Master Thesis

Solution of Non-Homogeneous Dirichlet Problems with FEM

Francesco Züger

Institut für Mathematik

Written under the supervision of
Prof. Dr. Stefan Sauter
and
Dr. Alexander Veit

August 27, 2013
Acknowledgment

I would like to thank Prof. Dr. S. Sauter for giving me the opportunity to write a master thesis about this very interesting topic. He, together with Dr. A. Veit, helped me with very useful hints along the way. I would also like to thank C. Lehrenfeld (developer of NetGen/NGSolve) for the (very prompt and exhaustive) explanations about the many features of the FEM solver. Lots of thanks also to Simona Trefalt and Marco Bernasocchi, who transformed my poor English into a readable text.

I would like to thank Noam Arnold as well: we attended many classes together and we worked in the same field for our master theses, which resulted in many discussions about FEM and NetGen/NGsolve that were very helpful.

Lastly I would like to thank my family who has supported me until now without pressure: I really appreciate their trust!
Contents

1 Preface ................................................. 1
  1.1 Introduction and Goal Setting ............................. 1
  1.2 NetGen - NGSolve ..................................... 1
  1.3 Structure ............................................ 2

2 Mathematical Framework ........................................ 3
  2.1 Partial Differential Equation (PDE) ......................... 3
    2.1.1 Classification of Second Order PDE ....................... 4
    2.1.2 Boundary Conditions ................................ 4
    2.1.3 Solution of Partial Differential Equations .............. 5
  2.2 Sobolev Space ........................................... 6
  2.3 Variational Form ......................................... 8
  2.4 Discretization ........................................... 10
    2.4.1 Ritz-Galerkin Discretization ......................... 10
    2.4.2 Finite Element Space ................................ 11
  2.5 Convergence and Error Bounds ............................. 13

3 Non-Homogeneous Dirichlet Boundary Conditions .......... 17
  3.1 Solution Split .......................................... 18
    3.1.1 Convergence and Error Estimator ....................... 20
    3.1.2 Implementation ..................................... 23
    3.1.3 Implementation in NGSolve ............................ 23
    3.1.4 Model Problem ...................................... 25
  3.2 Approximation using Robin Boundary Conditions .......... 29
    3.2.1 Convergence and Error Estimate ....................... 30
    3.2.2 Implementation ..................................... 32
    3.2.3 Implementation in NGSolve ............................ 32
    3.2.4 Model Problem ...................................... 33
  3.3 Time Performance ......................................... 36
    3.3.1 CG Algorithm ....................................... 36
    3.3.2 Solution Split ...................................... 37
    3.3.3 Approximation with Robin Boundary Conditions ........ 38
    3.3.4 Numerical Results ................................... 38
CONTENTS

4  Model Problem 2 ................................................. 41
   4.1  Setting ................................................................ 41
       4.1.1  Periodic Functions ...................................... 41
       4.1.2  Coefficient functions ................................. 41
   4.2  Full Space Problem ........................................... 42
   4.3  Numerical Solution of the Unit Cell Problem .......... 43
   4.4  Dirichlet Problem on a Finite Domain .................. 44
   4.5  Implementation .................................................. 45
       4.5.1  Unit Cell Problem Using Matlab .................... 45
       4.5.2  Implementation Using NetGen/NGSolve ............ 46
   4.6  Numerical Results ............................................ 47
       4.6.1  Error Behavior ............................................. 47
       4.6.2  Time Performance ........................................ 52

5  Conclusion .......................................................... 55

Bibliography ......................................................... 57

Appendix ............................................................... 59
Chapter 1

Preface

1.1 Introduction and Goal Setting

Many physical phenomena (heat, electrodynamics, fluid dynamics, ...) can be represented using partial differential equations (PDEs). This is why many efforts have been made to find appropriate tools to solve or approximate a given PDE. Thanks to its flexibility, the finite element method (FEM) is nowadays one of the most commonly employed mathematical method to approximate the solution of various problems.

In the current work we focus on the resolution of elliptic PDEs with non-homogeneous Dirichlet boundary conditions, also referred to as non-homogeneous Dirichlet problems, which indicate a problem where the searched solution has to coincide with a given function $g$ on the boundary of the domain. The “standard” FEM only allows to solve Dirichlet problems with homogeneous boundary conditions, and some approaches were developed to also handle non-homogeneous cases.

The goal of this work is to present possible ways to deal with non-homogeneous Dirichlet problems using FEM and to give the appropriate mathematical background. Central is the study of the convergence and error bounds for these methods. Moreover we performed some numerical tests to investigate the convergence rate and the time performance of the possible algorithms, comparing these results with the statements from the theoretical part.

1.2 NetGen - NGSolve

For the computational part of the study we use NetGen and NGSolve. These are two open source programs created by J. Schöberl and his team which work in a coupled way to solve PDEs with the FEM.

NetGen, the first of the two programs employed, is a mesh generator. For a given 2D or 3D geometry it generates the regular triangulation needed to implement a finite element space and it can refine a given mesh using a bisection algorithm. Moreover, it is possible to impose some a-priori refinement in particular areas of the domain along some edges or close to a particular point, and also implement adaptive refinement algorithms.

NGSolve, the second of the programs used, is the coupled FEM-solver. It can solve various types of PDEs and it is possible to implement additional features by modifying the source code. It is for example possible to implement the resolution of time dependent problems.
Both programs (and source-codes as well) can be downloaded from [Sch13a, Sch13b]. On the same websites it is also possible to consult a forum and a short “wiki”.

The official manual can be found in the documentation included in the installation files (ng4.pdf). A more detailed manual, written by Noam Arnold can be found in [Arn12]: in particular the author explains how to modify the source code to implement non-stationary problems.

1.3 Structure

The second chapter of this thesis is devoted to the introduction of the mathematical framework. The fundamentals of the FEM are summarized and the most useful error bounds are derived.

The third chapter covers the main topic of this thesis: possible methods to solve non-homogeneous Dirichlet problems are introduced and their convergence is analyzed. To test the reliability of the error bounds and the time performance of the algorithm some test for a first model problem are presented.

The fourth chapter is dedicated to the second model problem. Using this slightly more complex problem we verify the behavior of the methods to solve non-homogeneous Dirichlet problems.

The last chapter is a résumé of the most important results and a short outlook on possible further research in the area is given.
Chapter 2

Mathematical Framework

In this chapter the most important state-of-the-art results from the theory of the FEM are summarized. In particular practical error bounds that will be tested in empirical applications in the next chapter are derived. For more details about the FEM, Braess [Bra07], Brenner and Scott [BS94] as well as other authors, provide self contained expositions.

2.1 Partial Differential Equation (PDE)

A differential equation is an equation whose unknown is a function depending on one or more variables. We speak of PDE when the function depends on many variables and the problem involves partial derivatives.

The unknown function is noted by \( u \) and depends on the spatial variables \( x_1, x_2, \ldots, x_d \). In case of a non-stationary problem, a time variable \( t \) is added. Within this master thesis we will however only focus on stationary problems. For a shorter notation the spatial variables are collected in the vector \( x \in \mathbb{R}^d \).

We denote the partial derivate by
\[
\partial_i u(x) := \partial_{x_i} u(x) := \frac{\partial u(x)}{\partial x_i} \quad 1 \leq i \leq d.
\]

We then define a general PDE by using a functional \( F \) as:
\[
F(x_1, \ldots, x_d, u, \partial_1 u, \ldots, \partial_d u, \partial_1 \partial_1 u, \ldots, \partial_1 \partial_d u, \ldots) = f.
\]

The order of the highest derivative defines the order of the equation. Moreover, if \( F \) is a linear function of \( u \) and its derivatives, the PDE is called linear.

Example 2.1.1:
Using the Laplace Operator \( \Delta \) given by
\[
\Delta u(x) := \sum_{i=1}^{d} \frac{\partial^2 u(x)}{\partial x_i^2}
\]
we can define the Potential or Laplace Equation as:
\[
\Delta u = 0 \quad \text{in } \Omega \subset \mathbb{R}^d.
\]

This is a second order, linear PDE.
2.1.1 Classification of Second Order PDE

The general, linear second order PDE can be described as follows:

\[-\text{div}(A \text{ grad } u) + b \cdot \text{ grad } u + cu = f \quad \text{in } \Omega \subset \mathbb{R}^d,\]

(2.1)

where

\[
\text{grad } u(x) = \nabla u(x) := \left(\frac{\partial u(x)}{\partial x_i}\right)_{i=1}^d, \quad \text{div } v(x) := \sum_{i=1}^d \frac{\partial v_i(x)}{\partial x_i}.
\]

The coefficient \(A : \Omega \to \mathbb{R}^{d \times d}\) is a matrix of real-valued functions, \(b : \Omega \to \mathbb{R}^d\) is a vector of real-valued functions and \(c : \Omega \to \mathbb{R}\) is a real valued function. Since for \(u \in C^2 : \partial_i \partial_j u = \partial_j \partial_i u\), the matrix \(A\) can be chosen w.l.o.g. to be symmetric.

The general second order PDE (2.1) can also be described as

\[- \nabla \cdot (A \nabla u) + b \cdot \nabla u + cu = f\]

(2.2)

Definition 2.1.2:
The equation (2.1) is said to be:

1. **elliptic** in \(x \in \mathbb{R}^d\), if \(A(x)\) is positive definite or negative definite.
2. **hyperbolic** in \(x \in \mathbb{R}^d\), if \(d-1\) eigenvalue of \(A(x)\) are of the same sign and one is of the opposite sign.
3. **parabolic** in \(x \in \mathbb{R}^d\), if \(d-1\) eigenvalue of \(A(x)\) are of the same sign, one equals null and \(\text{Rang}(A, b) = d\).
4. **elliptic** in \(\Omega \subset \mathbb{R}^d\), if \(A(x)\) for all \(x \in \Omega\) is positive definite or if for all \(x \in \Omega\) is negative definite.

Such a classification is needed, since for every type of second order PDE, different approaches have to be used to find the solution \(u\). In the current thesis we only focus on the fourth case, i.e. elliptic equations.

Remark 2.1.3:
Not every PDE can be classified via Definition 2.1.2.

Example 2.1.4:
For the Laplace equation of Example 2.1.1, \(A\) is the identity matrix (always positive definite) and thus the Laplace equation is elliptic.

2.1.2 Boundary Conditions

Definition 2.1.5:
A problem is said **well-posed**, if exactly one solution exists and it continuously depends on the coefficients \(A, b, c\) and on the boundary condition [Bra07, p. 8].
Typically, PDEs, if not provided with additional information, are not well-posed because the solution is not unique. Consider again the Laplace equation (which is linear): if $u_1$ is a possible solution, then every scalar multiplication $ku_1$, $k \in \mathbb{R}$ is also a solution ($\Delta(ku_1) = k\Delta(u_1) = 0$). Moreover if $u_2$ is another possible solution, then every linear combination $ku_1 + ju_2$ for $k, j \in \mathbb{R}$ is a further solution of the same problem.

In order to have a unique solution we add further informations to the problem. In case of an ordinary differential equation (ODE) we impose some initial conditions. Similarly we add boundary condition for PDEs such that the problem has a unique solution. Since many PDEs arise from physical problems, where the behaviour of the unknown function can be imposed or measured on the boundary, the corresponding PDE will be accordingly equipped with some boundary conditions.

The most commonly used boundary conditions, defined over the boundary $\partial \Omega$, are the following ones:

- **Dirichlet (or Essential) Boundary Conditions**, defined as
  $$ u = g \quad \text{on } \partial \Omega. $$
  In particular, if $g = 0$ we speak of homogeneous boundary conditions.

- **Neumann (or Natural) Boundary Conditions**, defined as
  $$ \frac{\partial u}{\partial n} = g \quad \text{on } \partial \Omega, $$
  where $n$ is the outward pointing unit normal vector on $\partial \Omega$.

- **Robin Boundary Conditions**, defined as
  $$ \gamma u + \alpha \frac{\partial u}{\partial n} = g \quad \text{on } \partial \Omega. $$
  Even though in the literature this particular type of boundary conditions is not widely presented, it allows us to approximate a non-homogeneous Dirichlet boundary conditions. This particular case will be discussed in the next chapter.

If the boundary can be divided in more disjoint parts $\partial \Omega = \partial \Omega_1 \cup \partial \Omega_2 \cup \cdots$, different boundary conditions can be imposed on each of the parts $\partial \Omega_i$. In this case, we speak of **mixed boundary conditions**.

**Definition 2.1.6:**
A function, which satisfies a PDE as well as the boundary conditions is said **classical solution**.

### 2.1.3 Solution of Partial Differential Equations

Even if PDEs, with suitable boundary conditions are well-defined, the solution of these problems is far from being obvious. In some case it may also be proven that it is in fact impossible to find an exact solution. Since many PDEs correspond to problems from the “real world” that need to be solved, many efforts were made to find possible methods to approximate their solution and to prove that the approximate solution converges to the exact one.
In the 20th century two classes of approximation methods became popular: the finite difference method and the finite element method (FEM). The former replaces the derivatives with a difference quotient, for example for one dimensional problems this corresponds to \( u'(x) \approx \frac{u(x+h/2) - u(x-h/2)}{h} \). Using this approximation one can derive a linear system of equations and easily find an approximate solution ([Bra07, p. 15] and [SF08, p. 16]). Unfortunately it is not possible to derive sharp existence and uniqueness theorems and the method is restricted to polygonal domains with only right angles (\( \frac{\pi}{2} \) or \( \frac{3}{2} \pi \)).

In this master thesis we focus on the FEM, which is much more flexible allowing us to handle very complex problems. This method is based on the variational form: consider for example the homogeneous Poisson problem \(-\Delta u = f\) in \( \Omega \), \( u = 0 \) on \( \partial \Omega \). Multiplying the first equation by some suitable function \( v \) vanishing at the boundary, taking the integral and using the Gaussian Theorem, we derive the variational form: find an admissible \( u \) (thus vanishing at the boundary) such that

\[
\int_{\Omega} (\nabla u, \nabla v) = \int_{\Omega} fv \quad \text{for all test functions } v.
\]

Applying an ansatz \( u \approx \sum_j q_j \varphi_j \), where \( \varphi_j \) are basis functions for a finite dimensional subspace of the admissible function space, the Galerkin discretization leads to a linear system of equations

\[
Mq = r
\]

where \( (m)_{j,i} = \int_{\Omega} \varphi_j \varphi_i \) and \( (r)_i = \int_\Omega f \varphi_i \). The key point of the FEM is the choice of the basis functions: each basis function \( \varphi_j \) (also referred to as “hat”-functions) is a piecewise polynomial vanishing outside some “finite elements” (triangles or quadrilateral in 2D, polyhedral in 3D). With this choice the system matrix becomes sparse and iterative solvers can be applied efficiently.

### 2.2 Sobolev Space

In this section some notions and results from functional analysis that allow us to investigate the mathematical background of the FEM are introduced. Assume \( \Omega \subset \mathbb{R}^d (d \geq 2) \) is an open, bounded domain. Let us introduce the notation for derivatives with multi-index; for \( \alpha \in \mathbb{N}_0^d \) with \( |\alpha| = \alpha_1 + \cdots + \alpha_d \):

\[
D^\alpha \varphi := \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_d^{\alpha_d}} \varphi.
\]

**Definition 2.2.1:**
Assume \( \phi, \psi \in L^2(\Omega) \) and \( \alpha \) is a multi-index. Then \( \psi \) is the \((\alpha-)\) weak derivative of \( \phi \) (\( \psi = D^\alpha \phi \)), if

\[
\int_{\Omega} \psi \phi = (-1)^{|\alpha|} \int_{\Omega} D^\alpha v \phi \quad \forall v \in C_0^\infty(\Omega).
\]

**Remark 2.2.2:**
- If the \( \alpha \)-weak derivative exists, then it is unique.
- If \( u \in C^{[\alpha]} \), then the weak and the classical derivative coincide, which is why the same symbol \( D^\alpha \) is used.
2.2. SOBOLEV SPACE

- With the notion of weak derivatives we can define the derivative of some functions in \( L^2 \) whose classical (point-wise) derivatives do not exist.

**Example 2.2.3:**
Let \( \Omega = [-1,1] \). The weak derivative of the function \( \phi = 1 - |x| \) is given by

\[
D^1 \phi(x) = \begin{cases} 
1 & -1 \leq x < 0, \\
-1 & 0 < x \leq 1.
\end{cases}
\]

The proof can be found in [BS94, p. 26].

**Definition 2.2.4:**
For \( k \in \mathbb{N} \) and \( p \in [1, \infty) \), we define the **Sobolev Space** \( W^{k,p} \) as

\[
W^{k,p}(\Omega) := \{ \varphi \in L^p(\Omega) \mid \forall |\alpha| \leq k : D^\alpha \varphi \in L^p(\Omega) \}.
\]

These spaces are equipped with the following norm:

\[
\|\varphi\|_{W^{k,p}(\Omega)} := \left( \sum_{|\alpha| \leq k} \|D^\alpha \varphi\|_{L^p(\Omega)}^p \right)^{\frac{1}{p}}
\]

and seminorm:

\[
|\varphi|_{W^{k,p}(\Omega)} := \left( \sum_{|\alpha| = k} \|D^\alpha \varphi\|_{L^p(\Omega)}^p \right)^{\frac{1}{p}}.
\]

The standard setting for our application will be \( p = 2 \) and for simplicity we write short \( H^k(\Omega) := W^{k,2}(\Omega) \). If the domain is clear from the context we skip it in the notation of our function spaces. We have \( H^0 = L^2 \) and the associated norms coincide. The \( H^1 \)-norm can be written according to

\[
\|\phi\|_{H^1}^2 = \|\phi\|_{L^2}^2 + |\phi|_{H^1}^2 = \int \phi^2 + \int (\nabla \phi)^2.
\]

**Lemma 2.2.5:**
- The Sobolev space \( W^{k,p} \), with the norm \( \| \cdot \|_{W^{k,p}} \) is a complete normed vector space and thus a Banach space.
- \( H^k \), provided with the scalar product \( (\phi, \psi)_{H^k} := \sum_{|\alpha| \leq k} (D^\alpha \phi, D^\alpha \psi)_{L^2(\Omega)} \) is a Hilbert space.

**Proof:** See [BS94, p.28].

Before discussing the variational form, we employ the trace theorem, which will enable us to define essential boundary conditions.

**Definition 2.2.6:**
The domain \( \Omega \) has a **Lipschitz-boundary** (or \( \Omega \) is a **Lipschitz-domain**), if for \( N \in \mathbb{N} \) there exists some open sets \( U_1, \ldots, U_N \subset \mathbb{R}^d \) such that

1. \( \partial \Omega \subset \bigcup_{i=1}^N U_i \),
(2) \( \partial \Omega \cap U_i \) can be described as graph of a Lipschitz-continuous function for every \( 1 \leq i \leq N \).

**Trace Theorem 2.2.7:**
Let \( \Omega \) be a Lipschitz-domain, \( k \in \mathbb{N}, l \in \{0, 1, \cdots, k - 1\} \) and \( n \) be the outward pointing normal vector on \( \partial \Omega \). Then there exists a linear continuous map \( \gamma_i : W^{k,p}(\Omega) \to L^p(\partial \Omega) \) with the following property

\[
\gamma_i(\varphi) = \frac{\partial}{\partial n} \varphi \quad \forall \varphi \in C^k(\Omega). \tag{2.3}
\]

**Proof:** See [Alt12, S. 265].

From the trace theorem we can derive a very useful definition when dealing with homogeneous Dirichlet boundary conditions.

**Definition 2.2.8:**
Assume \( \gamma_i \) as in Theorem 2.2.7, then we define the Sobolev space with function vanishing at the boundary as

\[
W_0^{k,p}(\Omega) := \{ \varphi \in W^{k,p}(\Omega) \mid 0 \leq l \leq k - 1 : \gamma_l(\varphi) = 0 \}.
\]

In particular, for \( k = 1 \) and \( p = 2 \) it follows that:

\[
W_0^{1,2} =: H_0^1 = \{ u \in H^1 : u|_{\partial \Omega} = 0 \}.
\]

### 2.3 Variational Form

For the formulation of sharp existence and uniqueness theorems as well as to derive the numerical discretization we have to convert the problem from the classical formulation to the variational form. After analyzing the well-posedness, we will discuss under which assumptions the strong and the variational formulation coincide.

**Definition 2.3.1:**
Let \( H \) be a Hilbert space (with norm \( \| \cdot \| \)). A bilinear form \( a : H \times H \to \mathbb{R} \) is

- **bounded (or continuous)** if there exists a constant \( C > 0 \) such that:

  \[
  |a(u,v)| \leq C\|u\|\|v\| \quad \forall u, v \in H,
  \]

- **symmetric** if:

  \[
  a(u,v) = a(v,u) \quad \forall u, v \in H.
  \]

A bounded bilinear form \( a \) is **coercive (or elliptic)** if there exists an \( \alpha > 0 \) such that:

\[
a(v,v) \geq \alpha \|v\|^2 \quad \forall v \in H.
\]

**Remark 2.3.2:**
For a bilinear form \( a(\cdot, \cdot) \) we can define the energy norm as \( \|v\|_a = \sqrt{a(v,v)} \).

With these assumptions at hand, we can state a key result for the FEM that ensures existence and uniqueness of the solution.
Lax-Milgram Theorem 2.3.3:
Given a Hilbert space \( H \), a bounded, coercive bilinear form \( a : H \times H \to \mathbb{R} \) and a linear form \( l : H \to \mathbb{R} \), then the functional
\[
I(u) := \frac{1}{2} a(u, u) - l(u)
\]
has a unique minimum \( u^* \in H \). This is the unique solution of the variational problem
\[
a(u^*, v) = l(v) \quad \forall v \in H.
\] (2.4)

**Proof:** See [Bra07, p. 37].

The variational formulation is related to a minimization problem for the functional. Braess [Bra07, Charakterisierungssatz p.34] shows that if and only if \( u \) solves (2.4) then \( u \) is the unique minimum of \( I(v) \) and thus the two formulations are equivalent.

**Definition 2.3.4:**
Consider an elliptic PDE given by (2.1) with smooth coefficients \( A, b \) and \( c \) such that
\[
0 < \lambda := \inf_{x \in \Omega} \inf_{z \in \mathbb{R}^d \setminus \{0\}} \frac{z^T A(x) z}{z^T z} \leq \sup_{x \in \Omega} \sup_{z \in \mathbb{R}^d \setminus \{0\}} \frac{z^T A(x) z}{z^T z} =: \Lambda < \infty
\]
holds for some \( \lambda, \Lambda \). Then

- \( u \in H^1_0(\Omega) \) is a **weak solution** of (2.1) with homogeneous Dirichlet boundary conditions if:
\[
\int_{\Omega} (A \nabla u) \cdot \nabla v + (b \cdot \nabla u) v + cuv = \int_{\Omega} fv \quad \forall v \in H^1_0(\Omega),
\] (2.5)

- \( u \in H^1(\Omega) \) is a **weak solution** of (2.1) with Neumann boundary conditions if:
\[
\int_{\Omega} (A \nabla u) \cdot \nabla v + (b \cdot \nabla u) v + cuv = \int_{\Omega} fv + \int_{\partial \Omega} gv \quad \forall v \in H^1(\Omega).
\] (2.6)

By setting \( a(u, v) = \int_{\Omega} (A \nabla u) \cdot \nabla v + (b \cdot \nabla u) v + cuv \) and \( l(v) = \int_{\Omega} fv \), respectively \( l(v) = \int_{\Omega} fv + \int_{\partial \Omega} gv \) we see that (2.5) and (2.6) are of the form (2.4).

**Remark 2.3.5:**
The Dirichlet boundary conditions are also called “essential” because they are “build” directly in the space \( H^1(\Omega) \); on the other hand Neumann boundary conditions are called “natural” because the solution of the corresponding variational form satisfies the condition in a weak form (2.6).

**Lemma 2.3.6:**
Consider an elliptic PDE given by (2.1) with homogeneous Dirichlet Boundary conditions and the corresponding weak formulation from Definition 2.3.4. Assume that \( A \in L^\infty(\Omega, \mathbb{R}^{d \times d}) \) and symmetric, \( b = 0 \) and \( c \in L^\infty(\Omega) \) with \( c \geq 0 \). Then:

1. every classical solution is also a weak solution. Every weak solution which lies in \( C^2(\Omega) \cap C^0(\bar{\Omega}) \) is also a classical solution.
(2) For every \( f \in L^2(\Omega) \) there exists a unique weak solution in \( H^1_0 \).

Proof: (1) see [Bra07, Minimaleigenschaft p. 35].

(2) After proving that \( a(\cdot,\cdot) \) is a bounded, coercive bilinear form we can apply the Lax-Milgram Theorem (see [Bra07, Existenzsatz p. 39]). \( \square \)

For PDEs with Neumann boundary conditions (or mixed boundary conditions) we can derive existence and uniqueness in a similar way (see [Bra07, p. 43]).

2.4 Discretization

For a given Hilbert space \( H \), bounded, coercive bilinear form \( a : H \times H \to \mathbb{R} \) and a linear form \( l : H \to \mathbb{R} \), let us consider the problem in the variational form: find \( u \in H \) such that \( a(u,v) = l(v) \) for every \( v \in H \), where \( \Omega \) is a polyhedral domain with Lipschitz-boundary \( \partial \Omega \). Note that for the FEM it is standard to use \( H^1(\Omega) \) (or a proper subspace) as solution space and, for shorter notation, the domain \( \Omega \) is sometimes omitted.

2.4.1 Ritz-Galerkin Discretization

To approximate the solution \( u \) we construct a finite dimensional space \( S^h \subseteq H^1 \) (respectively \( S^h_0 \subseteq H^1_0 \) for homogeneous Dirichlet boundary conditions) such that \( S^h \) has a basis denoted by \( (\varphi_j)_{j=1}^N \) so that \( N = \dim(S^h) \). To approximate the solution \( u \in H^1 \) we employ the ansatz:

\[
    u \approx u^h = \sum_{j=1}^N q_j \varphi_j \in S^h. \tag{2.7}
\]

Using (2.7) we can rewrite the variational form as:

\[
    a(u^h,v^h) = l(v^h) \quad \forall v^h \in S^h,
\]

since \( (\varphi_j)_{j=1}^N \) is a basis for \( S^h \). This is equivalent to

\[
    a(u^h,\varphi_i) = l(\varphi_i) \quad \forall i = 1,2,\cdots,N. \tag{2.8}
\]

\( a(\cdot,\cdot) \) is a bilinear form, which means that employing the ansatz (2.7), we get

\[
    \sum_{j=1}^N q_j a(\varphi_j,\varphi_i) = l(\varphi_i) \quad \forall i = 1,2,\cdots,N.
\]

This is a linear system of equations which we write in the compact formulation

\[
    Mq = r \tag{2.9}
\]

where \( (m)_{j,i} = a(\varphi_j,\varphi_i) \) and \( (r)_i = l(\varphi_i) \).
2.4. Discretization

2.4.2 Finite Element Space

At this point the central issue is to find an appropriate finite element space $S_h$ as the span of some basis functions $\varphi_j$. In this section finite element spaces on polygonal domain in 2D are introduced. For higher spatial dimension the approach is conceptually similar and can be found in the literature (for example see [BS94, Chapter 5]).

Definition 2.4.1: (1) A subdivision of a domain $\Omega$ is a finite collection of open sets $\{\tau_i\}$, such that:

(i) $\tau_i \cap \tau_j = \emptyset$ if $i \neq j$,

(ii) $\bigcup \tau_i = \bar{\Omega}$.

(2) A subdivision $T = \{\tau_i : 1 \leq i \leq m_T\}$ (where $\tau_i$ are triangles) is a triangulation if no vertex of any triangle lies in the interior of some edges of another triangle (see Figure 2.1 (a)-(b)).

Moreover, if we define for every triangle $\tau \in T$: $h_\tau := \text{sup}_{x, y \in \tau} \|x - y\|$ and $\rho_\tau$ as the diameter of the largest circle fitting in $\tau$ then

(3) $T$ is regular triangulation if there exists a constant $c_T$ such that

$$\frac{h_\tau}{\rho_\tau} \leq c_T \quad \forall \tau \in T. \quad (2.10)$$

For a regular triangulation, we define the maximal mesh width $h$ (also referred to as mesh size) as follows:

$$h := \max_{\tau \in T} h_\tau.$$

Remark 2.4.2: • The condition defined in (2.10) is equivalent to a minimal angle condition (see Figure 2.1 (c)).

• For the subdivision for the FEM not only triangles, but also quadrilateral element can be used [BS94, p. 82]. In higher dimensions higher dimensional simplices are employed to define a “triangulation”.

After subdividing the domain $\Omega$ by a regular triangulation (mesh) we now define the finite element spaces $S^h \subset H^1$ and $S^h_0 \subset H^1_0$.

Definition 2.4.3:

The finite element space of degree $k$ is given by

$$S^h := \left\{ v \in C^0(\bar{\Omega}) | \forall \tau \in T : v|_\tau \in P_k \right\}, \quad \text{respectively} \quad (2.11)$$

$$S^h_0 := \left\{ v \in C^0(\bar{\Omega}) | \forall \tau \in T : v|_\tau \in P_k \right\} \cap H^1_0(\Omega), \quad (2.12)$$

where $P_k$ is the set of all polynomial in two variables of degree $\leq k$.

For simplicity let us consider the linear case ($k = 1$). Let us denote by $x_l$ for $l = 1, \cdots, N_T$ the vertices of the triangles of the triangulation. As we will show, in the linear case, the vertices coincide with the nodal points needed to define the finite element space. The vertices (or in
CHAPTER 2. MATHEMATICAL FRAMEWORK

(a) Admissible triangulation  (b) Non-admissible triangulation  (c) Regular triangulation

Figure 2.1: Examples of triangulations. (a) shows an admissible triangulation while (b) shows a triangulation with a “hanging node”. (c) illustrates the definition of shape-regularity for a triangle.

general the nodal points) not lying on the boundary are numbered from 1 to $\overline{N}_T$ (internal nodes), while the vertices from $\overline{N}_T + 1$ to $N_T$ lie on the boundary.

The restriction of every function $v \in S^h$ in a triangle $\tau$ is an affine function. This function on $\tau$ is uniquely determined by three linearly independent points, thus every polynomial $p \in \mathbb{P}_1$ is uniquely determined by the value at the three vertices. Consider now two triangles $\tau_1, \tau_2$ having a common edge $E$; for $p_1, p_2 \in \mathbb{P}_1$ we define

$$\varphi := \begin{cases} p_1 & \text{in } \tau_1, \\ p_2 & \text{in } \tau_2. \end{cases}$$

Since the restriction of $p_1$ to an edge $E$ is a linear function in one variable, then $\varphi$ is continuous if and only if the value of $p_1$ and $p_2$ coincide on the vertices of $E$.

We can conclude that every function $v \in S^h$ can be uniquely determined by the values at the vertices $x_l$ and thus

$$\dim S^h = N_T,$$

where $N_T$ is also referred to as number of degrees of freedom. For homogeneous Dirichlet boundary condition the value at vertices lying on the boundary has to be zero, thus $\dim S^h_0 = N_T$.

The last issue is to find an appropriate basis for the finite element space. For some $\tau \in \mathcal{T}$ and $i = 1, 2, 3$ we define the following function:

$$\lambda_{\tau,i}(x) := \frac{\det (x_{i+1}^\tau - x, x_{i-1}^\tau - x_{i+1}^\tau)}{\det (x_{i+1}^\tau - x_{i}^\tau, x_{i-1}^\tau - x_{i+1}^\tau)}$$
where \( x_i^\tau \) are the coordinates of the \( i \)-th vertex of \( \tau \) (vertex numbered counter clockwise) and the indices are to be interpreted in modulo 3. \( \lambda_{\tau,i} \) has the property:

\[
\lambda_{\tau,i}(x_j^\tau) = \begin{cases} 
1 & i = j, \\
0 & i \neq j.
\end{cases}
\]

For any node \( x_l \) we denote by \( \omega_l \) the union of the elements, i.e., triangles having \( x_l \) as vertex:

\[
\omega_l := \bigcup_{\tau \in \mathcal{T}, x_l \in \tau} \tau.
\]

We now define the basis functions for the finite element space as:

\[
\varphi_l(x) := \begin{cases} 
0 & x \notin \omega_l, \\
\lambda_{\tau,j}(x) & x \in \tau \subset \omega_l \text{ and } x_l = x_j^\tau.
\end{cases}
\]

This function is hat-shaped (continuous, piece-wise linear and vanishing outside \( \omega_l \)), it has value 1 at \( x_l \) and 0 for all other nodes. From the construction of \( \varphi_l \) we conclude that the functions \( \varphi_l \) is a basis for \( S^h \):

\[
S^h = \text{span}\{ \varphi_l : 1 \leq l \leq N_{\mathcal{T}} \}.
\]

**Remark 2.4.4:**

Consider an elliptic PDE with homogeneous Dirichlet boundary conditions or with Neumann boundary conditions and the corresponding linear system of equations \( Mq = r \) resulting from (2.8), (2.9) using the finite element spaces described above.

1. **The system matrix** \( M \) is symmetric and positive definite (see [Bra07, p. 51]).

2. **The unique solution of the basis representation** (2.7) corresponds to the solution of the variational form by means.

3. **The system matrix** \( M \) is sparse: the entry \( m_{j,i} = a(\varphi_j, \varphi_i) \) is non-zero only when the mesh points \( x_i \) and \( x_j \) belongs to the same edge of the mesh.

**Remark 2.4.5:**

For homogeneous Dirichlet boundary conditions, the value of the solution at the vertices lying on the boundary must equal to zero, thus \( q_j = 0 \) for all \( \mathcal{N}_\mathcal{T} < j \leq N_\mathcal{T} \). Therefore:

\[
S^h = \{ v \in C^0(\Omega) : \forall \tau \in \mathcal{T} : v|_{\tau} \in \mathbb{P}_k \text{ and } v(x_l) = 0 \text{ for all nodes } x_l \text{ on the boundary} \}.
\]

When increasing the degree of the polynomials it is obvious that the values at the triangle vertices no longer allow to uniquely determine the polynomials \( p_i \in \mathbb{P}_k \) (see Table 2.1): it follows that more constraints are needed. Using Lagrange elements we define a unisolvent set of points on the edges between vertices or inside every triangle that allow us to uniquely identify each polynomial [BS94, p. 70]. More complex elements involving derivatives at the vertices of triangles are possible as well (Hermite elements, Argyris elements; see [BS94, p. 73]).

### 2.5 Convergence and Error Bounds

In this section, we will prove that the approximate solution \( u^h \) given by the FEM converges to the weak solution \( u \). For this, it is important to derive some bounds for the error \( \|u - u^h\| \).
### Table 2.1: Dimension of \( P_k \) in two dimensions (from [BS94, p. 70]).

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \dim P_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( k )</td>
<td>( \frac{1}{2}(k+1)(k+2) )</td>
</tr>
</tbody>
</table>

Céa’s Lemma 2.5.1:

Let \( H \) be a Hilbert space and \( S^h \subset H \) a finite dimensional subspace. Let a bounded, coercive bilinear form \( a : H \times H \to \mathbb{R} \) and a linear form \( l : H \to \mathbb{R} \) be given. Let \( u \) be the weak solution of problem (2.4) and \( u^h \) be the corresponding weak solution in the finite dimensional subspace \( S^h \). Then

\[
\| u - u^h \|_H \leq \frac{C}{\alpha} \inf_{\nu^h \in S^h} \| u - \nu^h \|_H,
\]

where \( C \) and \( \alpha \) are the constants given in Definition 2.3.1.

**Proof:** See [BS94, p. 62].

We observe that the approximate solution is *quasi-optimal*, i.e. that the error is proportional to the best approximation in \( S^h \).

We investigate the right hand-side of (2.13) in order to give a more useful error bounds. For this we define the interpolation operator \( I^{k,h} : H^{k+1} \to S^h \), where \( S^h \) is a finite element space of degree \( k \), as:

\[
I^{k,h}(u) := \sum_{j=1}^{N_T} u(x_j) \varphi_j,
\]

where \( \varphi_j \) and \( x_j \) again denote the basis functions and the nodes. For the interpolation error we derive the following bound.

**Lemma 2.5.2:**

Given a regular triangulation \( T \) of \( \Omega \) (see Definition 2.4.1) with maximal mesh width \( h \) and the corresponding finite element space \( S^h \) with polynomials of maximal degree \( k \); then, for \( u \in H^{k+1} \), it holds

\[
\| u - I^{k,h}(u) \|_{L^2(\Omega)} \leq C_1 h^{k+1} \| u \|_{H^{k+1}(\Omega)};
\]

\[
\| u - I^{k,h}(u) \|_{H^1(\Omega)} \leq C_2 h^k \| u \|_{H^{k+1}(\Omega)}.
\]

The constant \( C_1, C_2 \) depend on \( \Omega, k \) and the regularity constant \( c_T \).

**Proof:** See [WM85, p. 189].

Since the error is quasi-optimal, in the finite dimensional space we can combine this result with (2.13) and derive the error bounds for conforming finite elements.
Corollary 2.5.3:
Let assume the same condition as in the Céa’s Lemma 2.5.1 and the condition of Lemma 2.5.2. Then there exist constants $C_3, C_4$ such that
\[
\|u - u^h\|_{L^2(\Omega)} \leq C_3 h^{k+1} \|u\|_{H^{k+1}(\Omega)},
\]
\[
\|u - u^h\|_{H^1(\Omega)} \leq C_4 h^{k} \|u\|_{H^{k+1}(\Omega)}.
\]
\[
(2.14)
\]

**Proof:** See [WM85, p. 189].
Chapter 3

Non-Homogeneous Dirichlet Boundary Conditions

In this chapter we explain how to handle non-homogeneous Dirichlet boundary conditions, i.e. when, at the boundary, the solution $u$ must coincide with a function $g$ not necessarily equal to $0$.

Assume $\Omega$ is a bounded domain in $\mathbb{R}^d$ (in our model problems we restrict to $d = 2$ but also higher dimensional cases can be handled similarly) with a Lipschitz boundary $\partial \Omega$. Then consider the following elliptic PDE:

$$
Lu := -\nabla \cdot (\alpha \nabla u) + cu = f \\
u = g
$$

in $\Omega$, on $\partial \Omega$.  

(3.1)

for some coefficient functions $\alpha, c$ sufficiently smooth such that:

$$
0 < \alpha_0 \leq \alpha(x) \leq \alpha_1 \\
0 \leq c_0 \leq c(x) \leq c_1
$$

with $\alpha_0, \alpha_1, c_0, c_1 \in \mathbb{R}$. For some $p \geq 2$ and given $f \in L^p(\Omega)$ and $g \in H^2(\Omega)$ the corresponding weak formulation [BE86, p. 344] is given by: find $u \in H^1_E(\Omega) := \{w \in H^1(\Omega) : w = g \text{ on } \partial \Omega\}$ such that:

$$
a(u,v) = l(v) \quad \forall v \in H^1_0 := \{w \in H^1(\Omega) : w = 0 \text{ on } \partial \Omega\},
$$

(3.2)

where

$$
a(w, v) := (\alpha \nabla w, \nabla v)_\Omega + (cw, v)_\Omega, \\
l(v) := (f, v)_\Omega
$$

and

$$(w, v)_G := \int_G wv \quad \text{for some domain } G.$$  

If we use the same approach as in the homogeneous case the non-homogeneous boundary conditions, being essential, must be imposed on the admissible functions, i.e. that every admissible function $u$ must coincide with $g$ on the boundary. With this assumption the set
of admissible functions, here $H^1_E$ i.e. the set of functions in $H^1$ which coincide with $g$ on the boundary, is no longer a linear but an affine space: the sum of two admissible functions equals $2g$ on the boundary and thus do not coincide with the boundary conditions (see [SF08]). Clearly we cannot apply the same methods used in the homogeneous case to solve this particular problem, but we need to develop additional tools.

[SF08, WM85, Fai78] describe two possible approaches to solve PDEs provided with non-homogeneous boundary conditions. The first option is to split the solution in two parts $u = u_g + u_0$. The first function coincides with the boundary conditions and the second one is the solution of a modified elliptic PDE equipped with homogeneous boundary conditions (thus solvable using standard procedures). The second option is to approximate the boundary conditions using appropriately chosen Robin boundary conditions with a penalty factor. In this case the test functions must not coincide with a function $g$ on the boundary but we ensure that the boundary conditions hold by adding suitable integrants in the linear and bilinear form.

One central issue is to investigate if the error estimates given at the end of the last chapter for the homogeneous case still holds for non-homogeneous cases or if they need to be adapted. Moreover it is of interest to test the numerical efficiency of the possible methods.

### 3.1 Solution Split

Let us consider again the problem (3.2). The solution $u$ lies in the set of admissible functions $H^1_E$, which in general is clearly not a subset of $H^1_0$. To describe this set we can choose any member $u_g \in H^1_E$ (in particular $u_g$ takes the value $g$ on the boundary), then every function $v \in H^1_E$ is determined by $u_g + v_0$ with $v_0 \in H^1_0$ (the difference between two functions in $H^1_E$ lies in $H^1_0$). Thus [SF08, p. 70]:

$$H^1_E = u_g \oplus H^1_0.$$ (3.3)

Applying the Ritz-Galerkin discretization process we define the admissible functions $v^h(x) = u_g(x) + \sum q_j \varphi_j(x)$ where $\varphi_j$ are the basis functions for the finite element space. The admissible class is then given by $S^h = u_g \oplus S^h_0$ for the finite dimensional space $S^h_0 \subset H^1_0$ containing functions vanishing at the boundary.

To actually solve the non-homogeneous problem we can make the assumption $u = u_0 + u_g$ and reformulate the weak formulation (3.2) as follows: find $u_0 \in H^1_0$ (respectively $u_0^h \in S^h_0$), such that

$$a(u_0, v) = l_0(v) \quad \forall v \in H^1_0,$$
$$a(u_0^h, v^h) = l_0(v^h) \quad \forall v \in S^h_0,$$ (3.4)

where

$$l_0(v) = l(v) - a(u_g, v) \quad \text{respectively} \quad l_0 = l(v^h) - a(u_g, v^h).$$

The problem (3.4) fulfills the assumptions of the Lax-Milgram Theorem, thus existence and uniqueness of the solution follow (see [Bra07, p. 40] or [WM85, p. 78]).

**Remark 3.1.1:**

We define $u_g = g$ on $\partial \Omega$ with $u_g \in H^1_E$ but $g$ may not lie in $H^1_E$. One remedy is to project
3.1. SOLUTION SPLIT

(e.g. via interpolation) the function \( g \) to the set \( H^1_\Omega \), then we can denote this approximation by \( u^h_g \approx u_g \). How to choose the function approximating the boundary conditions is an important issue and various solutions are possible, some of which are given in the next example.

**Example 3.1.2:**
The following example (from [WM85, Section 4.2.3]) shows how the function \( u_g \) (or \( u^h_g \)) are constructed. For a parameter \( \gamma > 0 \), let us consider the following elliptic PDE:

\[
\Delta u + 2\gamma \frac{\partial u}{\partial y} = 0 \quad \text{in } \Omega = \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)^2,
\]

subject to the boundary conditions:

\[
\begin{align*}
u(x, -\frac{\pi}{2}) &= 0 & |y| &\leq \frac{\pi}{2} \\
u(x, \frac{\pi}{2}) &= 0 & |x| &\leq \frac{\pi}{2} \\
u(\pm \frac{\pi}{2}, y) &= 0 & |x| &> \frac{\pi}{2} \\
u(x, \frac{\pi}{2}) &= \left(\frac{\pi}{2}\right)^2 - x^2 & |x| &\leq \frac{\pi}{2}.
\end{align*}
\]

On three edges, the boundary conditions are homogeneous, while on the upper edge \((y = \frac{\pi}{2})\) they are given by the function \( g = (\pi/2)^2 - x^2 \).

In this case, it is possible to find explicitly a function that matches exactly the boundary conditions. For example consider

\[
u^{(1)}_g(x, y) := \left(\left(\frac{\pi}{2}\right)^2 - x^2\right)\left(\frac{\pi}{2} + y\right)\frac{1}{\pi},
\]

clearly \( u^{(1)}_g \) satisfies the boundary conditions and thus we can modify (3.5) to a problem with homogeneous Dirichlet boundary conditions:

\[
\begin{align*}
\Delta u_0 + 2\gamma \frac{\partial u_0}{\partial y} &= -\Delta u^{(1)}_g - 2\gamma \frac{\partial u^{(1)}_g}{\partial y} = 2\left(\frac{\pi}{2} + y\right) - 2\gamma \left(\left(\frac{\pi}{2}\right)^2 - x^2\right) & \text{in } \Omega = \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)^2 \\
\Delta u_0 &= 0 & \text{on } \partial\Omega.
\end{align*}
\]

A different approach is to project the boundary conditions into the finite element space \( S^h \). This can be done, for example, by interpolating the function \( g \) at the boundary nodes (and setting the function at all other nodes to zero). In this case, we employ the ansatz:

\[
u^h = \sum_{j=1}^{N_T} q_j \varphi_j + \sum_{j=N_T+1}^{N_T} q_j \varphi_j
\]

for the approximation of (3.5). Recall that \( \varphi_j \) for \( j = 1, 2, \cdots, N_T \) are the basis functions vanishing at the boundary, while \( \varphi_j \) for \( j = N_T + 1, \cdots, N_T \) are those having non-vanishing at the boundary. In this case

\[
u^{(2)}_g(x, y) := \sum_{j=N_T+1}^{N_T} q_j \varphi_j
\]
will have the values of $g$ at the boundary nodes and will be equal to zero for all nodes inside $\Omega$.

A third possibility, if there exists a smooth continuation of $g$ inside the domain $\Omega$, is to interpolate the extended function $g$ using all basis functions, i.e. not just those on the boundary. In this case we define:

$$u_g^{h,(3)} := \sum_{j=1}^{N_T} q_j \varphi_j.$$ 

In Figure 3.1 on page 24, $u_g^{h,(2)}$ and $u_g^{h,(3)}$ are depicted for the numerical example (3.20).

### 3.1.1 Convergence and Error Estimator

After introducing the various methods to split the solution in a particular solution matching (approximately) the boundary condition and a part with homogeneous boundary condition, it is of interest to verify if the error bounds (2.14) given in the last chapter still hold. In particular it is of interest to investigate if the interpolation at the boundary of the function $u_g$ affects the convergence rate or if they are dominated by the interpolation error.

Let us again consider a general elliptic problem as in (3.1). In this case the following theorem holds.

**Theorem 3.1.3:** Assume that $u$ minimizes the variational form $I(v) = a(v, v) - 2(f, v)$ over $H^1_0$ and $u^h$ over $S^h$ (where $a$ is a bounded, coercive, symmetric bilinear form). Then the vanishing of the first variation is expressed by:

$$a(u, v_0) = l(v_0) \quad \forall v_0 \in H^1_0, \quad (3.6)$$

$$a(u^h, v^h_0) = l(v^h_0) \quad \forall v^h_0 \in S^h_0. \quad (3.7)$$

Moreover $u^h$ has the additional property:

$$a(u - u^h, u - u^h) = \min_{v^h \in S^h} a(u - v^h, u - v^h) \quad (3.8)$$

**Proof:** This proof follows [SF08, p. 200] and some argument are sketched. Since $u$ minimizes the functional $I(v) = a(v, v) - 2(f, v)$, any perturbation $\epsilon v_0$ must increase $I$:

$$I(u) \leq I(u + \epsilon v_0) = a(u + \epsilon v_0, u + \epsilon v_0) - 2(f, u + \epsilon v_0) =$$

$$a(u, u) - 2(f, u) + 2\epsilon \left( a(u, v_0) - (f, v_0) \right) + \epsilon^2 a(v_0, v_0) =$$

$$I(u) + 2\epsilon \left( a(u, v_0) - (f, v_0) \right) + \epsilon^2 a(v_0, v_0)$$

and therefore

$$0 \leq 2\epsilon \left( a(u, v_0) - (f, v_0) \right) + \epsilon^2 a(v_0, v_0).$$

Since this is true for small positive and negative $\epsilon$, the first term must vanish and this proves (3.6). Analogously (3.7) is proved.
To prove (3.8) consider that
\[ a(u - v^h, u - v^h) = a(u - u^h + u^h - v^h, u - u^h + u^h - v^h) = \]
\[ = a(u - u^h, u - u^h) + 2a(u - u^h, u^h - v^h) + a(u^h - v^h, u^h - v^h). \tag{3.9} \]
The second term vanishes; this follows by subtracting (3.7) from (3.6) and choosing \( v_0 = v_0^h = u^h \) to get:
\[ a(u, v_0) - a(u^h, v_0^h) = a(u - u^h, u^h - v^h) = (f, v_0) - (f, v_0^h) = (f, 0) = 0. \]
The last term of (3.9) is strictly positive unless \( v^h \) equals \( u^h \), in which case the minimum occurs at this point. The property (3.8) is therefore proven.

**Corollary 3.1.4:**
Consider the same assumptions as in Theorem 3.1.3, suppose that \( \Omega \) is a polygon and the non-homogeneous Dirichlet boundary conditions \( u = g \) on \( \partial \Omega \) are interpolated in the finite element space: at all boundary nodes, the trial functions satisfy \( v^h(x_j) = g(x_j) \). Then:
\[ a(u - u^h, u - u^h) \leq a \left( u - I^{k,h}(u), u - I^{k,h}(u) \right), \tag{3.10} \]
where \( I^{k,h}(u) \) is the interpolation of the solution in the finite element space.

**Proof:** See [SF08, p. 200]

Similarly we can also give an estimate for alternative norms. In particular let us consider the following elliptic problem:
\[ -\nabla \cdot (\alpha \nabla u) = f \quad \text{in} \ \Omega \]
\[ u = g \quad \text{on} \ \partial \Omega, \tag{3.11} \]
where \( \alpha(x) \) is a function satisfying
\[ 0 < \alpha_0 \leq \alpha(x) \leq \alpha_1 \quad \forall x \in \Omega \tag{3.12} \]
for some constants \( \alpha_0, \alpha_1 \).

**Theorem 3.1.5:**
Let \( u \) and \( u^h \) be defined as in Corollary 3.1.4, then
\[ |u - u^h|_{H^1} \leq C \inf_{\tilde{u} \in S^h_E} |u - \tilde{u}|_{H^1} \tag{3.13} \]
where \( C \) is a constant not depending on \( u \) and \( S^h_E := \{ v^h \in S^h : v^h(x_j) = g(x_j) \ \forall x_j \ \text{nodes on} \ \text{the boundary} \ \partial \Omega \} \).

**Proof:** This proof follows the proof in [Fai78, p. 91]. Recall, from Theorem (3.1.3), that the following holds true for the exact solution \( u \) of (3.11) and the corresponding weak solution \( u^h \):
\[ (\alpha \nabla u, \nabla v_0) = (f, v_0) \quad \forall v_0 \in H^1_0, \tag{3.14} \]
\[ (\alpha \nabla u^h, \nabla v_0^h) = (f, v_0^h) \quad \forall v_0^h \in S^h_0, \tag{3.15} \]
where $(\cdot, \cdot)$ is the $L^2$-scalar product. Since $S^h_0$ is a subspace of $H^1_0$ we can subtract (3.15) from (3.14) and setting $v_0 = v^h_0$, we get
\[ (\alpha \nabla(u - u^h), \nabla v^h_0) = 0 \quad \forall v^h_0 \in H^1_0. \tag{3.16} \]

For some $\tilde{u} \in S^h_E$ we can write:
\[ (\alpha \nabla(u - u^h), \nabla(u - u^h)) = (\alpha \nabla(u - u^h), \nabla(u - \tilde{u} + \tilde{u} - u^h)) =
\[ = (\alpha \nabla(u - u^h), \nabla(u - \tilde{u})) + (\alpha \nabla(u - u^h), \nabla(\tilde{u} - u^h)). \]
Moreover, since $\tilde{u} - u^h \in H^1_0$, from (3.16) it follows that $(\alpha \nabla(u - u^h), \nabla(\tilde{u} - u^h)) = 0$. Then, using the Cauchy-Schwarz inequality and (3.12) we can derive:
\[ (\alpha \nabla(u - u^h), \nabla(u - u^h)) \leq \|\alpha \nabla(u - u^h)\|_{L^2} \cdot |u - \tilde{u}|_{H^1} \leq \alpha_1 |u - u^h|_{H^1} |u - \tilde{u}|_{H^1}. \tag{3.17} \]

From (3.12) we can also derive:
\[ (\alpha \nabla(u - u^h), \nabla(u - u^h)) \geq \alpha_0 |u - u^h|^2_{H^1}. \tag{3.18} \]

Then combining (3.17) and (3.18):
\[ \alpha_0 |u - u^h|^2_{H^1} \leq (\alpha \nabla(u - u^h), \nabla(u - u^h)) \leq \alpha_1 |u - u^h|_{H^1} |u - \tilde{u}|_{H^1} \]
and thus
\[ |u - u^h|_{H^1} \leq \frac{\alpha_1}{\alpha_0} |u - \tilde{u}|_{H^1}. \]

Since $\tilde{u}$ is an arbitrary element of $S^h_E$ this concludes the proof. \hfill \Box

Until now we have defined the approximation $u^h_0 \approx u_g$ just by using the basis functions corresponding to the boundary nodes. If we now assume that there exists a smooth continuation of $g$ into $\Omega$ (and at least one exists: $u$ itself) we can then interpolate this function over $\Omega$ and derive a very effective error estimate.

**Theorem 3.1.6:**

Let $u$ be the solution (3.11) and let $w \in H^1$ be any smooth continuation of $g$ into $\Omega$. We can write the finite element approximation as $u^h = u^h_0 + w^h$, where $w^h \in S^h$ is the interpolation $I^{k,h}(w)$ in the finite element space and $u^h_0 \in S^h_0$. Then:
\[ |u - u^h|_{H^1} \leq C \left( |u - (u^h_0 + w^h)|_{H^1} + |w - w^h|_{H^1} \right) \quad \forall v^h_0 \in S^h_0. \tag{3.19} \]

**Proof:** See [WM85, p. 183]. \hfill \Box

Since $(u - (u^h_0 + w^h)) \in H^1_0$, the error bound is given by the sum of the interpolation error of $w$: $|w - w^h|_{H^1}$ together with the usual interpolation error. Therefore the usual error bounds for the interpolation error is applicable.

In order to have an optimal order $L^2$-estimate we can use the **Aubin-Nitsche Lemma**, that adds the additional $h$-factor to the error estimator [WM85, Section 6.2.2]. To achieve the same result, in [Faï78, p. 92], the problem is investigated from another point of view and the qualitative same estimator is derived.

We can also conclude that the error bounds (2.14) still hold.
3.1. SOLUTION SPLIT

3.1.2 Implementation

The splitting method can be implemented and integrated in a standard FEM solver. Consider again the problem (3.2) and again split the solution into two parts $u^h = u^h_0 + u^h_g$.

Then the solver must be modified as follows:

- Construction of the function $u_g$ and the corresponding projection $u^h_g$ in the finite element space: $u_g$ can be projected by interpolating over the boundary-nodes (solution split over boundary - $u^h_g(2)$ in Example 3.1.2) or interpolating a smooth continuation in the domain (solution split over the domain - $u^h_g(3)$ in Example 3.1.2).

- Adaptation of the linear form by adding $-a(u^h_g, v^h)$ to the right-hand side vector $r$. The system matrix $M$ remains unchanged.

- Solution of the linear system of equation $Mx = r$.

- Reconstruction of the solution of the original problem by adding the solution $u^h_0$ of the homogeneous problem to the function representing the boundary conditions: $u^h = u^h_0 + u^h_g$

Remark 3.1.7:

After the projection, the function $u^h_g$ can be described as a linear combination of the basis functions $\varphi_j$: $u^h_g = \sum_j q_j \varphi_j$ with $q_j \in \mathbb{R}$. If $a(\cdot, \cdot)$ is the bilinear form for the operator $L$, the second step of the algorithm is performed as follows:

$$-a(u^h_g, \varphi_i) = -a \left( \sum_j q_j \varphi_j, \varphi_i \right) = -\sum_j q_j a(\varphi_j, \varphi_i) \quad \forall \varphi_i \text{ basis function.}$$

This operation is easily performed by a computer with a matrix-vector multiplication. Let us consider for example $L = -\Delta$, then $-a(u^h_g, v^h)$ corresponds to:

$$\int_{\Omega} \nabla u^h_g \nabla v^h = \sum_j q_j \int_{\Omega} \nabla \varphi_j \nabla \varphi_i \quad \forall \varphi_i \text{ basis function.}$$

This is calculated as the multiplication $Mq$ where $M$ is the system matrix given by $(m)_{ji} = \int_{\Omega} \nabla \varphi_j \nabla \varphi_i$ and $q$ the vector containing the coefficients $(q_j)$ determining $u^h_g$.

3.1.3 Implementation in NGSolve

In NGSolve this method is already implemented in the procedure “setvalues”: with few lines of code we can adapt the file defining a homogeneous Dirichlet PDE (homogeneous.pde) into a file defining a PDE with non-homogeneous Dirichlet boundary conditions (nonhomogeneous_solsplit.pde).

The modification that must be done are the following:

- we still declare the finite element space having homogeneous Dirichlet boundary conditions using the command:

1The mentioned files can be found in the appendix.
CHAPTER 3. NON-HOMOGENEOUS DIRICHLET BOUNDARY CONDITIONS

define fespace v -order=1 -dirichlet=[1]

where [1] corresponds to the part of the boundary, where the boundary conditions are applied.

• Then we have to define the coefficient function determining to the boundary conditions:

define coefficient dirichlet_bc
(1/(x+y+1)),

• Before executing the actual solver (bvp), we have to call the procedure “setvalues” that calculates the projection $u_h^g$ and upload the linear form (right-hand side).

numproc setvalues [name] -gridfunction=u -coefficient=dirichlet_bc

• The sum $u_0 + u_g$ will be automatically performed after the resolution of the linear system of equations.

Remark 3.1.8:
The procedure “setvalue” projects the smooth continuation of $g$ over the whole domain $\Omega$ to the finite element space (solution split over domain).

If we want to define the function $u_h^g$ only using the boundary nodes (solution split over boundary) we have to modify the procedure “setvalues” by adding a “boundary”-flag:

numproc setvalues [name] -gridfunction=u -coefficient=dirichlet_bc -boundary

(a) Solution split over domain (without -boundary flag)
(b) Solution split over boundary (with -boundary flag)

Figure 3.1: The function $u_h^g$ using the “setvalues” procedure for (3.20) using the two possible projections.

\footnote{This example refers to the problem (3.20)}
Remark 3.1.9:
Instead of a simple interpolation NGsolve uses a $L^2$-projection to project $u_g$ to the finite element space. This means that we search for a function $u^h_g = \mathcal{P}_{L^2}(u_g)$ such that:

$$\int_{\Omega} u^h_g v_h = \int_{\Omega} u_g v_h \quad \forall v^h \in S^h$$

and is equivalent to the minimization of the $L^2$ error:

$$\|u^h_g - u_g\|_{L^2(\Omega)} \rightarrow \min.$$ 

When projecting over the boundary (procedure “setvalues” with “boundary”-flag) the integrals given above are calculated over the boundary $\partial \Omega$ instead of the domain $\Omega$.

3.1.4 Model Problem
Let us consider a non-homogeneous elliptic PDE given by:

$$-\Delta u = -\frac{4}{(x+y+1)^3} + 2\pi^2 \sin(\pi x) \sin(\pi y) \quad \text{in } \Omega = (0,1)^2$$

$$u = \frac{1}{x+y+1} \quad \text{on } \partial \Omega. \quad (3.20)$$

It is straightforward to prove that the function

$$u = \frac{1}{x+y+1} + \sin(\pi x) \sin(\pi y)$$

solves problem (3.20) and can thus be used as a reference solution to compute the error in the $L^2$ norm. Moreover the gradient of $u$ (needed to compute the error in the $H^1$ seminorm) reads

$$\nabla u = \left( \frac{\pi \cos(\pi x) \sin(\pi y) - \frac{1}{(x+y+1)^2}}{\pi \sin(\pi x) \cos(\pi y) - \frac{1}{(x+y+1)^2}} \right).$$

Using the file nonhomog.solsplit.pde we can define problem (3.20) for NGSolve. The same file allows us to calculate the error in the $L^2$ norm and the $H^1$ seminorm (see figure 3.2).

Remark 3.1.10:
The theoretical error bounds are derived for the $H^1$-norm, while we calculate the $H^1$-seminorm. Since the $H^1$-seminorm is majorized by the $H^1$-norm the error bound still holds.

To investigate the behavior of the error we can solve the same problem for multiple, say $s$, steps and at each step we refine the mesh using a bisection algorithm. Because of the particular mesh used for the domain $\Omega = (0,1)^2$ (see Figure 3.3), this algorithm will halve the size $h = \max_r h_r$ of the mesh at every step. According to the error bounds given by (2.14), we expect that, at each step $s$, the error $e^s_{L^2} = \|u - u^h\|_{L^2}$ (for the $L^2$-norm) will decrease with a factor $h^{k+1} = \frac{1}{2^k}$ or, in logarithmic scale, with a factor $-(k+1)$, where $k$ is the degree of the elements (maximal polynomial degree). Similarly we expect a factor $h^k = \frac{1}{2^k}$ (or $-k$ in logarithmic scale) for the $H^1$-seminorm.
CHAPTER 3. NON-HOMOGENEOUS DIRICHLET BOUNDARY CONDITIONS

(a) Solution $u^h$ with linear elements
(b) Error $\|u - u^h\|_{L^2}$

Figure 3.2: Solution and error of problem (3.20) using solution split over boundary method.

(a) Mesh at step $s$ with a size $h = 1/2$
(b) Mesh at step $s + 1$ with a size $h = 1/4$

Figure 3.3: Refining the mesh with a bisection algorithm

In Tables 3.1, 3.2, 3.3, and 3.4 the ratio between each refinement step is depicted: this ratio coincides with the factor given by $2^{k+1}$ ($L^2$ norm) and $2^k$ ($H^1$ seminorm). The errors are very similar for both projections (over boundary or over domain). This is because the bound for the error is dominated in both cases by the interpolation error (for the same finite element space).
3.1. SOLUTION SPLIT

<table>
<thead>
<tr>
<th>step</th>
<th>( k=1 )</th>
<th>( k=2 )</th>
<th>( k=3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \frac{1}{h} )</td>
<td>error</td>
<td>ratio</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.00531</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.00124</td>
<td>4.283</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>0.00030</td>
<td>4.153</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>7.47e-05</td>
<td>3.996</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.90e-05</td>
<td>3.927</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>4.84e-06</td>
<td>3.931</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>1.22e-06</td>
<td>3.954</td>
</tr>
</tbody>
</table>

Table 3.1: Error \( ||u - u^h||_L^2 \) for the \( L^2 \) norm using the solution split over domain

<table>
<thead>
<tr>
<th>step</th>
<th>( k=1 )</th>
<th>( k=2 )</th>
<th>( k=3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \frac{1}{h} )</td>
<td>error</td>
<td>ratio</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.07527</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.03843</td>
<td>1.959</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>0.01945</td>
<td>1.976</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>7.48e-05</td>
<td>3.996</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.90e-05</td>
<td>3.927</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>4.84e-06</td>
<td>3.931</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>1.22e-06</td>
<td>3.954</td>
</tr>
</tbody>
</table>

Table 3.2: Error \( |u - u^h|_{H^1} \) for the \( H^1 \) seminorm using the solution split over domain

<table>
<thead>
<tr>
<th>step</th>
<th>( k=1 )</th>
<th>( k=2 )</th>
<th>( k=3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \frac{1}{h} )</td>
<td>error</td>
<td>ratio</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.00472</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.00112</td>
<td>4.214</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>0.00028</td>
<td>4.058</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>6.90e-05</td>
<td>3.996</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.73e-05</td>
<td>3.996</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>4.32e-06</td>
<td>3.998</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>1.08e-06</td>
<td>3.999</td>
</tr>
</tbody>
</table>

Table 3.3: Error \( ||u - u^h||_L^2 \) for the \( L^2 \) norm using the solution split over boundary

<table>
<thead>
<tr>
<th>step</th>
<th>( k=1 )</th>
<th>( k=2 )</th>
<th>( k=3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \frac{1}{h} )</td>
<td>error</td>
<td>ratio</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.07748</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.03907</td>
<td>1.983</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>0.01959</td>
<td>1.995</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0.00818</td>
<td>1.998</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>0.00491</td>
<td>1.999</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>0.00245</td>
<td>2.000</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>0.00123</td>
<td>2.000</td>
</tr>
</tbody>
</table>

Table 3.4: Error \( |u - u^h|_{H^1} \) for the \( H^1 \) seminorm using the solution split over boundary
Figures 3.4 and 3.5 display the error (in logarithmic scale) for the two (semi)-norms: the expected behavior given by the slope $-k + 1$ respectively $-k$ are respected. The errors from the two projections are very similar and the graphs almost coincide.

![Graphs showing error for different degrees k](image)

Figure 3.4: Logarithm of the error for the $L^2$ norm for some degrees $k$.

![Graphs showing error for different degrees k](image)

Figure 3.5: Logarithm of the error for the $H^1$ seminorm for some degrees $k$. 

3.2 Approximation using Robin Boundary Conditions

In this section we will present a different approach to deal with non-homogeneous Dirichlet boundary conditions. Instead of imposing the boundary conditions and thus modifying the set of admissible functions, we approximate the Dirichlet boundary conditions with boundary conditions of Robin type. In this case we will add some terms in the linear and bilinear-form corresponding to the boundary conditions, without imposing any condition \( (u = g \text{ on } \partial \Omega) \) to the admissible functions.

**Remark 3.2.1:**

Sometimes this method is called penalty-method \([WM85, Bak73, BE86]\) and it may also be used to approximate homogeneous boundary conditions \([Bab73]\). The advantage of this approach is that, without imposing the boundary conditions, we must not ensure that all the basis functions vanish on the boundary.

The key idea of this method is to replace the boundary conditions \( u = g \text{ on } \partial \Omega \) of (3.1) by boundary conditions of Robin type with a penalty factor \( \varepsilon > 0 \):

\[
\alpha \frac{\partial u_\varepsilon}{\partial n} + \varepsilon^{-1}(u_\varepsilon - g) = 0 \quad \text{on } \partial \Omega,
\]

where \( n \) denotes the outward pointing unit normal vector on \( \partial \Omega \).

Let us consider the following penalized problem:

\[
Lu_\varepsilon = f \quad \text{in } \Omega,
\]

\[
\alpha \frac{\partial u_\varepsilon}{\partial n} + \varepsilon^{-1}(u_\varepsilon - g) = 0 \quad \text{on } \partial \Omega. \tag{3.21}
\]

The weak formulation is \([BE86, p. 345]\): find \( u_\varepsilon \in H^1(\Omega) \) such that:

\[
a_\varepsilon(u_\varepsilon, v) = l_\varepsilon(v) \quad \forall v \in H^1(\Omega), \tag{3.22}
\]

where

\[
a_\varepsilon(w, v) := a(w, v) + \varepsilon^{-1} \langle w, v \rangle_{\partial \Omega},
\]

\[
l_\varepsilon(v) := l(v) + \varepsilon^{-1} \langle g, v \rangle_{\partial \Omega}
\]

and

\[
\langle w, v \rangle_{\partial \Omega} := \int_{\partial \Omega} wv.
\]

Since \( \varepsilon > 0 \), it follows that \( a_\varepsilon(\cdot, \cdot) \) is continuous and coercive; therefore the problem is well-posed.

As previously stated, the boundary conditions are not imposed as essential condition for the admissible space \( H^1 \), but they are naturally satisfied by the weak formulation (3.22) (exactly as in the case of Neumann boundary conditions). In particular by the additional terms \( \varepsilon^{-1} \langle w, v \rangle_{\partial \Omega} \) for the bilinear-form and \( \varepsilon^{-1} \langle g, v \rangle_{\partial \Omega} \) for the linear-form.

For \( \varepsilon \rightarrow 0 \), the equation determining the boundary conditions in (3.21) is dominated by the part \( (u_\varepsilon - g) \), so we expect that this assumption approximates the usual Dirichlet boundary conditions. A proof that this approximate solution converges to the exact one is given in the next section.
3.2.1 Convergence and Error Estimate

Before discussing the error bound of the finite element approximation, we have to show that the approximation using Robin boundary conditions converges to the original problem when \( \varepsilon \to 0 \).

**Theorem 3.2.2:**
Consider problem (3.1), (3.21) and assume that the corresponding bilinear forms \( a \) and \( a_\varepsilon \) are bounded and coercive. Then the solution \( u \) of the original problem (3.1) and the solution \( u_\varepsilon \) of the penalized problem (3.21) satisfy:

\[
\|u - u_\varepsilon\|_{L^2(\Omega)} \leq C\varepsilon \|u\|_{H^2(\Omega)},
\]

for some constant \( C \) not depending on \( \varepsilon \).

**Proof:** The proof follows \([BE86, p. 348]\).

Subtracting (3.22) from (3.2) and using the definition of the Robin boundary conditions we get:

\[
a_\varepsilon(u - u_\varepsilon, v) = \varepsilon^{-1} (g - u_\varepsilon, v)_{\partial\Omega} - \left\langle \alpha \frac{\partial u}{\partial n}, v \right\rangle_{\partial\Omega} \quad \forall v \in H^1(\Omega). \tag{3.24}
\]

Choosing \( v = u - u_\varepsilon \), we obtain

\[
\varepsilon^{-1}\|u - u_\varepsilon\|_{L^2(\partial\Omega)}^2 = \varepsilon^{-1} \langle u - u_\varepsilon, u - u_\varepsilon \rangle_{\partial\Omega} \leq a(u - u_\varepsilon, u - u_\varepsilon) + \varepsilon^{-1} \langle u - u_\varepsilon, u - u_\varepsilon \rangle_{\partial\Omega} = a_\varepsilon(u - u_\varepsilon, u - u_\varepsilon) = \left\langle \alpha \frac{\partial u}{\partial n}, u - u_\varepsilon \right\rangle_{\partial\Omega}. \tag{3.25}
\]

Consider now the trace inequality (see \([BE86, p. 347]\)):

\[
\left\| \frac{\partial w}{\partial n} \right\|_{H^{m-1}(\partial\Omega)} \leq C\|w\|_{H^{m+1}(\Omega)} \quad \forall w \in H^{m+1}(\Omega) \tag{3.26}
\]

for some constant \( C \) and some \( m \in \mathbb{N} \). Using (3.26) and (3.25) we derive:

\[
\left\langle \alpha \frac{\partial u}{\partial n}, u - u_\varepsilon \right\rangle_{\partial\Omega} \leq C\alpha \left\| \frac{\partial u}{\partial n} \right\|_{L^2(\partial\Omega)} \|u - u_\varepsilon\|_{L^2(\partial\Omega)} \leq \|u\|_{H^2(\Omega)} \|u - u_\varepsilon\|_{L^2(\partial\Omega)}
\]

and thus

\[
\|u - u_\varepsilon\|_{L^2(\partial\Omega)} \leq C\varepsilon \|u\|_{H^2(\Omega)}. \tag{3.27}
\]

Using the Cauchy-Schwarz inequality, we observe that

\[
\|u - u_\varepsilon\|_{L^2(\partial\Omega)} = \sup_{\eta \in L^2(\Omega)} \frac{|(u - u_\varepsilon, \eta)_{\Omega}|}{\|\eta\|_{L^2(\Omega)}}. \tag{3.28}
\]

Recalling that \( L \) is the symmetric operator, for any \( \eta \in L^2(\Omega) \), we define \( z \) such that

\[
Lz = \eta \text{ in } \Omega \quad z = 0 \text{ on } \partial\Omega, \tag{3.29}
\]
3.2. APPROXIMATION USING ROBIN BOUNDARY CONDITIONS

then it follows from elliptic regularity, for either $\partial \Omega$ smooth or $\Omega$ convex polyhedral, that

$$
\|z\|_{H^2(\Omega)} \leq C\|\eta\|_{L^2(\Omega)}. \tag{3.30}
$$

Note that as $z \in H^1_0(\Omega)$ we have

$$
|(u - u_\epsilon, Lz)_{\Omega}| = \left| a(u - u_\epsilon, z) - \left< u - u_\epsilon, \frac{\partial z}{\partial n} \right> \right| \leq C\varepsilon\|u\|_{H^2(\Omega)}\|z\|_{H^2(\Omega)} \tag{3.31}
$$

where we have noted (3.24) and applied the bound (3.27) and the trace inequality (3.26). Combining (3.28), (3.29), (3.30) and (3.31) we get:

$$
\|u - u_\epsilon\|_{L^2(\Omega)} \leq C\varepsilon\|u\|_{H^2(\Omega)},
$$

and thus the theorem is proven.

The next issue is to determine wherever the error bound using finite elements is still governed by the interpolation error and thus the convergence rate is optimal. Let us define $u_h^\epsilon \in S^h$ such that:

$$
a_\epsilon(u_h^\epsilon, v^h) = l_\epsilon(v^h) \quad \forall v^h \in S^h. \tag{3.32}
$$

Considering that

$$
\|u - u_h^\epsilon\|_{L^2} = \|u - u_\epsilon + u_\epsilon - u_h^\epsilon\|_{L^2} \leq \|u - u_\epsilon\|_{L^2} + \|u_\epsilon - u_h^\epsilon\|_{L^2} \leq C_1\varepsilon\|u\|_{H^2} + C_2 h^{k+1}\|u\|_{H^{k+1}},
$$

one can assume that for $\varepsilon \leq h^{k+1}$ the usual convergence rate is valid.

The following theorem ensures that the usual convergence rate holds for the case where $\Omega$ is a convex polygon. For more complex cases regarding variational crimes one can find detailed explanations in [BE86].

**Theorem 3.2.3:**

Assume $\varepsilon = h^\lambda$ with $\lambda > 0$, then the solution $u$ of (3.1) and the solution $u_h^\epsilon$ of (3.32) satisfy:

$$
\|u - u_h^\epsilon\|_{L^2(\partial\Omega)} \leq \begin{cases} 
Ch^{\hat{\mu}_2}\|u\|_{H^{k+1}(\Omega)} \\
Ch^{\hat{\mu}_2}\|u\|_{H^{k+2}(\Omega)}
\end{cases} \tag{3.33}
$$

and for $i = 0, 1$ (thus $L^2$-norm and $H^1$-seminorm):

$$
|u - u_h^\epsilon|_{H^i(\Omega)} \leq \begin{cases} 
Ch^{\hat{\mu}_i}\|u\|_{H^{k+1}(\Omega)} \\
Ch^{\hat{\mu}_i}\|u\|_{H^{k+2}(\Omega)}
\end{cases} \tag{3.34}
$$

where

$$
\hat{\mu}_2 = \min[\lambda, k + \frac{1}{2}, k + \frac{1}{2}\lambda], \\
\hat{\mu}_2^* = \min[\lambda, k + 1, k + \frac{1}{2}\lambda], \\
\hat{\mu}_1^{(s)} = \min[k, \hat{\mu}_2^{(s)}], \\
\hat{\mu}_0^{(s)} = \min[\hat{\mu}_1^{(s)} + 1, \hat{\mu}_2^{(s)}],
$$

assuming $u$ sufficiently smooth. Here $k$ denotes the local polynomial degree.
CHAPTER 3. NON-HOMOGENEOUS DIRICHLET BOUNDARY CONDITIONS

Proof: See [BE86, p. 356]

It follows that \( \lambda \geq k + 1 \) yields optimal \( H^1 \) and \( L^2 \) convergence rate.

Remark 3.2.4:
Babuška [Bab73, p. 221] also shows that the approximate solution behaves quite robust with respect to the choice of the penalty parameter.

3.2.2 Implementation

We can adapt the finite element solver to handle Robin boundary conditions and, providing an appropriate penalty factor, to approximate Dirichlet boundary conditions.

- If we start from a finite element space for homogeneous boundary conditions (\( S^h_0 \)) we need to incorporate the basis functions with value on the boundary nodes in the finite element space \( S^h \). Thus each function \( v \in S^h \) can be defined as:

\[
v^h = \sum_{j=0}^{N_T} q_j \varphi_j + \sum_{j=N_T+1}^{N_T+N_T+1} q_j \varphi_j',
\]

where the basis functions \( \varphi_j \) are defined as:

\[
\varphi_j \in S^h_0, \quad j = 0, 1, \ldots, N_T,
\]

\[
\varphi_j \in S^h = S^h \setminus S^h_0, \quad j = N_T + 1, \ldots, N_T.
\]

- We have to choose the penalty factor \( \varepsilon \). In the previous section we have shown that for \( \varepsilon \leq h^{k+1} \) the solution converges and the error is dominated by the interpolation error. Moreover, the solution behave quite robust with respect to the penalty parameter (see Remark 3.2.4). This is why, for our computations we use a fixed factor (for example \( \varepsilon = 10^{-15} \)) that is smaller than the one needed for very fine mesh (size of order \( 10^{-3} \)).

We have to remark that the penalty factor is only applied for the nodes that lie on the boundary and this affect the condition number of the system matrix. To prevent that this matrix become ill-conditioned (affecting the accurateness of the iterative solver), very small penalty factors are to avoid.

- Lastly we have to adapt the bilinear and linear form by adding the corresponding integrals as in (3.22).

3.2.3 Implementation in NGSolve

We start from the file implemented to solve an elliptic problem provided with homogeneous Dirichlet boundary conditions (\( \text{homogeneous.pde} \)) and we adapt it to Robin boundary conditions (the adapted file is named \( \text{nonhomogeneous_robin.pde} \)).

Consider that the Robin boundary conditions are defined in NGSolve as

\[
\frac{\partial u_r}{\partial n} = \gamma(g - u_r)
\]

for a factor \( \gamma \), that clearly correspond to \( 1/\varepsilon \). To implement the Robin boundary conditions method we modify the following code lines:
3.2. APPROXIMATION USING ROBIN BOUNDARY CONDITIONS

- We define a constant encompassing the penalty factor $\gamma = 1/\varepsilon$:

  \[
  \text{define constant penal = 1e15}
  \]

- We define the coefficient functions that define $\gamma$ to implement $\int_{\partial \Omega} \gamma uv$ (in this case named $lhs$) and $\gamma g$ to implement $\int_{\partial \Omega} \gamma gv$ (here $rhs$)\(^3\):

  \[
  \begin{align*}
  &\text{define coefficient lhs (penal),} \\
  &\text{define coefficient rhs (penal*((1/(x+y+1))))},
  \end{align*}
  \]

- Once we define the finite element space used to solve the problem we remove the flag imposing the homogeneous boundary conditions (\texttt{-dirichlet=[1]}):

  \[
  \text{define fespace v -order=1}
  \]

- To define the bilinear form (left-hand side) we have to include the additional integral:

  \[
  \begin{align*}
  &\text{define bilinearform a -fespace=v -symmetric} \\
  &\text{laplace one} \\
  &\text{robin lhs}
  \end{align*}
  \]

  In this case \texttt{one} is the coefficient function for the Laplace operator (for problem (3.20) \texttt{one} = 1), while \texttt{robin lhs} will add $\int_{\partial \Omega} \gamma uv$ to the bilinear-form.

- Similarly we modify the linear form:

  \[
  \begin{align*}
  &\text{define linearform l -fespace=v} \\
  &\text{source f} \\
  &\text{neumann rhs}
  \end{align*}
  \]

  where \texttt{source f} stands for the integral $\int_{\Omega} fv$ and \texttt{neumann rhs} corresponds to $\int_{\partial \Omega} \gamma gv$.

  The remaining lines of code remain unchanged.\(^4\)

3.2.4 Model Problem

Consider the model problem given by (3.20).

The Robin boundary conditions approximation can be implemented in NGSolve using the file \texttt{nonhomogeneous_robin.pde}. We again solve this problem for various degrees $k$ and we refine the mesh for multiple steps $s$.

The results, listed in Tables 3.5 and 3.6, agree with the error bounds proven: the error decreases with a factor $h^{k+1}$ for the $L^2$ norm and with a factor $h^k$ for the $H^1$ seminorm. Moreover, the error is very similar to the error which results when using the solution split algorithm: clearly this is a consequence of the dominance of the interpolation error.

The same behaviour is displayed in the Figures 3.6 and 3.7: the error decreases at the expected rate.

\(^3\)In this case we refer to the boundary conditions for problem (3.20)

\(^4\)The file \texttt{nonhomogeneous_robin.pde} used to solve the problem (3.20) can be found in the appendix.
Table 3.5: Error $||u - u^h||_{L^2}$ for the $L^2$ norm using the Robin boundary conditions approximation

<table>
<thead>
<tr>
<th>step</th>
<th>$\frac{1}{h}$</th>
<th>$k=1$</th>
<th>error</th>
<th>ratio</th>
<th>$k=2$</th>
<th>error</th>
<th>ratio</th>
<th>$k=3$</th>
<th>error</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.00442</td>
<td>-</td>
<td>0.00052</td>
<td>-</td>
<td>4.53e-05</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.00114</td>
<td>3.857</td>
<td>7.37e-05</td>
<td>7.084</td>
<td>3.00e-06</td>
<td>15.120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>0.00029</td>
<td>3.972</td>
<td>9.67e-06</td>
<td>7.624</td>
<td>1.88e-07</td>
<td>15.973</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>7.18e-05</td>
<td>4.016</td>
<td>1.23e-06</td>
<td>7.888</td>
<td>1.17e-08</td>
<td>16.055</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.78e-05</td>
<td>4.022</td>
<td>1.54e-07</td>
<td>7.945</td>
<td>7.28e-10</td>
<td>16.051</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>4.44e-06</td>
<td>4.017</td>
<td>1.92e-08</td>
<td>8.055</td>
<td>4.54e-11</td>
<td>16.038</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>1.10e-06</td>
<td>4.021</td>
<td>2.39e-09</td>
<td>8.013</td>
<td>2.83e-12</td>
<td>16.020</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.6: Error $|u - u^h|_{H^1}$ for the $H^1$ seminorm using the Robin boundary conditions approximation

<table>
<thead>
<tr>
<th>step</th>
<th>$\frac{1}{h}$</th>
<th>$k=1$</th>
<th>error</th>
<th>ratio</th>
<th>$k=2$</th>
<th>error</th>
<th>ratio</th>
<th>$k=3$</th>
<th>error</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.08294</td>
<td>-</td>
<td>0.01068</td>
<td>-</td>
<td>0.00143</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.04101</td>
<td>2.022</td>
<td>0.00276</td>
<td>3.866</td>
<td>0.00018</td>
<td>7.729</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>0.02013</td>
<td>2.037</td>
<td>0.00069</td>
<td>3.962</td>
<td>2.33e-05</td>
<td>8.283</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0.00994</td>
<td>2.025</td>
<td>0.00017</td>
<td>3.996</td>
<td>2.70e-06</td>
<td>8.271</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>0.00494</td>
<td>2.013</td>
<td>4.35e-05</td>
<td>4.009</td>
<td>3.31e-07</td>
<td>8.161</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>0.00246</td>
<td>2.006</td>
<td>1.09e-05</td>
<td>4.007</td>
<td>4.09e-08</td>
<td>8.084</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>0.00123</td>
<td>2.003</td>
<td>2.74e-06</td>
<td>3.967</td>
<td>5.09e-09</td>
<td>8.043</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.6: Logarithm of the error for the $L^2$ norm for some degrees $k$. 
Figure 3.7: Logarithm of the error for the $H^1$ seminorm for some degrees $k$. 
3.3 Time Performance

After investigating the two possible methods and proving that the error is still bounded by the interpolation error, it is of great interest to investigate whether there is a difference in terms of the time performance between the two methods.

Note that $N$ is the amount of nodes in the mesh (amount of degrees of freedom). The time performance (or time complexity) of an algorithm is given as the amount of elementary operations required with respect to $N$.

Note that in the FEM every function $v \in S^h$ is uniquely determined by its coefficient $q_j$ (the basis functions $\varphi_j$ are fixed) and thus it can be stored in a vector of length $N$.

3.3.1 CG Algorithm

To solve a PDE using the standard finite element method few steps are needed: once the linear and bilinear forms are defined, the solver assembles the (sparse) system matrix and the vector containing the data from the right-hand side. The most expensive part is, however, the solution of the linear system of equations. NGSolve manages this part in multiple ways; for our computations we employ the well-known Conjugate Gradient algorithm (CG-algorithm).

In [Zul11, Chapter 8] the entire algorithm is presented, together with a convergence analysis. In the current section we restrict to briefly summarize the time complexity of this algorithm.

For a given symmetric, positive definite matrix $M \in \mathbb{R}^{N \times N}$ and a vector $r \in \mathbb{R}^N$ we want to solve a linear system of equations given by:

$$Mx = r. \quad (3.35)$$

Let us define the following norm:

$$\|x\|_M = \sqrt{x^T M x}.$$

The CG-method is an iterative solver and at each iteration step $s$ we have the following error estimate [Zul11, S. 108]:

$$\|x_s - x\|_M \leq 2q^s \|x_0 - x\|_M \quad \text{with } q = \frac{\sqrt{\kappa(M)} - 1}{\sqrt{\kappa(M)} + 1}, \quad (3.36)$$

where $\kappa(M)$ is the condition number for the system matrix $M$.

(3.36) gives an estimation with regard to the error at the first step $\|x_0 - x\|_M$ and we are interested to verify how many steps $S$ are needed to reduce this error by a factor $\varepsilon$. This can be guaranteed for

$$2q^S \leq \varepsilon,$$

and this means that we need

$$S \geq \frac{\ln(\varepsilon/2)}{\ln(q)} \quad (3.37)$$

iterations.
3.3. TIME PERFORMANCE

Note that:

\[ q = \frac{\sqrt{\kappa(M)} - 1}{\sqrt{\kappa(M)} + 1} = \frac{\sqrt{\kappa(M)} + 1 - 2}{\sqrt{\kappa(M)} + 1} = 1 - \frac{2}{\sqrt{\kappa(M)} + 1} \]  

(3.38)

and

\[ \frac{\ln(\varepsilon/2)}{\ln(q)} = - \frac{\ln(\varepsilon/2)}{-\ln(1 - \frac{2}{\sqrt{\kappa(M)} + 1})} \leq \left( -\frac{1}{2} \ln(\varepsilon/2) \right) \left( \sqrt{\kappa(M)} + 1 \right). \]

(3.39)

Thus we can conclude that we need

\[ S = \mathcal{O} \left( -\ln(\varepsilon/2)\sqrt{\kappa(M)} \right) \]

(3.40)

iterations to achieve the needed accuracy. It is to remark that the accuracy factor \( \varepsilon \) must decrease according to the mesh size, so that the solver can guarantee the needed accuracy with respect to the FEM error.

The system matrix \( M \) is, in the case of the FEM, a sparse matrix where the number of non-zero entries is of order \( N \), so that every iteration can be performed with linear order of elementary operation. It follows that to achieve the target accuracy we need

\[ \mathcal{O} \left( -\ln(\varepsilon/2)\sqrt{\kappa(M)} N \right) \]

(3.41)

elementary operations. Moreover for the discretization of a \( d \)-dimensional problem we know [Zul11, p. 110]: \( \kappa(M) = \mathcal{O}(h^{-2}) \) and \( N = \mathcal{O}(h^{-d}) \). Therefore for 2D problems, we need:

\[ \mathcal{O} \left( -\ln(\varepsilon/2)\sqrt{\kappa(M)} N \right) = \mathcal{O}(-\ln(\varepsilon/2)\sqrt{N} \cdot N) = \mathcal{O}(-\ln(\varepsilon/2)N^{\frac{3}{2}}) \]

(3.42)

elementary operations.

Since the solution of the linear system is the most expensive operation within the FEM, the time complexity for the whole algorithm is of order \( \mathcal{O}(-\ln(\varepsilon/2)N^{\frac{3}{2}}) \).

At this point it is of interest to know if the modification introduced in the Sections 3.1.2 and 3.2.2 used to solve non-homogeneous problems affect significantly the time complexity.

3.3.2 Solution Split

To analyse the cost of the solution split method, we have to consider every step introduced in Section 3.1.2.

- **Interpolation of \( u^h_g \):** To interpolate \( u_g \) in \( S^h \) we compute the corresponding value \( u^h_g(x_j) = u_g(x_j) \) for every node \( x_j \). If we interpolate the function by a function with degrees of freedom located only at the boundary (solution split over the boundary), this will be done just for the nodes that lie on the boundary. On the other hand, if we use the smooth continuation of \( g \) (solution split over the domain), then this interpolation must be carried out for every node.

In both cases the order of elementary operation needed is given by \( \mathcal{O}(N) \) but clearly the operations needed are significantly less when interpolating only on the boundary because the amount of nodes lying on the boundary is significantly less then the total amount of nodes.
CHAPTER 3. NON-HOMOGENEOUS DIRICHLET BOUNDARY CONDITIONS

- **Adaptation of the right-hand side:** From Remark 3.1.7 we know that the adaptation of the right-hand side is done by a matrix-vector multiplication \((Mq)\) where \(q\) is the vector determining \(u_h^q\) followed by the sum of the resulting vector (with negative sign) with the original vector \(r\). Clearly the sum of two vectors of size \(N\) needs \(O(N)\) elementary operations.

Since the system matrix \(M\) is a sparse matrix with a number of non-zero entries of order \(N\) the matrix-vector multiplication also needs \(O(N)\) elementary operations.

- **Reconstruction of the solution of the original problem:** In this step we sum the two vectors that define \(u_h^q\) and \(u_h^0\) (the solution of the homogeneous problem): this step needs \(O(N)\) operations.

In conclusion we see that the time complexity of the solution split method is still dominated by the cost of the solution of the linear system of equations.

### 3.3.3 Approximation with Robin Boundary Conditions

For the approximation of Dirichlet boundary condition by Robin boundary conditions only the last step from Section 3.2.2 adds some elementary operations to the time complexity of the algorithm.

Let us first consider the left-hand side. The matrix \(M_{\epsilon}\) is assembled as follows:

\[
(m_{\epsilon})_{ji} = a_{\epsilon}(\varphi_j, \varphi_i) = a(\varphi_j, \varphi_i) + \epsilon^{-1} \langle \varphi_j, \varphi_i \rangle_{\partial\Omega},
\]

and this matrix takes non-zero values only for \(O(N)\) entries. Therefore, in an analogous way as the assembling of the usual system matrix \(M\), the whole process needs \(O(N)\) elementary operations.

To assemble the right-hand side vector \((r_{\epsilon})_i = l(\varepsilon_i) + \epsilon^{-1} \langle g, \varphi_i \rangle_{\partial\Omega}\) we calculate numerically \(O(2N)\) times an integral. Using a quadrature rule (for example the Gaussian-quadrature), the amount of elementary operations needed is constant, which means that the whole process requires \(O(N)\) elementary operations. For the additional integrals due to the Robin boundary conditions, they must only be calculated for the basis functions not vanishing at the boundary.

In conclusion we find that the time complexity of the Robin boundary conditions approximation is again dominated by the solution of the linear system \(M_{\epsilon}x = r_{\epsilon}\) and thus the required linear operation are of order \(O(-\ln(\varepsilon/2)N^{3/2})\).

### 3.3.4 Numerical Results

The introduced methods are tested using the server “Asprey” from the Mathematics Institute of the University of Zurich. This server is provided with two Intel Xeon E5-2643 Quadcore CPU of 3.3 GHz and it runs the operating system “Ubuntu 12.04.1 LTS” with kernel version 3.2.0-35-generic.

In particular we solved the model problem (3.20) refining the mesh at each step. The bisection algorithm used to refine the mesh increases the number of nodes by a factor of about 4 \((N \rightarrow 4N)\) for two dimensional problem (not depending on \(k\)). We therefore expect that the elapsed time grows with a factor of \(4^{3/2} = 8\) (the accuracy factor \(\varepsilon\) is fixed by the iterative solver).

From Tables 3.7, 3.8 and 3.9 we can observe this factor for all our methods: the time elapsed increases by a factor between 4 and 6, thus less than the “worst case-factor” 8. The
3.3. TIME PERFORMANCE

The solver of NGSolve is refined to handle sparse matrices and uses a preconditioner, that is why the solution is faster than expected. More information about preconditioning in NGSolve can be found in the official manual (see Section 1.2). The elapsed time for the first steps is omitted in the following tables because it is too small to be meaningful compared.

<table>
<thead>
<tr>
<th>step</th>
<th>N</th>
<th>Sol-split Domain</th>
<th>Sol-split Boundary</th>
<th>Robin boundary c.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time (s) factor</td>
<td>Time (s) factor</td>
<td>Time (s) factor</td>
</tr>
<tr>
<td>3</td>
<td>289</td>
<td>0.02</td>
<td>0.04</td>
<td>2.000</td>
</tr>
<tr>
<td>4</td>
<td>1089</td>
<td>0.07</td>
<td>0.05</td>
<td>1.250</td>
</tr>
<tr>
<td>5</td>
<td>4225</td>
<td>0.19</td>
<td>0.17</td>
<td>3.400</td>
</tr>
<tr>
<td>6</td>
<td>16641</td>
<td>0.86</td>
<td>0.59</td>
<td>3.471</td>
</tr>
<tr>
<td>7</td>
<td>66049</td>
<td>3.34</td>
<td>3.04</td>
<td>5.153</td>
</tr>
<tr>
<td>8</td>
<td>263169</td>
<td>14.14</td>
<td>13.06</td>
<td>4.296</td>
</tr>
<tr>
<td>9</td>
<td>1050625</td>
<td>61.87</td>
<td>56.77</td>
<td>4.347</td>
</tr>
</tbody>
</table>

Table 3.7: Elapsed time for \( k = 1 \)

<table>
<thead>
<tr>
<th>step</th>
<th>N</th>
<th>Sol-split Domain</th>
<th>Sol-split Boundary</th>
<th>Robin boundary c.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time (s) factor</td>
<td>Time (s) factor</td>
<td>Time (s) factor</td>
</tr>
<tr>
<td>3</td>
<td>1089</td>
<td>0.04</td>
<td>0.03</td>
<td>1.000</td>
</tr>
<tr>
<td>4</td>
<td>4225</td>
<td>0.1</td>
<td>0.09</td>
<td>3.000</td>
</tr>
<tr>
<td>5</td>
<td>16641</td>
<td>0.43</td>
<td>0.37</td>
<td>4.111</td>
</tr>
<tr>
<td>6</td>
<td>66049</td>
<td>1.64</td>
<td>1.49</td>
<td>4.027</td>
</tr>
<tr>
<td>7</td>
<td>263169</td>
<td>7.85</td>
<td>7.33</td>
<td>4.919</td>
</tr>
<tr>
<td>8</td>
<td>1050625</td>
<td>42.92</td>
<td>40.65</td>
<td>5.546</td>
</tr>
<tr>
<td>9</td>
<td>4198401</td>
<td>215.13</td>
<td>211.95</td>
<td>5.214</td>
</tr>
</tbody>
</table>

Table 3.8: Elapsed time for \( k = 2 \)

<table>
<thead>
<tr>
<th>step</th>
<th>N</th>
<th>Sol-split Domain</th>
<th>Sol-split Boundary</th>
<th>Robin boundary c.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time (s) factor</td>
<td>Time (s) factor</td>
<td>Time (s) factor</td>
</tr>
<tr>
<td>3</td>
<td>2401</td>
<td>0.05</td>
<td>0.05</td>
<td>2.500</td>
</tr>
<tr>
<td>4</td>
<td>9409</td>
<td>0.15</td>
<td>0.12</td>
<td>2.400</td>
</tr>
<tr>
<td>5</td>
<td>37249</td>
<td>0.59</td>
<td>0.54</td>
<td>4.500</td>
</tr>
<tr>
<td>6</td>
<td>148225</td>
<td>2.73</td>
<td>2.48</td>
<td>4.593</td>
</tr>
<tr>
<td>7</td>
<td>591361</td>
<td>15.06</td>
<td>14.21</td>
<td>5.730</td>
</tr>
<tr>
<td>8</td>
<td>2362369</td>
<td>101.5</td>
<td>95.95</td>
<td>6.752</td>
</tr>
<tr>
<td>9</td>
<td>9443329</td>
<td>641.88</td>
<td>612.07</td>
<td>6.379</td>
</tr>
</tbody>
</table>

Table 3.9: Elapsed time for \( k = 3 \)
The same results can be observed in Figures 3.8, 3.9 and 3.10: the time complexity grows less than the expected factor $3/2$ (calculated in logarithm scale).

Figure 3.8: Logarithm of the elapsed time for $k = 1$.

Figure 3.9: Logarithm of the elapsed time for $k = 2$.

Figure 3.10: Logarithm of the elapsed time for $k = 3$. 
Chapter 4

Model Problem 2

To test the reliability and efficiency of the methods introduced in the previous chapters, let us consider a more complex example. This example was introduced by S. Sauter and used in [BN11] to test the performance of hp-adaptive Finite Elements.

4.1 Setting

4.1.1 Periodic Functions

Let $\tilde{Q} = (0,1)^2$ be the master cell. For $\epsilon = \frac{1}{N} (N \in \mathbb{N}_{\geq 1})$ and $\mu = (\mu_1, \mu_2)^T \in \mathbb{Z}^2$, we define the $\epsilon$-scaled and $\mu$-shifted version of $\tilde{Q}$ by

$$Q_{\epsilon, \mu} := ((\mu_1 - 1)\epsilon, \mu_1 \epsilon) \times ((\mu_2 - 1)\epsilon, \mu_2 \epsilon).$$

We can describe the change of coordinates ($\chi_{\epsilon, \mu} : \tilde{Q} \to Q$) as:

$$\chi_{\epsilon, \mu}(\tilde{x}) = \epsilon(\mu + \tilde{x}) \quad \text{with} \quad \tilde{x} \in \tilde{Q} \subset \mathbb{R}^2.$$ 

Clearly $\chi_{\epsilon, \mu}$ is an affine map whose inverse is given by: $\chi_{\epsilon, \mu}^{-1}(x) = \frac{x}{\epsilon} - \mu$.

**Definition 4.1.1:**
A function $g : \tilde{Q} \to \mathbb{C}$ is called $\tilde{Q}$-periodic if $\forall (x_1, x_2)^T \in \tilde{Q}$

$$g(0, x_2) = g(1, x_2) \quad \text{and} \quad g(x_1, 0) = g(x_1, 1)$$

Furthermore, we can define the periodic, $\epsilon$-scaled and $\mu$-shifted version of $g$ as:

$$g_{\epsilon, \mu}^{\text{per}} |_{Q_{\epsilon, \mu}} = g \circ \chi_{\epsilon, \mu}^{-1}, \quad g_{\epsilon, \mu}^{\text{per}} : Q \to \mathbb{C}.$$ 

For shorter notation we omit to write the vector $\mu$ ($g_{\epsilon}^{\text{per}}$ instead of $g_{\epsilon, \mu}^{\text{per}}$) and we just talk about $\epsilon$-scaled periodic functions.

4.1.2 Coefficient functions

For some real valued constants $\gamma, \Gamma, \beta$ let $a, b \in C^\infty(\tilde{Q})$ be any $\tilde{Q}$-periodic functions such that:

$$0 < \gamma \leq \inf_{x \in \tilde{Q}} a(x) \leq \sup_{x \in \tilde{Q}} a(x) \leq \Gamma \quad \text{and} \quad 0 < \beta \leq \inf_{x \in \tilde{Q}} b(x).$$

(4.1)
Further let $a_{\varepsilon}^{\text{per}}, b_{\varepsilon}^{\text{per}}$ be the $\varepsilon$-scaled (periodic) version of $a$ and $b$ (see Definition 4.1.1).

For a small and fixed $p \in \mathbb{N}_0$ consider the finite subset $T_p := \{ \kappa = (k_1, k_2) \in \mathbb{Z}^2 : -p \leq k_1, k_2 \leq p \} \subset \mathbb{Z}^2$ and the following (diffusion and reaction coefficient) functions
\[
a(x) = \sum_{\kappa \in T_p} a_{\kappa} e^{2\pi i \langle \kappa, x \rangle} \quad \text{and} \quad b(x) = \sum_{\kappa \in T_p} b_{\kappa} e^{2\pi i \langle \kappa, x \rangle}
\]
with real coefficients $a_{\kappa}, b_{\kappa}$ such that (4.1) is satisfied. Here $\langle \cdot, \cdot \rangle$ is the standard scalar product without complex conjugation.

**Remark 4.1.2:**
One can easily verify that if $a_{\kappa} = a_{-\kappa}$ and $b_{\kappa} = -b_{-\kappa}$, then $a(x), b(x)$ are real-valued functions and they can also be described as:
\[
a(x) = \sum_{\kappa \in T_p} a_{\kappa} \cos (2\pi \langle \kappa, x \rangle) \quad \text{and} \quad b(x) = \sum_{\kappa \in T_p} b_{\kappa} \cos (2\pi \langle \kappa, x \rangle).
\]
Furthermore the $\varepsilon$-scaled periodic versions read:
\[
a_{\varepsilon}(x) = \sum_{\kappa \in \mathbb{Z}_2} a_{\kappa} \cos \left( \frac{2\pi}{\varepsilon} \left\langle \kappa, \frac{x}{\varepsilon} \right\rangle \right) \quad \text{and} \quad b_{\varepsilon}(x) = \sum_{\kappa \in \mathbb{Z}_2} b_{\kappa} \cos \left( \frac{2\pi}{\varepsilon} \left\langle \kappa, \frac{x}{\varepsilon} \right\rangle \right) .
\]

**4.2 Full Space Problem**

Let us consider the following PDE:
\[
-\nabla \cdot (a_{\varepsilon}^{\text{per}} \nabla u) + b_{\varepsilon}^{\text{per}} u = f
\]  
(4.3)
We can rewrite (4.3) in the weak-formulation as follows: Find $u_{\varepsilon} \in V$ (where $V$ is a suitable subspace of $H^1(\mathbb{R}^2)$) such that:
\[
\int_{\mathbb{R}^2} (a_{\varepsilon}^{\text{per}} \nabla u_{\varepsilon} \cdot \nabla \bar{v}) + b_{\varepsilon}^{\text{per}} u_{\varepsilon} \bar{v} = \int_{\mathbb{R}^2} f \bar{v} \quad \forall \bar{v} \in V.
\]  
(4.4)
The exact solution can be written in the form
\[
u_{\varepsilon}(x) = \frac{1}{2\pi} \int_{\mathbb{R}^2} \tilde{f}(t) \psi(x, \varepsilon, t) dt \quad \forall x \in \mathbb{R}^2,
\]  
(4.5)
where $\psi(x, \varepsilon, t) = e^{i \langle t, x \rangle} \phi_{\varepsilon}^{\text{per}}(x, \varepsilon, t)$ and $\tilde{f}$ is the Fourier transform of $f$. The function $\phi_{\varepsilon}^{\text{per}}$ is the $\varepsilon$-scaled periodic version of the solution of the following unit cell problem: Find $\phi(\cdot, \varepsilon, t) \in H^1(\tilde{Q})$ such that:
\[
\Phi(\phi, \nu) := \int_{\tilde{Q}} a \left\langle (i \cdot t + \varepsilon^{-1} \nabla) \phi, (i \cdot t + \varepsilon^{-1} \nabla) \bar{\nu} \right\rangle + b \phi \bar{\nu} = \int_{\tilde{Q}} \bar{\nu} \quad \forall \nu \in H^1(\tilde{Q}).
\]  
(4.6)
Here $\langle x, y \rangle = \sum_{i=1}^2 x_i y_i$, without complex conjugation.

For some vector $\xi \in \mathbb{R}^2$ we can define the right-hand side of (4.4) using the function $f(x) = e^{2\pi i \langle \xi, x \rangle}$, whose Fourier transform is given by: $\tilde{f}(t) = 2\pi \delta(t - 2\pi \xi)$, where $\delta$ is the Dirac delta function. Hence (4.5) is:
\[
u_{\varepsilon}(x) = e^{2\pi i \langle \xi, x \rangle} \phi_{\varepsilon}^{\text{per}}(x, \varepsilon, 2\pi \xi).
\]  
(4.7)

**Remark 4.2.1:**
For the computational part we just consider the real-valued case ($f : \mathbb{R}^2 \to \mathbb{R}$) (see Remark 4.1.2), and we therefore define the right-hand side as $f(x) = \cos 2\pi \langle \xi, x \rangle$. 
4.3 Numerical Solution of the Unit Cell Problem

To solve (4.6) we employ a Galerkin discretization: we approximate the solution with a linear combination of basis functions:

\[ \phi_\approx \phi_p = \sum_{\kappa \in T_p} \phi_{\kappa,p} \varphi_{\kappa}. \quad (4.8) \]

For this purpose, let us define again \( T_p := \{ \kappa = (k_1, k_2) \in \mathbb{Z}^2 : -p \leq k_1, k_2 \leq p \} \subset \mathbb{Z}^2 \), for a small, fixed \( p \in \mathbb{N}_0 \). As basis functions we use the plane waves \( \varphi_{\kappa}(x) := e^{2\pi i (\kappa \cdot x)} \) and we get the space

\[ T_p := \text{span} \{ \varphi_{\kappa} : \kappa \in T_p \}. \quad (4.9) \]

Using (4.8) we can approximate (4.6) over (4.9). It follows that the approximate problem is:

Find \( \phi_p(\cdot, \varepsilon, t) \in T_p \) such that:

\[ \Phi_\varepsilon (\phi_p, v) = \int_{\mathcal{Q}} \bar{v} \right. \forall v \in T_p \quad (4.10) \]

Recall that we only have to solve this equation for \( t = 2\pi \varepsilon \).

Applying the representation (4.8) for problem (4.10) we get a linear system of equations \( (C_p \phi = r) \) for the unknown coefficient vector \( (\phi_{\kappa,p})_{\kappa \in T_p} \). To determine the right-hand side \( r = (r_{\lambda})_{\lambda \in T_p} \), observe that

\[ r_{\lambda} = \int_{\mathcal{Q}} e^{-2\pi (\lambda \cdot x)} dx = \delta_{(0,0), \lambda}. \]

The stiffness Matrix \( C_p = (c_{\lambda,\kappa})_{\lambda,\kappa \in T_p} \) can be calculated as follows

\[ c_{\lambda,\kappa} = 4\pi^2 a_{\lambda-\kappa} \left( \varepsilon + \frac{\kappa}{\varepsilon}, \varepsilon + \frac{\lambda}{\varepsilon} \right) + b_{\lambda-\kappa} \]

Consequently the coefficient vector \( \phi_{\kappa,p} \) is then determined as the solution of

\[ \sum_{\kappa \in T_p} c_{\lambda,\kappa} \phi_{\kappa,p} = r_{\lambda} \quad \forall \lambda \in T_p. \quad (4.11) \]

Finally we can approximate the solution of the full space problem (4.4) with

\[ u_\varepsilon(x) \approx u_{\varepsilon,p}(x) = e^{2\pi i (\xi \cdot x)} \left( \sum_{\kappa \in T_p} \phi_{\kappa,p} e^{2\pi i (\kappa \cdot x)} \right)_{\varepsilon}^{\text{per}} = \sum_{\kappa \in T_p} \phi_{\kappa,p} e^{2\pi i (\xi + \frac{\kappa}{\varepsilon} \cdot x)} \quad (4.12) \]

**Remark 4.3.1:**

Representation (4.12) shows clearly the two scale behavior of the solution: while the term \( e^{2\pi i (\xi \cdot x)} \) (for moderate \( \xi \)) slowly varies, the derivatives of the function \( (\phi_{\kappa,p})^{\text{per}}_{\varepsilon} \) oscillate because of the composition of \( \phi_{p} \) with the affine transformation \( \chi_{\varepsilon,\mu} \).

**Remark 4.3.2:**

The parameter \( p \) is said to be small and fixed, but from the theory of Galerkin discretization it is clear that \( \lim_{p \to \infty} u_{\varepsilon,p} = u_\varepsilon \).
Remark 4.3.3:
For our computation, we again consider the real-valued case, so the plane waves are given by:
\[ \varphi_\kappa(x) = \cos 2\pi \langle \kappa, x \rangle \]
and thus the solution reads:
\[ u_{\varepsilon,p}(x) = \sum_{\kappa \in T_p} \phi_{\kappa,p} \cos \left( 2\pi \left( \xi + \frac{\kappa}{\varepsilon}, x \right) \right). \tag{4.13} \]

4.4 Dirichlet Problem on a Finite Domain

In this section we formulate the analogous problem on a finite domain. With this assumption we get a PDE provided with non-homogeneous Dirichlet boundary conditions and thus we can test the different methods introduced in the last chapter.

Let us choose the parameter \( \xi \in \mathbb{R}^2, p \in \mathbb{N}_0 \) and \( \varepsilon = \frac{1}{N} \) for \( N \in \mathbb{N}_{>0} \). Recall that \( T_p := \{ \kappa = (k_1, k_2) \in \mathbb{Z}^2 : -p \leq k_1, k_2 \leq p \} \subset \mathbb{Z}^2 \).

Now consider the non-homogeneous Dirichlet problem: Find \( u \in H^1(\Omega) \) such that:
\[
-\nabla \cdot (a_{\varepsilon}^{\text{per}} \nabla u) + b_{\varepsilon}^{\text{per}} u = f \quad \text{in } \Omega = (0, 1)^2, \]
\[ u = u_{\varepsilon,p} \quad \text{on } \partial \Omega, \tag{4.14} \]
where
\[ f(x) := \cos(2\pi \langle \xi, x \rangle), \quad u_{\varepsilon,p} := \sum_{\kappa \in T_p} \phi_{\kappa,p} \cos \left( 2\pi \left( \xi + \frac{\kappa}{\varepsilon}, x \right) \right). \]

The coefficient vector \( \phi_p = (\phi_{\kappa,p})_{\kappa \in T_p} \) is calculated as the solution of the following linear system of equations (unit cell problem from (4.10)):
\[ \sum_{\kappa \in T_p} c_{\lambda,\kappa} \phi_{\kappa,p} = r_\lambda \quad \forall \lambda \in T_p, \tag{4.15} \]
where
\[ c_{\lambda,\kappa} = 4\pi^2 a_{\lambda-\kappa} \langle \xi + \frac{\kappa}{\varepsilon}, \xi + \frac{\lambda}{\varepsilon} \rangle + b_{\lambda-\kappa}, \quad r_\lambda = \delta_{(0,0),\lambda}. \]

The parameter determining the coefficients \( a_\kappa, b_\kappa \) can be fixed; for this computation we use:
\[ a_\kappa = \begin{cases} 10 & \text{if } \kappa = (0,0) \\ 1 & \text{if } \max(|\kappa_1|, |\kappa_2|) = 1 \\ 0 & \text{otherwise} \end{cases} \quad b_\kappa = \begin{cases} 1 & \text{if } \kappa = (0,0) \\ 0 & \text{otherwise} \end{cases} \]
i.e.:
\[ a_{\varepsilon}^{\text{per}}(x) = 2 \left( 5 + \cos \frac{2\pi x_1}{\varepsilon} + \cos \frac{2\pi x_2}{\varepsilon} + \cos \frac{2\pi (x_1 + x_2)}{\varepsilon} + \cos \frac{2\pi (x_1 - x_2)}{\varepsilon} \right) \]
and \( b_{\varepsilon}^{\text{per}}(x) = 1. \tag{4.16} \)

Remark 4.4.1:
Clearly \( a_{\varepsilon}^{\text{per}} \) and \( b_{\varepsilon}^{\text{per}} \) are bounded according to (4.1), thus the problem has a unique solution (Lax-Milgram Theorem).
4.5 Implementation

In this particular model problem, two combined implementations are involved: first we have to solve the Unit Cell Problem for a given $p$, then we can use the information gained to construct the PDE with non-homogeneous Dirichlet boundary conditions and solve the finite space problem (4.14) using FEM. To perform these tasks we employ two separate tools: first we implement an algorithm (using Matlab) to approximate the solution of the unit cell problem, then we import the calculated coefficient vector $\phi_{p,p}$ into NGsolve and solve the finite space problem using the algorithms for non-homogeneous problems introduced in the last chapter. Note that it is not possible to directly solve both parts using NGsolve, which is why we use an additional program.

4.5.1 Unit Cell Problem Using Matlab

To solve the Unit Cell Problem we can use the following code files:

```matlab
function [ PHI ] = phifunc( E, eps , P )
    format long;
    p=P:P;

    %Creating the 2D-array with the values of kappa and lambda
    [X,Y]=ndgrid(p,p);
    k=zeros((2*P+1)^2,2);
    for i=1:(2*P+1)^2
        k(i,1)=X(floor(((i-1)/(2*P+1)))+1,mod(i-1,2*P+1)+1);
        k(i,2)=Y(floor(((i-1)/(2*P+1)))+1,mod(i-1,2*P+1)+1);
    end
    lambda=k;
    l=size(k,1);

    %Evaluating the coefficient of the matrix C
    C=zeros(l);
    for i=1:l
        C(i,:)=coefficient(E,eps,lambda(i,:),k);
    end

    %Creating the vector r (right-hand side of the equation)
    r=zeros((2*P+1)^2,1);
    r(floor(((2*P+1)^2)/2)+1,1)=1;

    %Solving the linear problem with the build-in algorithm
    PHI=C\r;

Listing 4.1: function phifunc.m
```

```matlab
function [ c ] = coefficient( E , eps , lambda , k )
    l=size(k,1);
    c=zeros(1,l);
    for i=1:l
        c(i)=4*pi*pi*fc(a(lambda-k(i,:),eps)) dot(E+k(i,:),eps,E+lambda/eps);
        if lambda-k(i,:) == [0 0]
            c(i)=c(i)+1;
        end
    end
end
end
end

Listing 4.2: function coefficient.m
```

```matlab
function [ out ] = fcta( in , eps )
    out=0;
end
```

Listing 4.3: function fcta.m
Listing 4.3: function fcta.m

The function phifunc.m in listing 4.1 returns the coefficient vector \( \phi_{\kappa, p} \) for given coefficients \( \xi, \epsilon, p \) (\( E, \text{eps} \) and \( P \) in the code). To complete this task, the function calls the additional functions coefficient.m (listing 4.2) that calculates the values of the vector \( C_p \) (part of the system matrix) and the function fcta.m (listing 4.3), that returns the value of the coefficient \( a_\kappa \).

**Choice of \( p \)**

One of the first issues relative to this computation is the choice of \( p \). As previously mentioned in Remark 4.3.2, by increasing \( p \) we decrease the offset between the solution \( u_\epsilon \) and \( u_{\epsilon, p} \): for a larger \( p \), the PDE determining \( u_{\epsilon, p} \) has refined non-homogeneous Dirichlet boundary conditions that lead to a solution \( u_{\epsilon, p} \) closer to the original solution \( u_\epsilon \).

On the other hand a large \( p \) is disadvantageous: the cost of inverting the Matrix \( C_p \) of size \((2p+1)^2 \times (2p+1)^2\) grows very fast, as well as the size of the vector determining the boundary condition \((|\phi_p| = (2p+1)^2)\). That is why \( p \) is chosen to be small.

In conclusion, since we are specifically interested in the performance of the FEM solver, we just need to fix \( p \) as small as possible, but providing an appropriate approximation of the boundary conditions (and thus of the exact solution) such that the behavior of the FEM error is still dominant. In other terms: we can describe the error \( e \) for a given norm \( \| \cdot \| \) as:

\[
e = \| u_{\epsilon, p, h} - u_\epsilon \| = \| u_{\epsilon, p, h} - u_{\epsilon, p} + u_{\epsilon, p} - u_\epsilon \| \leq \| u_{\epsilon, p, h} - u_{\epsilon, p} \| + \| u_{\epsilon, p} - u_\epsilon \| = e_{\text{FEM}} + e_p.
\]

where \( u_\epsilon \) is the exact solution of (4.14), \( u_{\epsilon, p} \) is the approximate solution for a given \( p \) using the Galerkin discretization introduced in Section 4.3 and \( u_{\epsilon, p, h} \) is the approximated solution using the FEM. Since we can only calculate the error \( e_{\text{FEM}} \), we need \( e_p \) to be negligible (such that \( e \approx e_{\text{FEM}} \)).

One method to investigate the error \( e_p \) is to calculate at each step \( p \to p + 1 \) by how much the solution is improved: \( \| u_{\epsilon, p+1} - u_{\epsilon, p} \| \). In Figure 4.1 the difference \( \delta = \| u_{\epsilon, p+1} - u_{\epsilon, p} \|_{L^2} \) using the \( L^2 \)-norm (in linear and logarithmic scale) is displayed. It can be seen that from \( p = 4 \) to \( p = 5 \) the solution only changes by a magnitude of about \( 10^{-8} \). In most of our FEM approximations we reach a refinement of up to \( 10^{-10} \). It follows that by using \( p = 10 \) we can ensure that the error \( e_p \) is negligible.

### 4.5.2 Implementation Using NetGen/NGSolve

Once the appropriate \( p \) is determined, the unit cell problem is solved and the coefficient \( \phi_{\kappa, p} \) are calculated, we can write the NGSolve file describing problem (4.14). Again we implement the two methods described in the last chapter: solution split (over domain/boundary)
4.6 Numerical Results

In this section we present the results found with the numerical computations. In particular, we are interested in investigations of the error behavior and the time performance.

As previously mentioned we set \( p = 10 \). We moreover set \( \varepsilon = \frac{1}{N} = \frac{1}{10} \) and \( \xi = [1, 1]^T \).

The solutions found by the three possible approaches nearly coincide and they converge to the reference solution \( u_{\varepsilon,p} \) for \( h \rightarrow 0 \). One example of the solution is given in Figure 4.2.

### 4.6.1 Error Behavior

Within the theoretical part, it was highlighted how the error estimation is dominated by the error given by the interpolation in the finite element space, in particular it was proven that, for \( r = 0, 1 \):

\[
\|u - u_h^k\|_{H^r} \leq Ch^{k+1-r}\|u\|_{H^{k+1}}
\]  

(4.18)

where \( u \) is the exact solution and \( u^k \) is the finite element approximation.

In the last chapter we found that this estimate holds for simple examples and it is now of interest to check if it also holds for a more complex example. Again we can solve the PDE for consecutive refinement steps \( s \) and at each step we refine the mesh using a bisection algorithm. Using the particular mesh for the domain \( \Omega = (0,1)^2 \) depicted in Figure 3.3, the bisection algorithm halve the size \( h = \max h_s \) of the mesh at each refinement step. According to (4.18) we expect that the error \( e_{H^r}^s = \|u_{\varepsilon,p,h} - u_{\varepsilon,p}\|_{L^2} \) will decrease with a factor \( h^{k+1-r} = \frac{1}{2^{k+1-r}} \)

---

\(^1\)http://www.math.uzh.ch/compmath/index.php?id=dipl
CHAPTER 4. MODEL PROBLEM 2

(a) Solution $u_{e,p,h}$

(b) Flux of the solution

Figure 4.2: Solution and flux of the solution of problem (4.14) using linear elements and Solution Split over Domain. The behavior is as stated in Remark 4.3.1

or, in logarithmic scale, with a slope of $-(k + 1 - r)$, where $k$ is the polynomial order of the elements.

Remark 4.6.1:
As stated in section 4.5.1, we cannot use the exact solution $u_\varepsilon$ as reference solution. Using a large enough $p$ (in this case $p = 10$) we can still use $u_{e,p}$ as reference solution, ensuring enough accuracy.

From Tables 4.1-4.6 we can clearly see that, after some iteration steps, the behavior of the error follows what stated in (4.18) for both methods: solution split (just on the boundary and also with smooth continuation) and approximation with the Robin boundary conditions.

For our particular case (with $\varepsilon = \frac{1}{10}$), we notice that the different algorithms need some refinement steps to reach the expected convergence rate: which is achieved once the mesh size is of order $10^{-2}$ and can thus ensure the appropriate approximation of the rapid oscillation of the solution’s derivative: this oscillation is determined by the factor $\varepsilon$ defining the problem (see Remark 4.3.1). It could be interesting to investigate how the choice of $\varepsilon$ affects the minimal mesh-size needed to ensure the expected convergence rate.
4.6. NUMERICAL RESULTS

<table>
<thead>
<tr>
<th>step s</th>
<th>(\sqrt{2}/h)</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.001293</td>
<td>-</td>
<td>0.000225</td>
<td>-</td>
<td>0.000174</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.000152</td>
<td>8.508</td>
<td>0.000120</td>
<td>1.860</td>
<td>6.46E-05</td>
<td>2.690</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1.55e-04</td>
<td>0.981</td>
<td>6.51e-05</td>
<td>1.854</td>
<td>2.21e-05</td>
<td>2.924</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>6.96e-05</td>
<td>2.224</td>
<td>1.63e-05</td>
<td>4.008</td>
<td>4.23e-06</td>
<td>5.217</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.95e-05</td>
<td>3.563</td>
<td>3.48e-06</td>
<td>4.667</td>
<td>4.54e-07</td>
<td>9.326</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>5.21e-06</td>
<td>3.750</td>
<td>3.65e-07</td>
<td>9.537</td>
<td>3.98e-08</td>
<td>11.417</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>1.23e-06</td>
<td>4.221</td>
<td>4.90e-08</td>
<td>7.445</td>
<td>1.854</td>
<td>6.51e-05</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>3.07e-07</td>
<td>4.018</td>
<td>6.18e-09</td>
<td>7.930</td>
<td>1.42e-10</td>
<td>16.149</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>7.66e-08</td>
<td>4.010</td>
<td>7.78e-10</td>
<td>7.946</td>
<td>9.16e-12</td>
<td>15.488</td>
</tr>
</tbody>
</table>

Table 4.1: Error \(|u_{\varepsilon,h,p} - u_{\varepsilon,p}|_{L^2}\) using the solution split method over domain

<table>
<thead>
<tr>
<th>step s</th>
<th>(\sqrt{2}/h)</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.001591</td>
<td>-</td>
<td>0.000402</td>
<td>-</td>
<td>0.000229</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.000150</td>
<td>10.579</td>
<td>0.000111</td>
<td>3.618</td>
<td>6.76e-05</td>
<td>3.385</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1.76e-04</td>
<td>0.857</td>
<td>5.22e-05</td>
<td>2.131</td>
<td>2.06e-05</td>
<td>3.281</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>6.66e-05</td>
<td>2.636</td>
<td>1.67e-05</td>
<td>3.123</td>
<td>4.42e-06</td>
<td>4.666</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.89e-05</td>
<td>3.520</td>
<td>3.54e-06</td>
<td>4.717</td>
<td>4.41e-07</td>
<td>10.019</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>5.08e-06</td>
<td>3.725</td>
<td>3.76e-07</td>
<td>9.407</td>
<td>3.94e-08</td>
<td>11.202</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>1.22e-06</td>
<td>4.174</td>
<td>4.98e-08</td>
<td>7.560</td>
<td>2.29e-09</td>
<td>17.216</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>3.05e-07</td>
<td>3.997</td>
<td>6.21e-09</td>
<td>8.023</td>
<td>1.42e-10</td>
<td>16.123</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>7.62e-08</td>
<td>3.999</td>
<td>7.76e-10</td>
<td>7.996</td>
<td>9.16e-12</td>
<td>15.501</td>
</tr>
</tbody>
</table>

Table 4.2: Error \(|u_{\varepsilon,h,p} - u_{\varepsilon,p}|_{L^2}\) using the solution split method over boundary

<table>
<thead>
<tr>
<th>step s</th>
<th>(\sqrt{2}/h)</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.001591</td>
<td>-</td>
<td>0.000402</td>
<td>-</td>
<td>0.000229</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.000150</td>
<td>10.579</td>
<td>0.000111</td>
<td>3.618</td>
<td>6.76e-05</td>
<td>3.385</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1.76e-04</td>
<td>0.857</td>
<td>5.22e-05</td>
<td>2.131</td>
<td>2.06e-05</td>
<td>3.281</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>6.66e-05</td>
<td>2.636</td>
<td>1.67e-05</td>
<td>3.123</td>
<td>4.42e-06</td>
<td>4.666</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.89e-05</td>
<td>3.520</td>
<td>3.54e-06</td>
<td>4.717</td>
<td>4.41e-07</td>
<td>10.019</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>5.08e-06</td>
<td>3.725</td>
<td>3.76e-07</td>
<td>9.407</td>
<td>3.94e-08</td>
<td>11.202</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>1.22e-06</td>
<td>4.174</td>
<td>4.98e-08</td>
<td>7.560</td>
<td>2.29e-09</td>
<td>17.216</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>3.05e-07</td>
<td>3.997</td>
<td>6.21e-09</td>
<td>8.023</td>
<td>1.42e-10</td>
<td>16.123</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>7.62e-08</td>
<td>3.999</td>
<td>7.76e-10</td>
<td>7.996</td>
<td>9.16e-12</td>
<td>15.501</td>
</tr>
</tbody>
</table>

Table 4.3: Error \(|u_{\varepsilon,h,p} - u_{\varepsilon,p}|_{L^2}\) using the Robin boundary conditions approximation method
### Table 4.4: Error $|u_{e,h,p} - u_{e,p}|_{H^1}$ using the solution split method over domain

<table>
<thead>
<tr>
<th>step s</th>
<th>$\sqrt{2}/h$</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.010392</td>
<td>-</td>
<td>0.00574</td>
<td>-</td>
<td>0.00462</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.00405</td>
<td>2.568</td>
<td>0.00275</td>
<td>2.085</td>
<td>2.18e-03</td>
<td>2.120</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>3.58e-03</td>
<td>1.292</td>
<td>2.10e-03</td>
<td>1.313</td>
<td>1.38e-03</td>
<td>1.576</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>1.97e-03</td>
<td>1.819</td>
<td>1.27e-03</td>
<td>1.655</td>
<td>5.80e-04</td>
<td>2.385</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.35e-03</td>
<td>1.456</td>
<td>4.78e-04</td>
<td>2.646</td>
<td>1.30e-04</td>
<td>4.469</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>7.37e-04</td>
<td>1.836</td>
<td>1.30e-04</td>
<td>3.677</td>
<td>2.31e-05</td>
<td>5.627</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>3.70e-04</td>
<td>1.992</td>
<td>3.59e-05</td>
<td>3.628</td>
<td>2.77e-06</td>
<td>8.324</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>1.86e-04</td>
<td>1.991</td>
<td>9.10e-06</td>
<td>3.943</td>
<td>3.49e-07</td>
<td>7.936</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>9.30e-05</td>
<td>1.998</td>
<td>2.25e-06</td>
<td>4.041</td>
<td>4.43e-08</td>
<td>7.876</td>
</tr>
</tbody>
</table>

### Table 4.5: Error $|u_{e,h,p} - u_{e,p}|_{H^1}$ using the solution split method over boundary

<table>
<thead>
<tr>
<th>step s</th>
<th>$\sqrt{2}/h$</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.01141</td>
<td>-</td>
<td>0.00562</td>
<td>-</td>
<td>0.00456</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.00397</td>
<td>2.875</td>
<td>0.00281</td>
<td>1.999</td>
<td>2.09e-03</td>
<td>2.181</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>3.63e-03</td>
<td>1.092</td>
<td>2.04e-03</td>
<td>1.377</td>
<td>1.39e-03</td>
<td>1.506</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>1.97e-03</td>
<td>1.845</td>
<td>1.26e-03</td>
<td>1.620</td>
<td>5.83e-04</td>
<td>2.378</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.36e-03</td>
<td>1.447</td>
<td>4.79e-04</td>
<td>2.631</td>
<td>1.29e-04</td>
<td>4.516</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>7.40e-04</td>
<td>1.838</td>
<td>1.30e-04</td>
<td>3.678</td>
<td>2.31e-05</td>
<td>5.595</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>3.70e-04</td>
<td>1.999</td>
<td>3.59e-05</td>
<td>3.628</td>
<td>2.77e-06</td>
<td>8.325</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>1.86e-04</td>
<td>1.991</td>
<td>9.16e-06</td>
<td>3.919</td>
<td>3.49e-07</td>
<td>7.944</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>9.30e-05</td>
<td>1.998</td>
<td>2.26e-06</td>
<td>4.047</td>
<td>4.41e-08</td>
<td>7.914</td>
</tr>
</tbody>
</table>

### Table 4.6: Error $|u_{e,h,p} - u_{e,p}|_{H^1}$ using the Robin boundary conditions approximation

<table>
<thead>
<tr>
<th>step s</th>
<th>$\sqrt{2}/h$</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
<th>error</th>
<th>factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.01141</td>
<td>-</td>
<td>0.00562</td>
<td>-</td>
<td>0.00456</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.00401</td>
<td>2.843</td>
<td>0.00282</td>
<td>1.992</td>
<td>2.09e-03</td>
<td>2.181</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>3.65e-03</td>
<td>1.100</td>
<td>1.97e-03</td>
<td>1.433</td>
<td>1.39e-03</td>
<td>1.506</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>1.97e-03</td>
<td>1.853</td>
<td>1.26e-03</td>
<td>1.557</td>
<td>5.83e-04</td>
<td>2.378</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.37e-03</td>
<td>1.437</td>
<td>4.78e-04</td>
<td>2.649</td>
<td>1.29e-04</td>
<td>4.516</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>7.41e-04</td>
<td>1.846</td>
<td>1.30e-04</td>
<td>3.663</td>
<td>2.31e-05</td>
<td>5.595</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>3.70e-04</td>
<td>2.002</td>
<td>3.59e-05</td>
<td>3.631</td>
<td>2.77e-06</td>
<td>8.325</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>1.86e-04</td>
<td>1.992</td>
<td>9.10e-06</td>
<td>3.944</td>
<td>3.49e-07</td>
<td>7.944</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>9.30e-05</td>
<td>1.998</td>
<td>2.25e-06</td>
<td>4.053</td>
<td>4.41e-08</td>
<td>7.914</td>
</tr>
</tbody>
</table>
4.6. NUMERICAL RESULTS

Figure 4.3: Logarithm of the error for the $L^2$ norm for some degrees $k$.

Figure 4.4: Logarithm of the error for the $H^1$ seminorm for some degrees $k$. 
4.6.2 Time Performance

As in the last chapter it is of interest to verify the time needed by each approach to solve the non-homogeneous problem at each refinement step. We expect, as proven in theory and as shown in the simpler examples, that the performance evolves with order $O(N^{3/2})$. Since at every refinement step the amount of unknowns or degrees of freedom ($N$) quadruplicate, the expected time cost should evolve with a factor $4^{3/2} = 8$.

In Tables (4.7), (4.8) and (4.9) the expected order is visible: again we see that NGSolve is faster than expected because its solver is specifically built for sparse matrices and exploits a preconditioner.

The computing times for small $N$ are not significant because the time elapsed is too small to be meaningful in comparison.

<table>
<thead>
<tr>
<th>step $s$</th>
<th>$N$</th>
<th>Robin approximation</th>
<th>Solution Split over $\partial\Omega$</th>
<th>Solution split over $\Omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>time</td>
<td>factor</td>
<td>time</td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>0.02</td>
<td>-</td>
<td>0.02</td>
</tr>
<tr>
<td>2</td>
<td>81</td>
<td>0.03</td>
<td>1.500</td>
<td>0.03</td>
</tr>
<tr>
<td>3</td>
<td>289</td>
<td>0.04</td>
<td>1.333</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>1089</td>
<td>0.1</td>
<td>2.500</td>
<td>0.1</td>
</tr>
<tr>
<td>5</td>
<td>4225</td>
<td>0.24</td>
<td>2.400</td>
<td>0.22</td>
</tr>
<tr>
<td>6</td>
<td>16641</td>
<td>0.99</td>
<td>4.125</td>
<td>0.95</td>
</tr>
<tr>
<td>7</td>
<td>66049</td>
<td>3.59</td>
<td>3.626</td>
<td>3.52</td>
</tr>
<tr>
<td>8</td>
<td>263169</td>
<td>14.73</td>
<td>4.103</td>
<td>14.58</td>
</tr>
<tr>
<td>9</td>
<td>1050625</td>
<td>63.4</td>
<td>4.304</td>
<td>61.16</td>
</tr>
</tbody>
</table>

Table 4.7: Elapsed time (in seconds) for the three methods for $k = 1$

<table>
<thead>
<tr>
<th>step $s$</th>
<th>$N$</th>
<th>Robin approximation</th>
<th>Solution Split over $\partial\Omega$</th>
<th>Solution split over $\Omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>time</td>
<td>factor</td>
<td>time</td>
</tr>
<tr>
<td>1</td>
<td>81</td>
<td>0.03</td>
<td>-</td>
<td>0.02</td>
</tr>
<tr>
<td>2</td>
<td>289</td>
<td>0.03</td>
<td>1.000</td>
<td>0.02</td>
</tr>
<tr>
<td>3</td>
<td>1089</td>
<td>0.06</td>
<td>2.000</td>
<td>0.06</td>
</tr>
<tr>
<td>4</td>
<td>4225</td>
<td>0.18</td>
<td>3.000</td>
<td>0.17</td>
</tr>
<tr>
<td>5</td>
<td>16641</td>
<td>0.5</td>
<td>2.778</td>
<td>0.49</td>
</tr>
<tr>
<td>6</td>
<td>66049</td>
<td>1.88</td>
<td>3.760</td>
<td>1.83</td>
</tr>
<tr>
<td>7</td>
<td>263169</td>
<td>8.67</td>
<td>4.612</td>
<td>8.4</td>
</tr>
<tr>
<td>8</td>
<td>1050625</td>
<td>45.95</td>
<td>5.300</td>
<td>44.47</td>
</tr>
<tr>
<td>9</td>
<td>4198401</td>
<td>220.37</td>
<td>4.796</td>
<td>231.37</td>
</tr>
</tbody>
</table>

Table 4.8: Elapsed time (in seconds) for the three methods for $k = 2$

One interesting point is the behavior of the algorithm using the solution split over domain method: even if the evolution is still of order $O(N^{3/2})$ (or even less), this approach is up to ten times slower than the other two methods. This can be explained by considering the construction of the algorithm: while the other two approaches only evaluate integrals over the boundary nodes to deal with the non-homogeneous Dirichlet boundary conditions, the solution split over domain evaluate the $L^2$-projection over every node in the mesh, clearly slowing down the process. This behavior is also apparent in Figure 4.5.
Table 4.9: Elapsed time (in seconds) for the three methods for $k = 3$

<table>
<thead>
<tr>
<th>step $s$</th>
<th>$N$</th>
<th>time</th>
<th>factor</th>
<th>$N$</th>
<th>time</th>
<th>factor</th>
<th>$N$</th>
<th>time</th>
<th>factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>169</td>
<td>0.01</td>
<td>-</td>
<td>0.03</td>
<td>-</td>
<td>0.05</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>625</td>
<td>0.04</td>
<td>4.000</td>
<td>0.07</td>
<td>2.333</td>
<td>0.19</td>
<td>3.800</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2401</td>
<td>0.08</td>
<td>2.000</td>
<td>0.07</td>
<td>2.333</td>
<td>0.63</td>
<td>3.316</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>9409</td>
<td>0.22</td>
<td>2.750</td>
<td>0.21</td>
<td>3.000</td>
<td>2.33</td>
<td>3.698</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>37249</td>
<td>0.72</td>
<td>3.273</td>
<td>0.71</td>
<td>3.381</td>
<td>9.25</td>
<td>3.970</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>148225</td>
<td>3.1</td>
<td>4.306</td>
<td>2.96</td>
<td>4.169</td>
<td>37.54</td>
<td>4.058</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>591361</td>
<td>16.2</td>
<td>5.226</td>
<td>15.84</td>
<td>5.351</td>
<td>154.01</td>
<td>4.103</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>2362369</td>
<td>103.18</td>
<td>6.369</td>
<td>101.75</td>
<td>6.424</td>
<td>662.82</td>
<td>4.304</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>9443329</td>
<td>670.35</td>
<td>6.497</td>
<td>619.7</td>
<td>6.090</td>
<td>2857.2</td>
<td>4.311</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 4.5: Logarithm of the elapsed time with respect to the degrees of freedom $N$ (also in logarithmic scale)
Chapter 5

Conclusion

In the current work we introduced two methods to solve elliptic PDE with non-homogeneous Dirichlet boundary conditions: the Solution Split method (Solution Split over domain or over boundary) and the approximation by Robin boundary conditions. We were able to prove the following properties mathematically and their sharpness by numerical tests.

(1) **Error bounds:** the approximation error of both methods is dominated by the interpolation error in the finite element space. Since this space is the same for both methods we can conclude that the error bounds given by

\[ ||u - u^h||_{H^r(\Omega)} \leq Ch^{k+1-r}||u||_{H^{k+1}(\Omega)} \]

for \( r = 0, 1 \) hold for both methods. For the Robin boundary conditions approximation (penalty method) we have to select a penalty factor \( \varepsilon \leq h^{k+1} \) to ensure this rate of convergence.

(2) **Time complexity:** it is not straightforward to distinguish the two methods since the time complexity is dominated by the resolution of the linear system of equations and it evolves with order \( \mathcal{O}(N^{3/2}) \) with respect to the amount of nodes \( N \) (using the CG-algorithm).

The only major difference is that the Solution Split over Domain method is significantly slower than the other methods. This is because further integrations over the whole domain are needed. Despite this the order remains unchanged.

There are further interesting points which could be investigated in new studies. For example the behavior of the same methods in higher dimensional problems could be tested: this could be done, for three dimensional problems, using NetGen/NGSolve.

Another interesting point are the time-dependent problems: the reliability of the methods when the problem is of the non-stationary type and the boundary conditions are given as functions with respect to the time variable could be investigated.

A third possible research could be the introduction of variational crimes: for this thesis we just considered polyhedral domains and thus the triangulation coincided exactly with the original domain. In more general problems the boundary can be given by a smooth curve, which requires curved elements at the boundary or the domain with the triangulation \( \Omega^h \approx \Omega \) would need to be appropriately approximated. In [BE86, SF08] such techniques are discussed and it could be of interest to test those methods.
Bibliography


Appendix

# Solving \(-\text{laplace}(u) = f\) and homogeneous Dirichlet boundary condition

# Solution \(u = \sin(\pi x) \sin(\pi y)\)

# Loading geometry and mesh

gamey = square.in2d
mesh = square.vol

# Resolution of the problem

define constant pi = 3.1415926535897932384626

# Laplace coefficient
define coefficient one 1,

# Source integral \(f\)
define coefficient f 
\((2*\pi*\pi*\sin(\pi*x)*\sin(\pi*y))\),

# Creating FE-space
define fespace \(v\) -order=1 -dirichlet=[1]
define gridfunction \(u\) -fespace=v -nested

# Bilinear form
define bilinearform \(a\) -fespace=v -symmetric
laplace one

# Linear form
define linearform \(l\) -fespace=v
source \(f\)

# Numerical solution
define preconditioner \(c\) -type=direct -bilinearform=a
numproc bvp solve_dirichlet_bc -bilinearform=a -linearform=l -gridfunction=u -
preconditioner=c -maxsteps=100

# Error evaluation

# FE-space for error - gridfunctions
define fespace \(v_{err}\) -l2 -order=0

# Coefficient for \(u_{ref}\) and \(\text{grad}_{u_{ref}}\)
define coefficient \(u_{ref}\) 
\((\sin(\pi*x)\sin(\pi*y))\),
define coefficient \(\text{grad}_{u_{ref}}\) 
\((\pi*x\cos(\pi*x)\sin(\pi*y)\),\pi*\sin(\pi*x)\cos(\pi*y))\),

# Scalar product for L2-norm
define bilinearform \(a_{l2}\) -fespace=v -nonassemble
mass one

# Scalar product for H1-seminorm
define bilinearform \(a_{h1}\) -fespace=v -nonassemble
laplace one

# Gridfunctions for error
define gridfunction \(\text{err}_{l2}\) -fespace=verr
define gridfunction \(\text{err}_{h1}\) -fespace=verr
# Evaluating the errors
numproc difference l2error -bilinearform1=a12 -solution=u -function=uref -diff=errl2
numproc difference h1error -bilinearform1=a1h1 -solution=u -function=graduref -diff=errh1

Listing 1: homogeneous.pde

# Model problem 1 using Solution Split
# Loading geometry and mesh
geometry = square.in2d
mesh = square.vol

# Resolution of the problem
define constant pi = 3.1415926535897932384626

# Laplace coefficient
define coefficient one 1,

# Source integral (f)
define coefficient f ((4)/(x+y+1)*(x+y+1)*(x+y+1))+2*pi*pi*sin(pi*x)*sin(pi*y),

# Dirichlet boundary condition
define coefficient dirichlet_bc (1/(x+y+1)),(1/(x+y+1)),(1/(x+y+1)),(1/(x+y+1)),

# Creating FE-space
define fespace v -order=1 -dirichlet=[1]
define gridfunction u -fespace=v -nested

define bilinearform a -fespace=v -symmetric
define linearform l -fespace=v

# Solution Split procedure
numproc setvalues upload_dirc -gridfunction=u -coefficient=dirichlet_bc -boundary

# Numerical solution
define preconditioner c -type=direct -bilinearform=a
numproc bvp solve_dirichlet_bc -bilinearform=a -linearform=l -gridfunction=u -preconditioner=c -maxsteps=100

# Error evaluation
# FE-space for error-gridfunctions
define fespace verr -12 -order=0

# Coefficient for uref and graduref
define coefficient uref (sin(pi*x)*sin(pi*y)+1/(x+y+1)),
define coefficient graduref (pi*cos(pi*x)*sin(pi*y)-1/((x+y+1)*(x+y+1)), pi*sin(pi*x)*cos(pi*y)-1/((x+y+1)*(x+y+1)));

# Scalarproduct for L2-norm
define bilinearform al2 -fespace=v -nonassemble

# Scalarproduct for H1-seminorm
define bilinearform ahl -fespace=v -nonassemble

define gridfunction errl2 -fespace=verr
define gridfunction errh1 -fespace=verr
# Evaluating errors
numproc difference l2error -bilinearform1=a l2 -solution=u -function=uref -diff=errl2
numproc difference l1error -bilinearform1=ah l1 -solution=u -function=graduref -diff=errh1

Listing 2: nonhomogeneous_solsplit.pde

# Model problem 1 using Solution Split
# Loading geometry and mesh
g = square.in2d
mesh = square.vol

define constant pi = 3.1415926535897932384626
define constant penal = 1e15

# Resolution of the problem ######
# Laplace coefficient
define coefficient one 1,
def source integral (f)
define coefficient f
((-4)/((x+y+1)+(x+y+1)))+2*pi*pi*sin(pi*x)*sin(pi*y),
define coefficient lhs (penal),
def source rhs (penal*((1/(x+y+1))))),

# Creating FE-space
def fespace v -order=1
def grid function u -fespace=v -nested

# Bilinear form
define bilinearform a -fespace=v -symmetric
laplace one
robin lhs

# Linear form
def linearmform l -fespace=v
source f
neumann rhs

# Numerical solution
define preconditioner c -type=direct -bilinearform=a
numproc bpv solve_dirichlet bc -bilinearform=a -linearmform=l -gridfunction=u -
preconditioner=c -maxsteps=100

# Error evaluation #######
# FE-space for error-gridfunctions
def fespace verr l2 -order=0

# Coefficient for uref and graduref
define coefficient uref
(sin(pi*x)*sin(pi+y)+(x+y+1)),
define coefficient graduref
(pi*cos(pi*x)*sin(pi+y)/(x+y+1) , pi*sin(pi*x)*cos(pi*y)-1/(x+y+1) ),

# Scalar product for L2-norm
define bilinearform al2 -fespace=v -nonassemble
mass one
# Scalar product for H1-seminorm
define bilinearform ah l -fespace=v -nonassemble
laplace one
# Gridfunctions for error
define gridfunction errl2 -fespace=verr
define gridfunction errh1 -fespace=verr

# Evaluating errors
numproc difference l2error -bilinearform1=a12 -solution=u -function=uref -diff=errl2
numproc difference h1error -bilinearform1=ah1 -solution=u -function=graduref -diff=errh1

Listing 3: nonhomogeneous_robin.pde