mathbf{n} = g_N.$$

The trial function values on the ghost cell interface stay zero as before and the jump of the test function is just like before:

$$[v] = v^{E^1} - v^{E^2} = v^{E^1} - g_D.$$

Inserting the derived Neumann boundary condition in (5.12) into the face integral I_1 we get:

$$\begin{aligned} I_1 &= -\int_e \{\nabla v \cdot \mathbf{n}_e\}[w]\, ds = -\int_e \frac{1}{2}\Big(\frac{\alpha}{\beta}(g_D - v^{E^1}) + g_N + g_N\Big) w_e^{E^1}\, ds \\ &= \frac{\alpha}{2\beta}\int_e v^{E^1} w_e^{E^1}\, ds - \frac{\alpha}{2\beta}\int_e g_D w_e^{E^1}\, ds - \int_e g_N w_e^{E^1}\, ds. \end{aligned}$$

The remaining face integrals are computed as before with just the changed values in the jump of the test function.

$$\begin{aligned} I_2 &= \epsilon \int_e \{\nabla w \cdot \mathbf{n}_e\}[v]\, ds = \epsilon \int_e \nabla w_e^{E^1} \cdot \mathbf{n}_e v_e^{E^1}\, ds - \epsilon \int_e \nabla w_e^{E^1} \cdot \mathbf{n}_e \alpha g_D\, ds, \\ I_3 &= \frac{\sigma}{h^\beta}\int_e [v][w]\, ds = \frac{\sigma}{h^\beta}\int_e v_e^{E^1} w_e^{E^1}\, ds - \frac{\sigma}{h^\beta}\int_e \alpha g_D w_e^{E^1}\, ds. \end{aligned}$$

Splitting the face integrals containing boundary information of the ghost cell to the right-hand side and using the basis function representation of the test and trial functions, we get:

$$\begin{aligned} \left(M_e^{11}\right)_{ij} &= \Big(\frac{\alpha}{2\beta} + \frac{\sigma}{h^\beta}\Big)\int_e v^{E^1} w_e^{E^1}\, ds + \epsilon \int_e \nabla v^{E^1} \cdot n w_e^{E^1}\, ds, \\ \left(b_e^R\right)_i &= \Big(\frac{\alpha}{2\beta} + \frac{\sigma}{h^\beta}\Big)\int_e g_D w_e^{E^1}\, ds + \epsilon \int_e g_N w_e^{E^1}\, ds. \end{aligned}$$

5.3 Implementation of the Schwarz Methods

The implementation of the additive Schwarz method and the optimized Schwarz method differs only in the applied boundary conditions. Therefore, the same method can be used for both boundary conditions, with the only difference being the functions used to compute the local matrix, right-hand side, and residual. For the additive Schwarz method, we use the local forms (3.27) and (3.25) for the right-hand side and local matrix, while for the optimized Schwarz method, we use (5.8) and (5.7) instead. The computation of the residual is also done in a local manner, by looping over the subdomains and their non-overlapping cells. The scheme for the Schwarz methods is shown in Algorithm 4.

Algorithm 4 Schwarz Domain Decomposition with DG for Poisson

Input: Initial global solution $\mathbf{u}$, subdomain partitioning, α, tolerance tol
Output: Approximate global solution $\mathbf{u}^{(k)}$
Initialize $\mathbf{u}^{(0)} = \mathbf{u}$ and define stopping criterion $\|\mathbf{r}^{(k)}\| < \text{tol}$
(1) Iterative Schwarz loop
for $k = 1, 2, \ldots$ until convergence **do**
    Initialize $\mathbf{u}^{(k)} = 0$ (reset global solution for the new iteration)
    (2) Loop over subdomains: Solve local DG problems
    for each subdomain Ω_i **do**
        Restrict global solution to local subdomain: $\mathbf{u}_{\Omega_i}^{(k-1)} = \mathcal{R}_i \mathbf{u}^{(k-1)}$
        Compute local right-hand side : $\mathbf{b}_i = \texttt{compute\_rhs}(\mathbf{u}_{\Omega_i}^{(k-1)}, \alpha)$
        Solve local problem using GMRES: $\mathbf{u}_{\Omega_i} = \texttt{gmres}(\texttt{compute\_Au}(\alpha), \mathbf{b}_i, \mathbf{u}_{\Omega_i}^{(k-1)})$
        Extend local solution back to global domain: $\mathbf{u}^{(k)} \mathrel{+}= \mathcal{E}_i \mathbf{u}_{\Omega_i}$
    end for
    (3) Compute residual and check convergence
    Compute global residual: $\mathbf{r}^{(k)} = \texttt{compute\_residual}(\mathbf{u}^{(k)})$
    if $\|\mathbf{r}^{(k)}\| < \text{tol}$ **then**
        Break (solution has converged)
    end if
end for

5.4 Numerical Results

The Poisson equation is solved on the domain $\Omega = [-1, 2] \times [-1, 2]$, ensuring that the solution

$$u(x, y) = x(1 - x) + y(1 - y)e^{-(x^2+y^2)}$$

has non-homogeneous boundary conditions. The exterior boundaries are always set Dirichlet conditions.

We investigate the convergence behavior of different Schwarz methods for the Poisson equation and evaluate the influence of the penalty parameter σ and the factor α in front of the Dirichlet part of the Robin boundary condition. Both parameters influence the stability of the DG method on subdomain interfaces, which is crucial for the convergence of domain decomposition methods in terms of the number of iterations required to reach a given tolerance and the accuracy of the global solution.

The experiments are performed with polynomial degree $k = 2$ on a mesh with 8×8 elements and 4 subdomains, if not stated otherwise. The Schwarz iteration starts with $u_0 = 0$ as an initial guess, testing the method's convergence as the initial guess is far from the exact solution. For reference, the global problem is solved with DG without domain decomposition, using the same symmetry parameter as the local problems and always a penalty parameter $\sigma = 18$.

5.4.1 Progress of the Solution in the Optimized Schwarz Method

The OSM method converges after roughly 300 iterations to the desired accuracy, as shown in Figure 5.2.
The following Figures 5.3 - 5.5 illustrate the evolution of the global solution throughout the iterations of the optimized Schwarz method for the Poisson equation. The left plots shows the numerical solution of the global problem and the right plots show the spatial error between the current numerical solution and the analytical solution. The highest error initially appears on the subdomain interfaces and is iteratively reduced. In the early iterations, the error is primarily concentrated at the subdomain interfaces, until it is reduced to the same magnitude as the local DG discretization error, meaning that the subdomain interfaces no longer contribute significantly to the overall error.

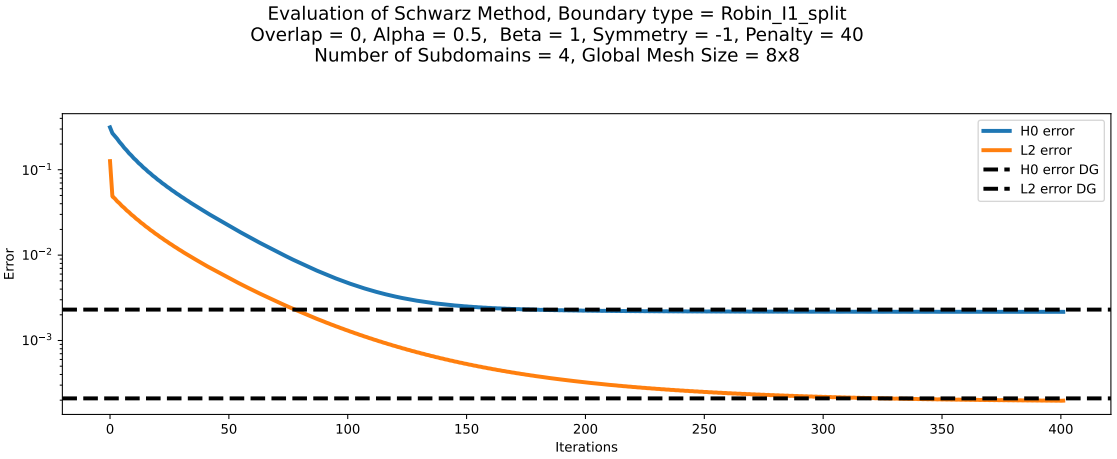


Figure 5.2: Convergence of non-overlapping optimized Schwarz method with $\alpha = 0.5$, $\beta = 1$, $\sigma = 40$ and SIPG solving the subproblems

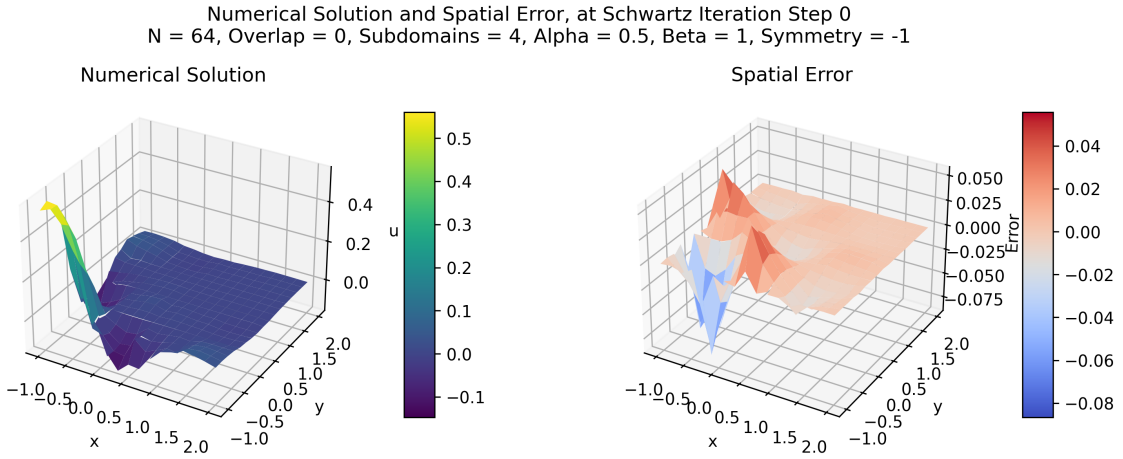


Figure 5.3: Solution of the optimized Schwarz method at timestep 0

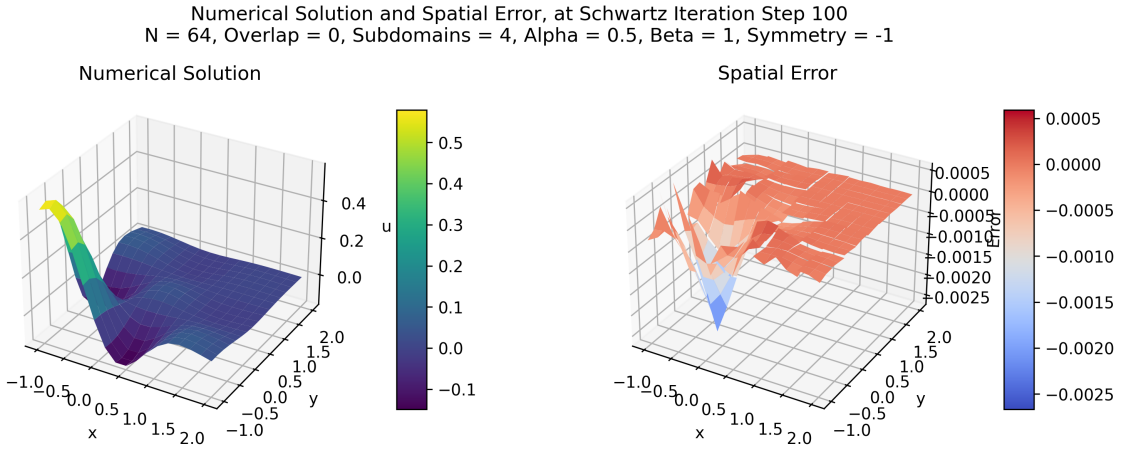


Figure 5.4: Solution of the optimized Schwarz method at timestep 100

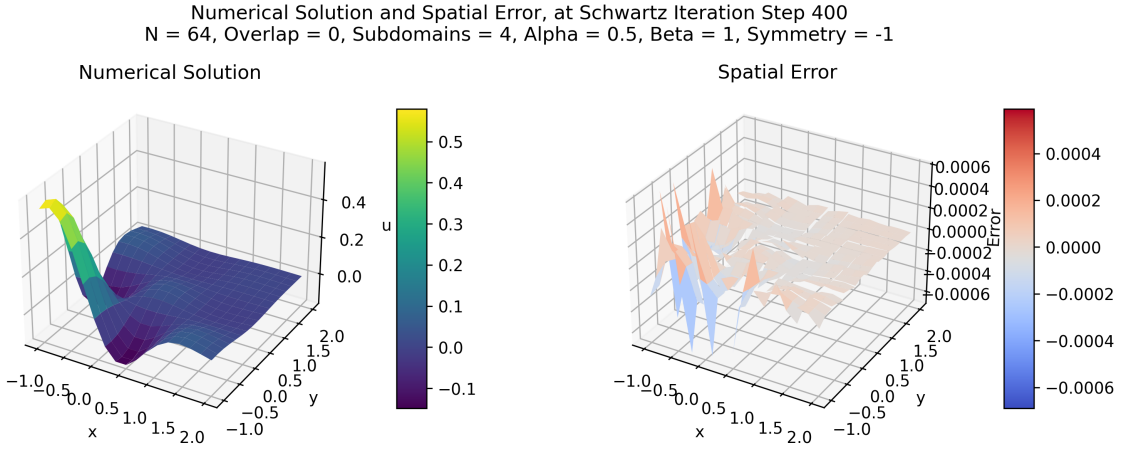


Figure 5.5: Fully converged solution of the optimized Schwarz method at iteration 400

5.4.2 Comparing the Convergence of the ASM and OSM for Overlapping Subdomains

The additive Schwarz method with minimal overlap of one element and Dirichlet boundary conditions on the subdomain interfaces, converges to the global reference DG solution in roughly 40 iterations, as shown in Figure 5.6. For the optimized Schwarz method, we introduce the Robin boundary conditions implemented with Approach 2. The method converges in roughly 30 iterations, as shown in Figure 5.7. This is because of the explicit introduction of the normal derivative of the solution in the interface conditions, which improves the convergence of the method.

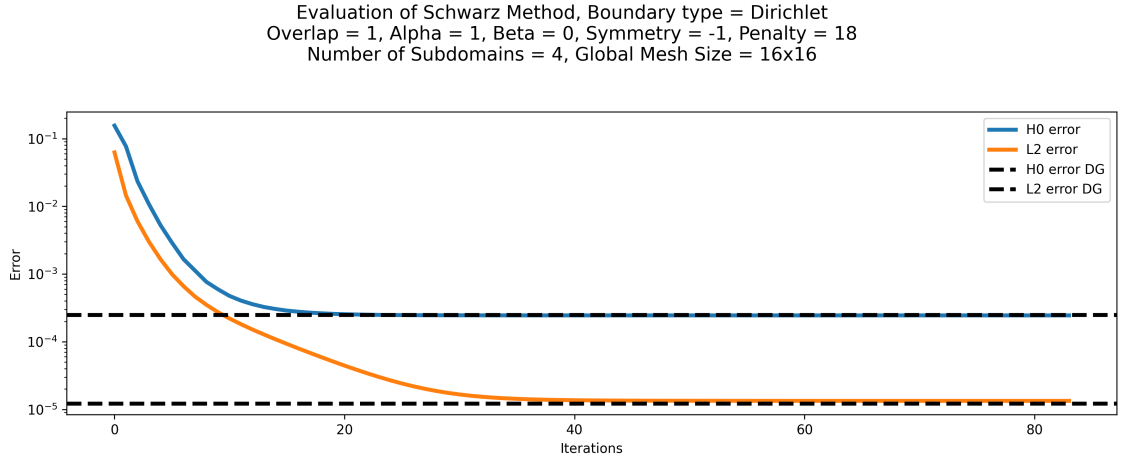


Figure 5.6: Convergence of the additive Schwarz method.

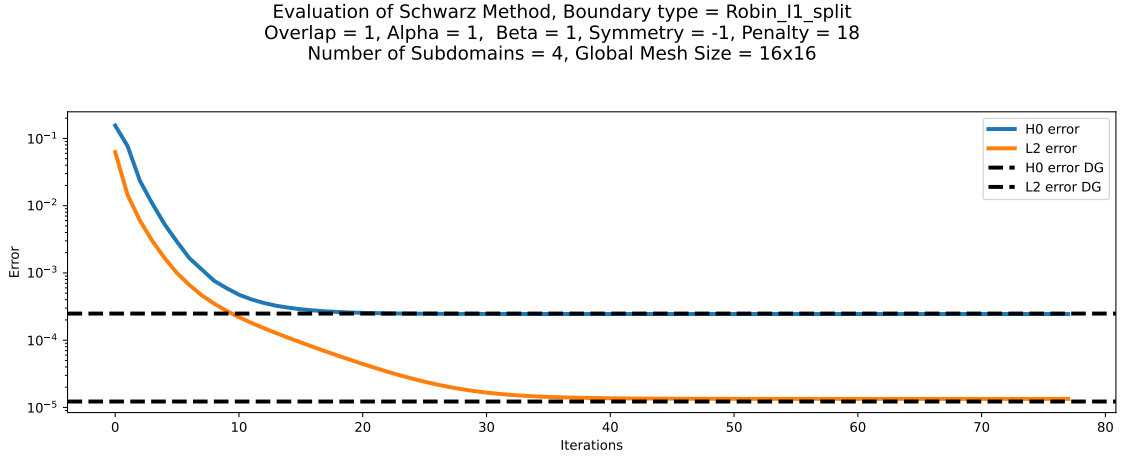


Figure 5.7: Convergence of the optimized Schwarz method.

5.4.3 Convergence of the Optimized Schwarz Method with Varying Penalty Parameters

In this section, we investigate the effect of the penalty parameter σ on the convergence of the optimized Schwarz method. We test the behavior for all three penalty methods IIPG, NIPG and SIPG, using the same setup as before. The interface conditions are implemented using Approach 2, as described in Equation (5.10).

We observe that if the penalty parameter for the local subproblems is set to the same value as in the global problem, the methods do not converge to the same error level as the

global DG solution. By increasing the penalty parameter, this can be resolved and even improve the error of the problem solved with OSM. However, the drawback of increasing the penalty parameter is that it requires more iterations to reach the same tolerance.

Looking at the face integral coming from the stability term we see the relation between the penalty parameter σ and the Robin boundary condition parameter α:

$$I_3 = \frac{\alpha \cdot \sigma}{h^\beta} \left(\int_e v_e^{E^1} w_e^{E^1} \, ds - \int_e g_D w_e^{E^1} \, ds \right).$$

The product $\alpha \cdot \sigma$ is weighting the stability term on the interfaces. This suggests that we look at the effect of changing both parameters in relation to each other in a following experiment.

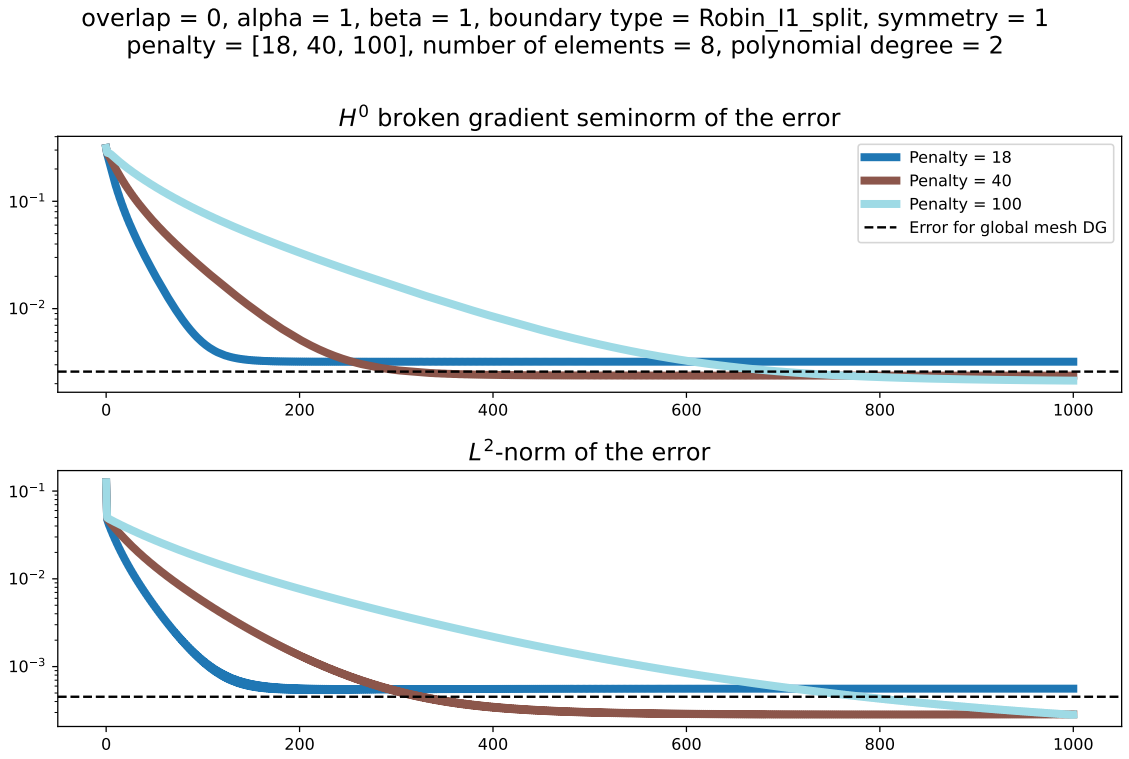


Figure 5.8: Convergence of the OSM with varying penalty parameters using IIPG.

5.4.4 Comparison of Interface Condition Implementation Approaches

In this experiment, we compare the performance of the different interface condition implementations. Approach 1 and Approach 3 behave similarly, with both methods needing a higher penalty parameter to reach the reference error level. While Approach 2 converges faster and with a lower penalty parameter, as shown in Figure 5.9.

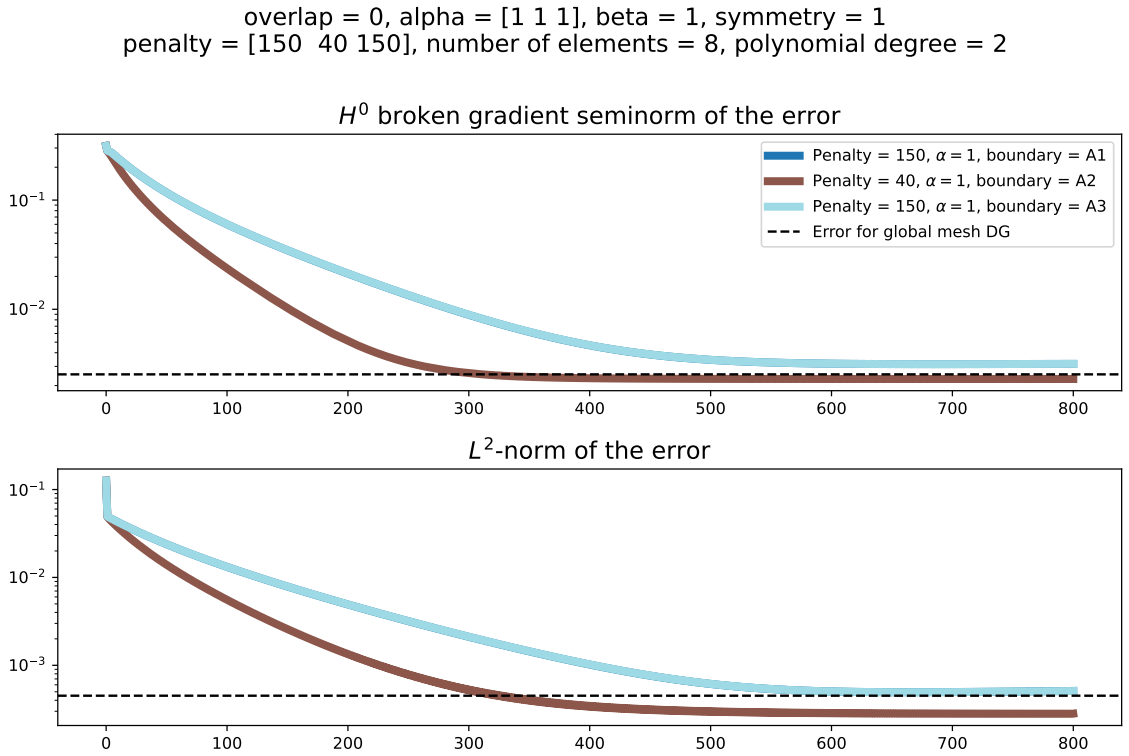


Figure 5.9: Convergence of the OSM with different interface condition implementations.

5.4.5 Optimizing the Convergence of the Optimized Schwarz Method with Varying Robin Boundary Condition and Penalty Parameter

In this experiment, we compare the interaction between the penalty parameter σ and the factor α in front of the Dirichlet part of the Robin boundary condition, with the interface conditions implemented using Approach 2. The parameter α also acts as a penalty parameter, but only on the boundaries of the subdomains. It influences both the convergence rate and the accuracy of the solution.
Figure 5.10 presents results for different combinations of α and σ to identify the optimal balance between convergence speed and accuracy. We can also see that changing α requires adjusting the penalty parameter σ to maintain the desired accuracy. This dependency arises because α changes the boundary condition and therefore the function space of the local problems differ from the global problem. To ensure a coercive bilinear form in the local subproblems, the penalty parameter σ must be adjusted according to the modified boundary conditions. A good choice of the parameters is $\alpha = 0.5$ and $\sigma = 40$, as it converges fast and reaches the reference error level.

This result is strongly dependent on the applied implementation of the Robin boundary conditions.

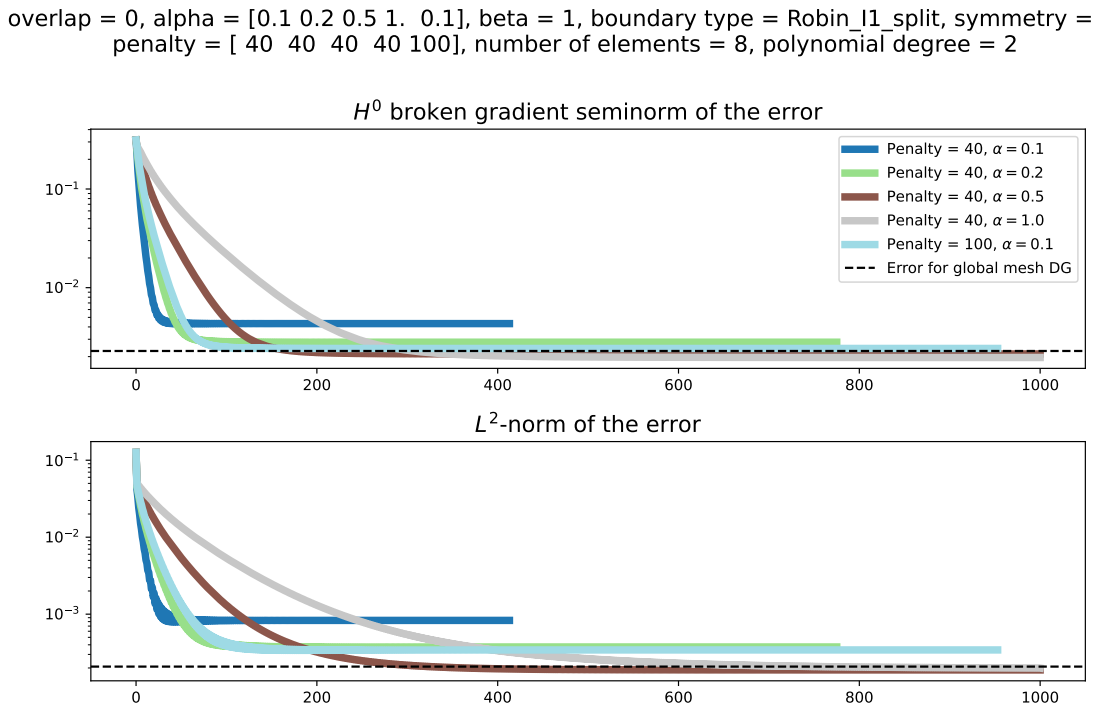


Figure 5.10: Convergence of the OSM with varying penalty parameters and Robin boundary condition factors.

6 Allen-Cahn Equation

The Allen-Cahn equation is a reaction-diffusion equation that arises in materials science for modeling phase separation in binary alloys. The equation has the form of the heat equation with an additional nonlinear perturbation term:

$$\frac{1}{K}\frac{\partial u}{\partial t} - \Delta u + \frac{1}{\varepsilon^2} f(u) = 0 \quad \text{in } \Omega \times (0,T), \tag{6.1}$$

$$u = 0 \quad \text{on } \partial\Omega \times (0,T), \tag{6.2}$$

$$u(\mathbf{x},0) = u_0(\mathbf{x}) \quad \text{in } \Omega, \tag{6.3}$$

where $\Omega \subset \mathbb{R}^d$ is a bounded domain, and

$$f(u) = u(1-u)(1-2u) - \mu 6u(1-u).$$

The parameters K and ε are positive constants.

For the initial condition u_0, we consider the function:

$$u_0(\mathbf{x}) = \frac{1}{2}\left(1 + \tanh \frac{R_0 - \sqrt{x_1^2 + x_2^2}}{2\varepsilon}\right), \tag{6.4}$$

where R_0 is the initial radius of the phase separation.
The two phases are represented by $u = 0$ and $u = 1$, and the parameter ε determines the width of the interface between the two phases. In the considered case, the parameter μ is set to 0, which causes the phase separation radius to shrink over time. As $\varepsilon \to 0$, the equation becomes stiffer, and the interface between the two phases becomes thinner, making it an interesting case for evaluating numerical methods.
Figure 6.1 shows a possible initial solution of the Allen-Cahn equation.

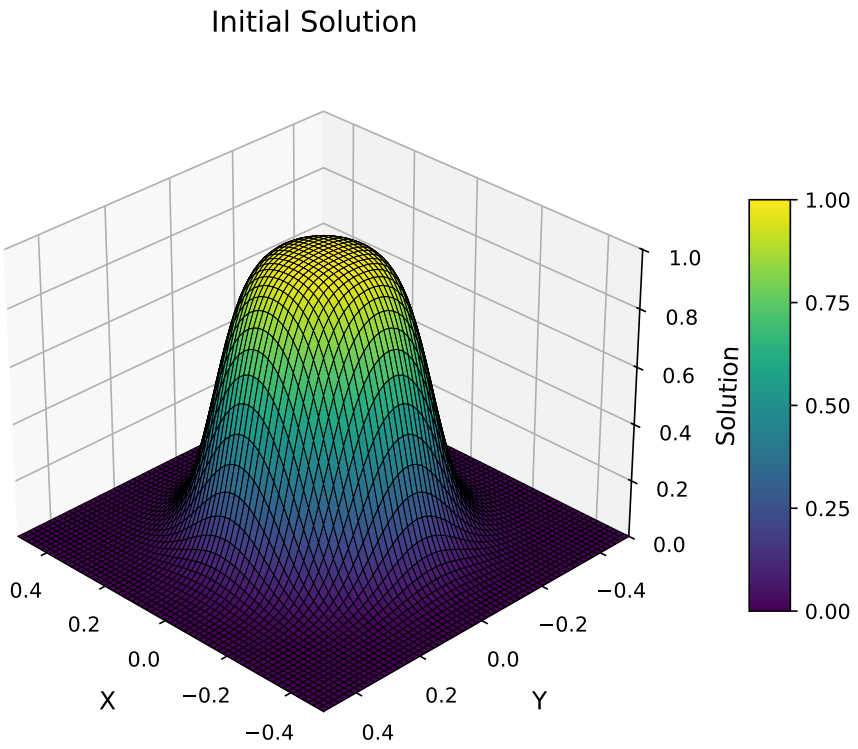


Figure 6.1: Initial solution of the Allen-Cahn equation with $R_0 = 0.25$, $\varepsilon = 0.04$ and $K = 1$.

6.1 Numerical Solution of the Allen-Cahn Equation

Because of the nonlinearity, iterative solvers are required to handle the resulting system efficiently. Since the solution of the Allen-Cahn equation has sharp interfaces, decomposing the domain into subdomains helps distribute the computational work while maintaining high accuracy. This makes the combination of the discontinuous Galerkin method with domain decomposition methods a well-suited approach for solving the equation. We introduce the DG discretization of the Allen-Cahn equation as presented by Feng et al. in [19], which is modified in the next step. To solve the equation without requiring a nonlinear solver, we apply an implicit-explicit (IMEX) scheme to the equation. Finally, we discuss how to apply domain decomposition methods to the formulated DG scheme.

6.1.1 Time Discretization

When discretizing the equation (6.1), we use the backward Euler operator

$$\partial_t u \approx \frac{u^{n+1} - u^n}{dt},$$

with dt being the time step size, to obtain the semi-discrete form of the Allen-Cahn equation:

$$\frac{1}{K}\frac{u^{n+1} - u^n}{dt} - \Delta u^{n+1} + \frac{1}{\varepsilon^2} f(u^{n+1}) = 0 \quad \text{in } \Omega, \tag{6.5}$$

where u^n is the solution at time $t_n = n \cdot dt$ and u^{n+1} is the solution at time $t_{n+1} = (n+1) \cdot dt$. It must hold that $dt \ll \varepsilon^2$ to ensure the stability of the solution.
Multiplying equation (6.5) by K and dt, and rearranging the terms yields:

$$u^{n+1} - u^n + dtK(-\Delta u^{n+1}) + \frac{dtK}{\varepsilon^2} f(u^{n+1}) = 0. \tag{6.6}$$

6.1.2 Discretization of the Laplacian Using the Discontinuous Galerkin Method

To discretize the spatial domain, we use the discontinuous Galerkin method. We therefore need to construct a weak formulation and find the solution in the broken Sobolev space $H^s(\mathcal{T}_h)$, $s \geq 3/2$.
In the following, the inner product over the mesh $\mathcal{T}_h$ is defined as:

$$(u, v)_{\mathcal{T}_h} = \sum_{E \in \mathcal{T}_h} \int_E u \cdot v \, dx,$$

and for any subset $\Gamma_h \subseteq \Gamma$ of all element faces, the inner product is defined as:

$$\langle u, v \rangle_{\Gamma_h} = \sum_{e \in \Gamma_h} \int_e u \cdot v \, ds.$$

Multiplying equation (6.6), where $u^{n+1}, u^n \in \mathcal{D}^k(\mathcal{T}_h)$, by a test function $v \in H^s(\mathcal{T}_h)$ and integrating over the domain Ω yields:

$$(u^{n+1}, v)_{\mathcal{T}_h} - (u^n, v)_{\mathcal{T}_h} + dtK(-\Delta u^{n+1}, v)_{\mathcal{T}_h} + \frac{dtK}{\varepsilon^2}(f(u^{n+1}), v)_{\mathcal{T}_h} = 0. \tag{6.7}$$

The negative Laplacian term can be approximated by the interior penalty DG bilinear form $a_\epsilon(v, w)$ introduced in Chapter 3:

$$a_\epsilon(v, w) = (\nabla v, \nabla w)_{\mathcal{T}_h} - \langle \nabla v \cdot \mathbf{n}_e, [w] \rangle_{\Gamma_h} + \epsilon \langle \nabla w \cdot \mathbf{n}_e, [v] \rangle_{\Gamma_h} + \frac{\sigma}{h^\beta} \langle [v], [w] \rangle_{\Gamma_h},$$

with the known parameters $\epsilon, \sigma, \beta > 0$.
The weak formulation (6.7) can be rewritten as:

$$(u^{n+1}, v)_{\mathcal{T}_h} - (u^n, v)_{\mathcal{T}_h} + dtK a_\epsilon(u^{n+1}, v) + \frac{dtK}{\varepsilon^2}(f(u^{n+1}), v)_{\mathcal{T}_h} = 0. \tag{6.8}$$

Using the DG formulation, the solution is represented element-wise, which allows the test and trial functions to be expressed locally within each element. The description of the test and trial functions in one element via the basis functions ϕ_i, ϕ_j and the coefficients u_i is:

$$u^{n+1} = \sum_{j=1}^{N} u_j^{n+1} \phi_j, \qquad u^n = \sum_{j=1}^{N} u_j^n \phi_j, \qquad v = \sum_{i=1}^{N} \phi_i,$$

where $N = (k+1)^2$ is the number of nodes in one element. We can rewrite the weak formulation (6.8) as a system of equations:

$$(\phi_j, \phi_i)_{\mathcal{T}_h} u_j^{n+1} - (\phi_j, \phi_i)_{\mathcal{T}_h} u_j^n + dtK a_\epsilon(\phi_j, \phi_i) u_j^{n+1} + \frac{dtK}{\varepsilon^2}(f(u_j^{n+1}\phi_j), \phi_i)_{\mathcal{T}_h} = 0. \tag{6.9}$$

If the equation is satisfied, the solution u^{n+1} is determined. The objective is to minimize the residual of the equation, which is the left-hand side, and find the minimal residual solution. Because of the nonlinearity of the equation, a suitable nonlinear iterative solver is required to find the solution.

6.1.3 Implicit-Explicit Scheme (IMEX) Applied to the Allen-Cahn Equation

Another approach is to apply an IMEX scheme to equation (6.9) by using the previous solution u^n in the nonlinear term. This prevents the need for a complicated nonlinear solver and reduces the computational cost. The idea is to treat some terms implicitly to

improve the stability of the solution, while other terms are treated explicitly to reduce the computational cost or simplify the handling of the nonlinear term. The diffusion term is treated implicitly, and the nonlinear term is treated explicitly, which avoids the need for a nonlinear solver.
By bringing the terms containing u^n to the right-hand side of equation (6.8), we obtain the system of equations:

$$(\phi_j, \phi_i)_{\mathcal{T}_h} u_j^{n+1} + dtKa_\epsilon(\phi_j, \phi_i) u_j^{n+1} = (\phi_j, \phi_i)_{\mathcal{T}_h} u_j^n - \frac{dtK}{\varepsilon^2}(f(u_j^n \phi_j), \phi_i)_{\mathcal{T}_h}. \tag{6.10}$$

This equation can now be solved in each time step n in a matrix-free manner, as described in Section 3.2.11.
The DG bilinear form also contributes to the right-hand side of the equation and enforces the boundary condition on the exterior faces of the domain. The time-stepping scheme for solving each time step is shown in Algorithm 5.

Algorithm 5 Time-stepping scheme for solving the linearized Allen-Cahn equation

Require: u^0, N_{timestep}, tol
1: **for** $n = 0$ to $N_{\text{timestep}} - 1$ **do**
2: $u_{\text{old}} \leftarrow u^n$
3: Compute right-hand side: rhs $\leftarrow$ ComputeRHS(u_{old})
4: Solve the linear system:

$$u^{n+1} \leftarrow \text{GMRES}(\text{ComputeAu}, u_{\text{old}}, \text{tol})$$

5: Update solution: $u^n \leftarrow u^{n+1}$
6: **end for**

Right-Hand Side

To compute the right-hand side, we have to consider three different terms: the boundary contribution from the DG bilinear form, the mass matrix contribution from the time discretization, and the nonlinear term. The boundary contribution of the DG bilinear form remains unchanged from previous chapters. The mass matrix contribution is also straightforward, obtained by computing the local product of the mass matrix with the corresponding node values of the element and transforming it into the global index space.

The nonlinear term is more complex to compute because the nonlinear function $f(u_j^n \phi_j)$ needs to be evaluated at the quadrature points of the element.

$$\begin{aligned}(f(u_j^n \phi_j), \phi_i)_{\mathcal{T}_h} &= \sum_{E \in \mathcal{T}_h} \int_E f(u_j^n \phi_j) \phi_i \, dx \\ &\approx \sum_{E \in \mathcal{T}_h} \sum_{q=1}^{N_q} w_q f(u_j^n \phi_j(\mathbf{x}_q)) \phi_i(\mathbf{x}_q).\end{aligned}$$

By looping over all elements, we obtain the nonlinear contribution as shown in Algorithm 6.

Algorithm 6 Compute nonlinear right-hand side (RHS)

Require: u^n
Initialize rhs_nonlinear $\leftarrow 0$ (size N_{nodes})
for each element E in the mesh **do**
    node_idx $\leftarrow$ local to global index mapping
    Initialize $f \leftarrow 0$ (size N_{quad})
    for each quadrature point q **do**
        $t = 0$
        for each basis function ϕ_j **do**
            $t+ = \phi_j(x_q) \cdot u[\text{node\_idx}[j]]$
        end for
        $f[q] = w_q \cdot t(1-t)(1-2t)$
    end for
    for each basis function ϕ_i **do**
        for each quadrature point q **do**
            rhs_nonlinear[node_idx[i]]$+ = f[q] \cdot \phi_i(x_q)$
        end for
    end for
end for
rhs_nonlinear$* = \frac{-K \cdot dt}{\varepsilon^2}$
return rhs_nonlinear

Matrix-Free Solver

Once the right-hand side is computed, the system of equations is solved using a matrix-free approach. We use GMRES for this purpose, with the system matrix-vector product following the same principle as discussed in the previous chapters.
The two contributions to the system matrix, the DG bilinear form and the mass matrix, are again applied locally to each element, and the result is transformed to the global index space. The DG bilinear form needs to be scaled by the time step size and the diffusion coefficient K when applied to the solution vector.

6.1.4 Fully Implicit Scheme

When computing the solution of the Allen-Cahn equation discretization derived in (6.9), the residual of the equation is minimized to find the solution.

$$R(u^{n+1}, u^n) = u^{n+1} - u^n - dtK\left(\Delta u^{n+1} - \frac{1}{\epsilon^2} f(u^{n+1})\right).$$

A nonlinear iterative solver, such as Newton's method, can be used to find a solution to the equation. Newton's method iteratively solves the equation:

$$R'(u^n)s^n = -R(u^n, u^n),$$

where R' is the Jacobian of the residual, and s^k is the update direction that gets added to the current solution:

$$u^{n+1} = u^n + s^n.$$

The computational effort can be reduced by using an inexact Newton method, where the Jacobian is approximated by a matrix-free approach.

6.1.5 Domain Decomposition

Combining the time-stepping schemes with domain decomposition methods can lead to more efficient solvers. In each time step, Schwarz methods are applied to find a solution for the next time step. The boundary conditions on the interior boundaries are enforced by the DG bilinear form and are the same as for the Poisson equation in previous chapters. The algorithm for the additive Schwarz method is shown in Algorithm 7.

Algorithm 7 Additive Schwarz Method for the Allen-Cahn Equation

Require: Maximum iterations max_iter, tolerance max_residual
for time_step = 1 to N_{timestep} **do**
  Initialize residual array residual ← 0
  for each subdomain DG solver *dg* **do**
    Compute and set right-hand side: *dg*.mesh.set_rhs(*dg*.compute_rhs())
    Save the previous solution: *dg*.mesh.set_u_old()
  end for
  Compute the initial residual: residual[0] ← compute_residual()
  for $i = 1$ to max_iter **do**
    for each subdomain DG solver *dg* **do**
      Update right-hand side: *dg*.mesh.set_rhs(*dg*.update_rhs())
      Solve local time step: *dg*.allen_cahn_time_step()
    end for
    Extend the local solutions to the global mesh: compose_global_solution()
    Compute the residual: residual[i] ← compute_residual()
    if residual[i] < max_residual **then**
      break
    end if
  end for
  Store iteration count: dd_iterations[time_step − 1] ← $i + 1$
end for

6.2 Numerical Results

For the numerical experiments, the derivation of the interface radius is taken from an internal report of the Jülich Supercomputing Centre and the parameter selection and initial conditions are based on the work of Speck [20] on spectral deferred correction methods for the Allen-Cahn equation. To evaluate the performance of the implemented solver for the Allen-Cahn equation, we consider the development of the radius of the phase separation over time. Setting $\mu = 0$ and inserting the initial condition u_0 and its derivatives into the Allen-Cahn equation, we obtain the ordinary differential equation:

$$\frac{\partial R(t)}{\partial t} = -\frac{K}{R(t)}.$$

This equation can be solved by separation of variables. Integrating both sides over time yields:

$$\int_{R(t_0)}^{R} R dR = -K \int_{t_0}^{t} dt, \tag{6.11}$$

$$\frac{1}{2}R^2 - \frac{1}{2}R_0^2 = -K(t - t_0), \tag{6.12}$$

$$R(t) = \sqrt{R_0^2 - 2K(t - t_0)}. \tag{6.13}$$

Using this analytical description of the radius, we compare the phase separation radius of the numerical solution to the analytical value.

The Allen-Cahn equation is solved using the IMEX scheme with the discontinuous Galerkin method and the additive Schwarz method. The domain is set to $\Omega = [-0.5, 0.5]^2$ and is triangulated with a uniform mesh of 16×16 elements, leading to a spatial resolution of $h = 0.0625$. We choose the parameters $K = 1$, $\varepsilon = 0.04$, the initial radius $R_0 = 0.25$, and the time resolution $dt = 0.001$.

6.2.1 Numerical Radius Determination of the Phase Separation

The initial solution described in (6.4) of the Allen-Cahn equation with parameters $K = 0.4$, $\varepsilon = 0.04$, and an initial radius of $R_0 = 0.25$ is shown in Figure 6.1. This bell-shaped curve represents the initial phase seperation with 0 and 1 as the two phases. With the chosen parameters, the radius of the phase separation decreases over time and the bell shape becomes sharper. The behavior of the solution over time is shown in Figure 6.2.

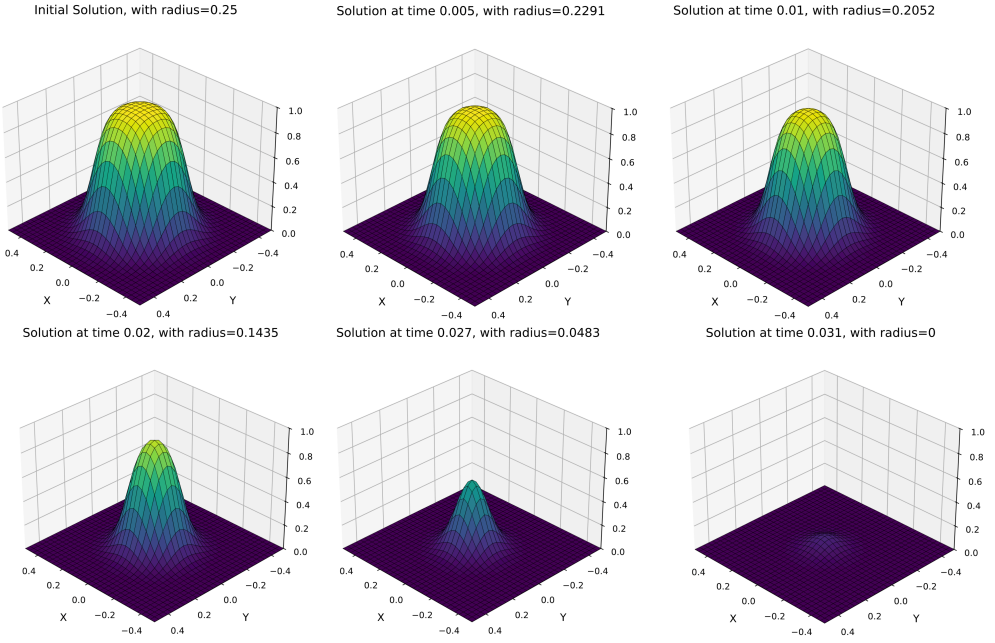


Figure 6.2: Solution of the Allen-Cahn equation over time.

To evaluate the numerical solution, we compare the analytically derived radius in Equation (6.13) to the radius of the numerical solution. The radius of the phase separation is between the two phases, meaning where the solution is 0.5. Because of the symmetric nature of the solution, the radius can easily be determined by looking at the solution 1D profile of $u(x, 0)$. The slice of the solution and the radius is shown in Figure 6.3. The radius is the x-coordinate where $u(x, 0) = 0.5$, which is computed by a simple linear approximation between the two nodes where the solution transitions from above u_{above} to below u_{below} at 0.5. The approximate radius is then given by:

$$r = x_{\text{above}} + \frac{0.5 - u_{\text{above}}}{u_{\text{below}} - u_{\text{above}}} \cdot (x_{\text{below}} - x_{\text{above}}).$$

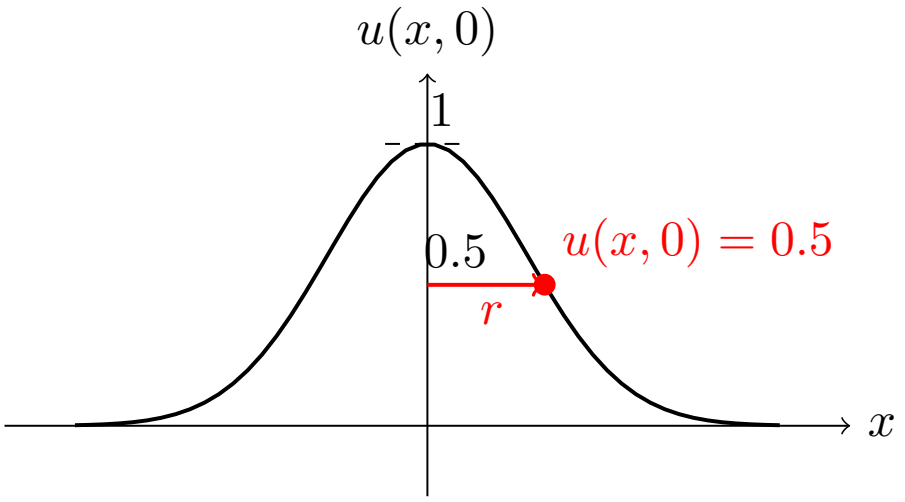


Figure 6.3: Radius of the phase separation over time.

6.2.2 Comparing the Numerical and Analytical Radii

With the selected parameters, we can compute when the radius of the phase separation reaches zero:

$$\begin{aligned} R(t) = \sqrt{R_0^2 - 2t} &= 0, \\ R_0^2 - 2t &= 0, \\ t = \frac{R_0^2}{2} &= 0.03125. \end{aligned}$$

In this experiment we solve the Allen-Cahn equation with the IMEX discretisation. We compare the performance of the global DG solver to the additive Schwarz method with four subdomains and local DG solvers. The results are shown in Figure 6.4 and compared to the analytical solution. In the first half of the simulation, the radius in the numerical solutions follow the analytical values closely. Until at around 0.015 seconds the numerical radii deviate from the analytical values, because the numerical error propagates over time. This leads to a difference of roughly 0.003 seconds in time when the radius reaches zero. The additive Schwarz method performs slightly better than the global DG solver, but the difference is negligible.

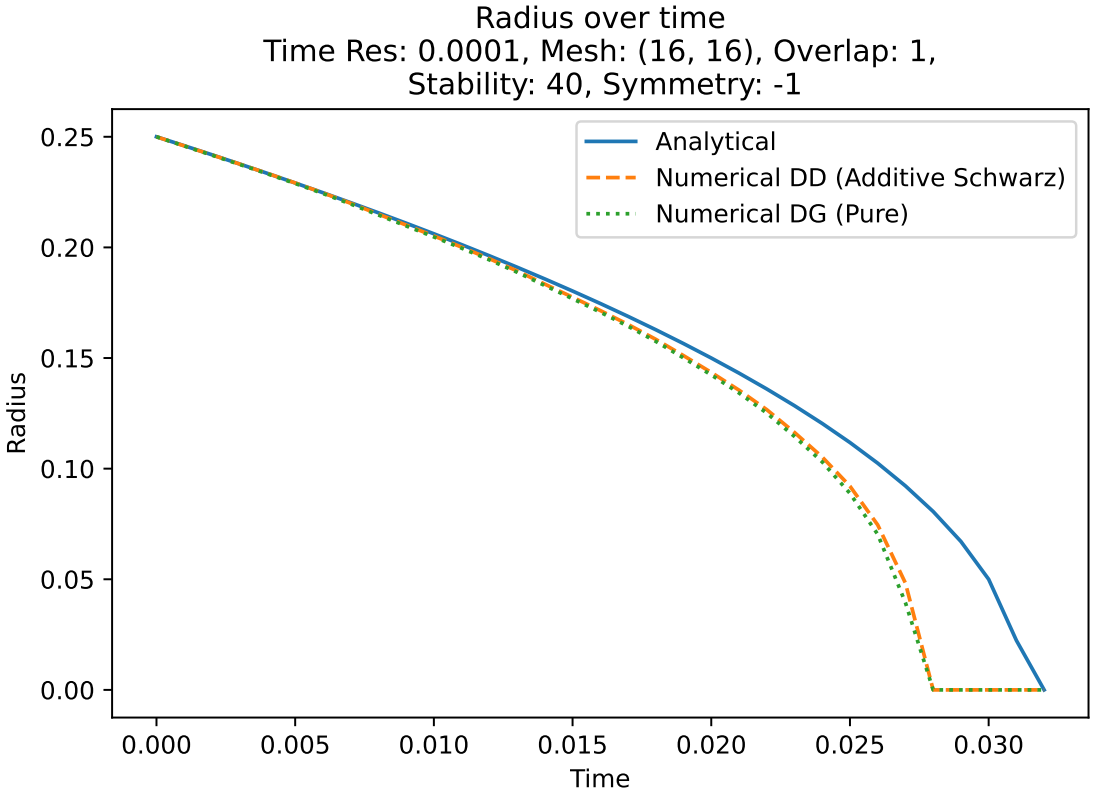


Figure 6.4: Analytically and numerically determined radius of the phase separation over time.

7 Conclusion

In this thesis, we investigated the combination of domain decomposition approaches with discontinuous Galerkin methods for solving linear and nonlinear partial differential equations. The goal was to evaluate the performance of these combined methods and apply the optimized Schwarz method with an interface transmission condition derived from the DG method. The Poisson equation served as primary test case for validating and testing the methods, followed by the Allen-Cahn equation, which was succesfully solved using the combination of DG and domain decomposition methods.

Three different approaches have been developed for implementing Robin interface conditions in the DG method within a domain decomposition framework. Through multiple numerical experiments, the successful application of the additive Schwarz and optimized Schwarz method within a DG framework has been shown. Introducing Robin boundary conditions, convergence for overlapping Subdomains was improved and the optimized Schwarz method was shown to be more efficient than the additive Schwarz method. Furthermore, the interaction between the penalty parameter of the DG method and the interface conditions was investigated, showing that the choice of the penalty parameter significantly influences the convergence of the Schwarz methods.

For the Allen-Cahn equation, we applied an implicit-explicit scheme to handle the nonlinear term of the equation. The results showed good agreement with the analytical predictions and demonstrated the potential of combining DG and domain decomposition methods for solving nonlinear PDEs.

Despite the promising results of DG-based domain decomposition methods, some challenges remain, that need to be addressed. The interplay between the penalty parameter of the DG method and the interface conditions requires careful parameter tuning to ensure optimal convergence. The slow convergence of non-overlapping subdomains in the optimized Schwarz method is another challenge that remains to be improved.

Future work could focus on further investigating the influence of the penalty parameter and the interface conditions on the convergence of the Schwarz methods. Additionally, further work is needed to evaluate the full potential of DG-based domain decomposition methods for solving the Allen-Cahn equation and applying a fully implicit scheme to solve the Allen-Cahn equation. Integrating multigrid methods could also further improve the convergence of Schwarz methods.

List of Figures

3.1 Example of a structured mesh used for continuous finite elements with $N_E = 9$ elements. . . . 15
3.2 Example of a structured mesh used for discontinuous finite elements with $N_E = 9$ elements. . . . 15
3.3 The unit square reference element $\hat{E}$. . . . 19
3.4 Distribution of Lagrange nodes for different polynomial degrees k. . . . 23
3.5 Gauss Quadrature Points for the face quadrature (left) and element quadrature (right). . . . 26
3.6 Display of Ghost Element on the domain boundary. . . . 28

4.1 Domain decomposition of a 2D domain Ω into Ω_1 and Ω_2. . . . 36
4.2 Orientation of the normal vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ for overlapping and non-overlapping subdomains. . . . 38

5.1 Partitioning of the domain into subdomains. . . . 40
5.2 Convergence of non-overlapping optimized Schwarz method with $\alpha = 0.5$, $\beta = 1$, $\sigma = 40$ and SIPG solving the subproblems 45
5.3 Solution of the optimized Schwarz method at timestep 0 46
5.4 Solution of the optimized Schwarz method at timestep 100 46
5.5 Fully converged solution of the optimized Schwarz method at iteration 400 . 46
5.6 Convergence of the additive Schwarz method. . . . 47
5.7 Convergence of the optimized Schwarz method. . . . 47
5.8 Convergence of the OSM with varying penalty parameters using IIPG. . . . 48
5.9 Convergence of the OSM with different interface condition implementations. 49
5.10 Convergence of the OSM with varying penalty parameters and Robin boundary condition factors. . . . 50

6.1 Initial solution of the Allen-Cahn equation with $R_0 = 0.25$, $\varepsilon = 0.04$ and $K = 1$. . . . 52
6.2 Solution of the Allen-Cahn equation over time. . . . 57
6.3 Radius of the phase separation over time. . . . 58
6.4 Analytically and numerically determined radius of the phase separation over time. . . . 59

List of Tables

3.1 Gauss Quadrature Points and Weights for the Interval $[0, 1]$ 25

Bibliography

[1] H. Schwarz, "Über einige abbildungsaufgaben," *Ges. Math. Abh., Berlin*, vol. 2, pp. 65–82, 1890.

[2] P.-L. Lions *et al.*, "On the schwarz alternating method. i," in *First international symposium on domain decomposition methods for partial differential equations*, vol. 1, p. 42, Paris, France, 1988.

[3] A. Toselli and O. Widlund, *Domain decomposition methods-algorithms and theory*, vol. 34. Springer Science & Business Media, 2006.

[4] M. J. Gander, "Optimized schwarz methods," *SIAM Journal on Numerical Analysis*, vol. 44, no. 2, pp. 699–731, 2006.

[5] P. F. Antonietti and B. Ayuso, "Schwarz domain decomposition preconditioners for discontinuous galerkin approximationsof elliptic problems: non-overlapping case," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 41, no. 1, pp. 21–54, 2007.

[6] M. J. Gander and S. Hajian, "Analysis of schwarz methods for a hybridizable discontinuous galerkin discretization," *SIAM Journal on Numerical Analysis*, vol. 53, no. 1, pp. 573–597, 2015.

[7] P. Lu, A. Rupp, and G. Kanschat, "Two-level schwarz methods for hybridizable discontinuous galerkin methods," *Journal of Scientific Computing*, vol. 95, no. 1, p. 9, 2023.

[8] B. Rivière, *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[9] J. Bergh and J. Löfström, *Interpolation spaces: an introduction*, vol. 223. Springer Science & Business Media, 2012.

[10] R. A. Adams and J. J. Fournier, *Sobolev spaces*. Elsevier, 2003.

[11] M. G. Larson and F. Bengzon, *The finite element method: theory, implementation, and practice*, vol. 10. 2010.

[12] L. Babuška and M. Suri, "The optimal convergence rate of the p-version of the finite element method," *SIAM Journal on Numerical Analysis*, vol. 24, no. 4, pp. 750–776, 1987.

[13] P. D. Lax and A. N. Milgram, *IX. Parabolic Equations*, pp. 167–190. Princeton: Princeton University Press, 1955.

[14] V. DOLEJŠÍ, "Finite element methods: Implementations," tech. rep., Technical report, Faculty of Mathematics and Physics, Charles University Prague, 2011.

[15] P. C. Matthews, *Vector calculus*. Springer Science & Business Media, 2012.

[16] W. H. Press, *Numerical Recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[17] P. Solin, K. Segeth, and I. Dolezel, *Higher-order finite element methods*. Chapman and Hall/CRC, 2003.

[18] V. Dolean, P. Jolivet, and F. Nataf, *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. SIAM, 2015.

[19] X. Feng and Y. Li, "Analysis of symmetric interior penalty discontinuous galerkin methods for the allen–cahn equation and the mean curvature flow," *IMA Journal of Numerical Analysis*, vol. 35, no. 4, pp. 1622–1651, 2015.

[20] R. Speck, "pysdc-prototyping spectral deferred corrections," *arXiv preprint arXiv:1808.02731*, 2018.

[21] J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

A Appendix

A.1 Explanation of Code Structure

Attached to this thesis are two folders containing the modular python code used for solving the Poisson and Allen-Cahn equations. The Codes within these folders have the same structure, but are adapted to the problem at hand. The implementation consists of a python module folder and a main function file called `run_module.py`. The python module consists of the following classes:

- `Mesh`
- `SubMesh`
- `DG`
- `BasisFunction`
- `Quadrature`

In the main function `run_module.py`, the user can create instances of the classes and define and solve the problem.

A.1.1 Mesh

The class `Mesh` takes as input:

- `cell_shape`: tuple of the number of cells in each direction.
- `dof`: $(dof + 1)^2$ defines the number of degrees of freedom per element.
- `x_domain`: tuple of the domain in the x-direction.
- `y_domain`: tuple of the domain in the y-direction.

With this information it creates a triangulation of the given domain with the desired resolution. It contains information about element connectivity, face neighbours and boundary faces. As well as the nodal values of the numerical solution and the analytical solution.

A.1.2 SubMesh

the class `SubMesh` is a subclass of `Mesh` and is used to create a submesh of the original mesh, taking as input:

- `parent_mesh`: the original mesh.
- `submesh_idx`: the index of the submesh.
- `overlap`: the number of overlapping cells between submeshes.
- `number_of_submeshes`: the number of submeshes in each direction.

the submesh inherits the information from the parent class and extends it with additional details about neighbouring meshes, its position within the global domain and whether a boundary face is an exterior or interior boundary face.

A.1.3 DG

the class `DG` takes as input:

- `mesh`: the mesh or submesh object.
- `penalty`: the penalty parameter.
- `symmetry_parameter`: the symmetry parameter determining the IPG method.
- `superpenalisation`: if this value is set greater than 1, superpenalisation is applied.
- `linear_solver_tol`: the tolerance for the linear solver.
- `alpha`: the parameter for the Robin boundary conditions.
- `beta`: the parameter for the Robin boundary conditions.
- `boundary`: determines the approach of implementation of Robin boundary conditions. If set to 0 Dirichlet Boundary conditions are used.

The class solves the Poisson or Allen-Cahn equation using the discontinuous Galerkin method, with the set parameters, on the given mesh. Within the DG object instances of the classes `BasisFunction` and `Quadrature` are created, which are then used to used to calculate the local formulations of the DG method.

A.1.4 Domain Decomposition

The class unites all the classes and solves the Poisson or Allen-Cahn equation using the domain decomposition method. It takes all necessary information for the previous classes as input and creates the necessary instances to solve the problems with the selected parameters.