Multigrid smoothers for saddle point systems

Dissertation

Bergische Universität Wuppertal
Fakultät für Mathematik und Naturwissenschaften

eingereicht von
Lisa Claus, M. Sc.
zur Erlangung des Grades eines Doktors der Naturwissenschaften

Betreut durch Prof. Dr. Matthias Bolten

Juni 2019

The PhD thesis can be quoted as follows:

urn:nbn:de:hbz:468-20190723-103226-4
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20190723-103226-4]

DOI: 10.25926/msc6-eh86
[https://doi.org/10.25926/msc6-eh86]

Acknowledgments

Matthias, thank you for giving me the opportunity to work on my PhD, to dig deeper into challenging and interesting problems and to present my research at a variety of conferences. Also, I would like to thank you for always answering ALL my questions. (There have been a lot ;)) Moreover, without your support the highly instructive research stays at LLNL would not have been possible.

Rob, it's been a great experience to join you and your research group at LLNL (not only because of the extraordinary friendly and enjoyable people and the sunny California weather). It was a challenging and enjoyable time. I learned so much more than I could have imagined. Thanks for always forcing me to dive into the details and your support in growing my understanding of AMG.

Sarah & Jan, you both did an awesome job proofreading this thesis, you gave me interesting insights and thereby improved this work a lot.

Thanks to all my colleagues: Sergiy, Jens, Tim, Stephanie, Werner and Frau Birkenfeld. I enjoyed discussions, cakes and being distracted by "basketball" sessions, Minions and lightsabers.

Furthermore, thank you Scott, your LFA suggestions always pushed me in the right direction.

Thanks to my friends and family who did a great job in keeping me away from work by sport and other entertainment. Also, I am grateful for your continuous support.

Camilla, I would like to thank you not only for preparing delicious dinners, but also for being a great house mate in general and of course an enjoyable colleague. You have been a good companion during the last eight years.

George, without your support, patience and entertainment it would have been impossible for me to accomplish all this, thank you!

Contents

1 Introduction

Multigrid (MG) methods are efficient methods to solve systems of partial differential equations (PDEs). They are iterative methods that have been developed to solve scalar PDEs. The focus of this thesis is the solution of linear systems of PDEs that arise from mathematical modeling of physical systems. In particular, we are interested in the efficient solution of linear systems of equations, $A\mathbf{u} = \mathbf{f}$, that result from finite difference or finite element discretizations of PDEs. The resulting systems are large and sparse.
An iterative method is a procedure that uses an initial guess to generate a sequence of improving approximate solutions, such that a new approximation is derived from the previous one. In contrast, in the absence of rounding errors, direct methods would provide an exact solution by a finite sequence of operations. In general, iterative methods are preferred over direct methods for the solution of large linear systems as they arise from the discretization of systems of PDEs. The reason is that direct methods may be prohibitively computationally expensive. Multigrid methods in particular are known to be optimal methods. They can solve certain PDEs (to a given accuracy) in a number of operations that is proportional to the number of unknowns. This excellent property distinguishes multigrid methods from other iterative methods such as Krylov subspace methods. As a consequence of the growth of iteration costs, Krylov methods share with direct elimination methods the disadvantage that the computational time needed to solve a problem is not proportional (or nearly proportional) to the total number of degrees of freedom.
In the 1960s R.P. Fedorenko [33] developed the first multigrid method for the solution of the Poisson equation in a unit square. A. Brandt [15] and W. Hackbusch [34] were able to show the efficiency of the multigrid approach in the 1970s. Since then, other mathematicians have extended Fedorenko's idea, not only for other scalar PDEs, but also for systems of PDEs.
The growth of the computational cost for larger problems dictates the need to use parallel computing, which decreases simulation time by using additional resources simultaneously. Multigrid methods take order n of work to solve a sparse linear system with n unknowns. That gives them excellent scaling potential, i.e. the time to solution remains constant as the problem size together with the number of parallel processors is proportionally increased.
Today, multigrid methods are used in nearly every field where partial differential equations are solved by numerical methods. They have been shown to be efficient

iterative solvers for a variety of applications, such as fluid and solid mechanics. However, there is a need to examine the solutions in these fields more closely. Numerical modeling of those problems poses some difficulties which makes the development of fast and efficient linear solvers challenging.
Examples that arise in fluid and solid mechanics are the Stokes equations and Maxwell's equations. The Stokes equations or the Stokes flow is a type of fluid flow where advective inertial forces are small compared with viscous forces. This describes a typical situation in flows where the fluid velocities are very slow, the viscosities are very large, or the length-scales of the flow are very small. The constant movement of the Earth crust caused by the mantle convection is one application that can be described and simulated by means of the Stokes equations. Maxwell's equations form the foundation of classical electromagnetism and electric circuits such as power generation or electric motors. Discretizations of these equations naturally lead to so called saddle point systems, a specific type of linear systems. The name stems from the characteristics of the solution as will be described in more detail in the course of this thesis.
Multigrid methods find an approximate solution to a linear system through two complementary processes: relaxation and coarse-grid correction. Relaxation cheaply removes parts of the error from the approximate solution, while coarse-grid correction constructs a lower dimensional problem to remove the error remaining after relaxation. This thesis focuses on improving the efficiency of the relaxation process for saddle point systems.
The classical multigrid algorithms are often referred to as geometric multigrid methods since they are based on a hierarchy of grids and their operators depend on the geometry of these grids. Algebraic multigrid (AMG) methods were developed by Brandt, McCormick, and Ruge [18] to overcome limitations due to geometric properties of the mesh. In geometric multigrid, the grid hierarchy is known and appropriate relaxation methods have to be defined to achieve multigrid optimality. Conversely, the goal of classical AMG, as stated in [52], is to maintain simple relaxation methods and find a suitable coarse-grid correction. The AMG method is a powerful extension that is based on a black-box idea, i.e. it solely depends on matrix coefficients.
When considering multigrid algorithms, there is an enormous degree of freedom in choosing the algorithmic components. The question is how to choose individual multigrid components for concrete situations. Local Fourier analysis (LFA) is considered as the main analysis tool to establish quantitative convergence estimates and to optimize geometric multigrid components such as smoothers or intergrid transfer operators. The idea was introduced by Brandt [15] and later extended and refined in [16]. In the AMG setting, the evaluation of different multigrid components cannot be established by LFA and is more complicated as we describe and explain in Section 5.
Our research primarily focuses on the development of multigrid smoothers for

two-dimensional systems of PDEs, in both a geometric and algebraic multigrid context. This thesis is structured as follows: in Chapter 2 we introduce the general definition and classification of partial differential equations. Moreover, we study different discretization methods, particularly the finite difference method and the finite element method. Subsequently, an introduction into multigrid methods is given in Chapter 3. After an overview of basic iterative methods, we give a detailed description of geometric multigrid methods. We continue with basics about LFA, an algebraic multigrid introduction and some parallelization aspects.
The main part of this thesis proceeds in Chapter 4 with the treatment of the Stokes equations in the context of geometric multigrid methods. We employ LFA to help us analyze and construct better algorithms for the solution of the Stokes equations with multigrid methods. We develop and compare different smoothers by using LFA. The well-known Vanka smoother [73] already serves as a good smoother in geometric multigrid methods for the Stokes equations. However, due to its overlapping sets, it is computationally expensive and not easy to parallelize. To this end, we introduce a non-overlapping smoother in Chapter 4. We analyze this non-overlapping smoother in comparison with well-known smoothers for the Stokes equations. While doing this, we investigate the parallelization capacity of all these smoothers in comparison. Mismatching numerical results show limitations of LFA which lead us to new algorithmic developments based on the non-overlapping smoother.
Geometric multigrid methods serve as efficient solvers for saddle point systems. However, the geometric multigrid methods are limited due to geometric properties of the underlying grids. To this end, this chapter focuses on algebraic multigrid methods to overcome this limitation and to address general matrix equations. Algebraic multigrid methods are flexible methods since they are based on general matrix equations. We primarily focus on extending the applicability of algebraic multigrid to a broader class of problems, for example, the second-order definite Maxwell's equations (5.1). The difficulty of these equations is characterized by a large near null-space where a pointwise smoother leads to a non-optimal method. Although traditional AMG methods are based on simple pointwise smoothers, non-pointwise smoothers are needed to overcome this problem, such as the overlapping Schwarz smoother by Arnold, Falk and Winther [5] or the distributive relaxation by Hiptmair [40]. These methods are based on geometric multigrid settings. There are also algebraic multigrid algorithms that are able to overcome the difficulty of Maxwell's equations [41, 42, 48, 62]. These methods are based on projections into auxiliary spaces or on auxiliary nodal matrices and depend on geometric information. This lack of fully algebraic algorithms motivates us to the automatic construction of algebraic multigrid smoothers and complemetary coarse-grid correction procedures. The theory developed in [30] provides guidance in constructing these algorithms.
We finalize this thesis with the conclusion in Chapter 6.

2 Partial differential equations

Differential equations have one basic characteristic in common: the equations relate functions to derivatives in such a way that the functions themselves can be determined. The functions represent physical quantities and the derivatives represent their rates of change, whereas the equations define the relationship between the two. We subdivide differential equations into ordinary differential equations (ODEs) and partial differential equations (PDEs). ODEs contain functions of one variable and the derivatives of those functions, while PDEs include partial derivatives. This chapter focuses on the definition of different types of PDEs as well as the introduction of discretization methods. We highlight important topics that will be important in the remainder of this thesis. The content of this chapter is mainly based on the textbooks [63, 72].

2.1 Definition and classification of partial differential equations

In what follows, we call a bounded and connected subset of $\mathbb{R}^d, d \in \mathbb{N}$, a *domain* and denote it by Ω and its boundary by $\partial\Omega$.

Definition 2.1. (Partial differential equation) Let $\Omega \subset \mathbb{R}^d$ and $u(\mathbf{x})$ be a k-times continuously differentiable function on Ω. A partial differential equation (PDE) for the function $u(\mathbf{x})$ that depends on $\mathbf{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_d) \in \Omega$ is an equation of the form:

$$F(\mathbf{x}, u(\mathbf{x}), \frac{\partial u(\mathbf{x})}{\partial \mathrm{x}_1}, \ldots, \frac{\partial u(\mathbf{x})}{\partial \mathrm{x}_d}, \frac{\partial^2 u(\mathbf{x})}{\partial \mathrm{x}_1 \partial \mathrm{x}_1}, \ldots, \frac{\partial^2 u(\mathbf{x})}{\partial \mathrm{x}_1 \partial \mathrm{x}_d}, \ldots) = 0,$$

where F depends on $\mathbf{x}$, the value of $u(\mathbf{x})$ and the partial derivatives of $u(\mathbf{x})$ at $\mathbf{x}$.

For the treatment of partial differential equations it is useful to introduce a classification scheme which identifies classes of equations with common characteristics. In this section, we provide the basic classifications of PDEs. The type of an equation determines the *order*, the *homogeneity* and the *linearity*. The *order* of an equation refers to the highest order derivative included in the equation.

Additionally, PDEs can be distinguished by their *homogeneity*. Some equations are labeled as *non-homogeneous*. They include terms that do not depend on the unknown function u. Moreover, four types of *linearity* can be identified. If F depends only linearly on u and all partial derivatives, the PDE is called *linear*, i.e. coefficient functions depend only on variables $\mathbf{x}$, not on u or derivatives of u. *Semilinear* PDEs are ones in which the coefficient functions of the partial derivative of highest-order depend only on $\mathbf{x}$, not on u or derivatives of u. A PDE is called *quasilinear* if the coefficient functions of the partial derivatives of highest degree depend only on $\mathbf{x}$, u or lower-order derivatives of u. Otherwise, the equation is called a *non-linear* PDE. We direct our attention to linear PDEs. More specifically, we concentrate on second-order PDEs, which can be classified according to the following definition:

Definition 2.2. (Classification of linear PDEs of second order) A linear PDE of second order has the form

$$\mathcal{A}u(\mathbf{x}) := \sum_{i,j=1}^{d} a_{i,j}(\mathbf{x}) \frac{\partial^2}{\partial \mathrm{x}_i \partial \mathrm{x}_j} u(\mathbf{x}) + \sum_{j=1}^{d} b_j(\mathbf{x}) \frac{\partial}{\partial \mathrm{x}_j} u(\mathbf{x}) + c(\mathbf{x}) u(\mathbf{x}) = f(\mathbf{x}).$$

Depending on the eigenvalues of the coefficient matrix $A = (a_{i,j})_{i,j=1}^{d}$, these PDEs are called:

- *elliptic* - all eigenvalues of A have the same sign,
- *parabolic* - all eigenvalues of A, except for one vanishing eigenvalue, have the same sign,
- *hyperbolic* - all eigenvalues of A have the same sign, except for one eigenvalue that has the opposite sign.

A common and simple example for an elliptic second-order linear PDE is the Poisson equation (2.1). In order to facilitate readability, we simplify our representation such that u and f denote the appropriate functions that depend on $\mathbf{x}$.

Definition 2.3. (Poisson equation) An elliptic linear PDE of second order of the following form,

$$-\sum_{i=1}^{d} \frac{\partial^2}{\partial \mathrm{x}_i^2} u(\mathbf{x}) = f(\mathbf{x}) \quad \Leftrightarrow \quad -\Delta u = f, \tag{2.1}$$

is called Poisson equation.

The Poisson equation is a special case of the elliptic diffusion problem (2.2) where $a : \mathbb{R}^d \to \mathbb{R}$ is chosen such that $a(\mathbf{x}) = 1$. Different choices of the function $a(\mathbf{x})$ lead to diverse properties of the equation and thereby require various solution

strategies. Some of the considerations in Chapter 5 are based on the diffusion problem and its difficulties. However, for the sake of simplicity, we use the Poisson equation as a model problem to introduce the basic principles.

Definition 2.4. (Elliptic diffusion equation) An elliptic linear PDE of second order of the following form,

$$-\nabla(a(\mathbf{x})\nabla u) = f, \tag{2.2}$$

is called elliptic diffusion equation.

2.1.1 Systems of PDEs

While the last section focuses on PDEs with only one variable function u, so-called *scalar PDEs*, we now turn our attention to *systems of PDEs*. Here, we search for more than one unknown function, i.e. we have a vector of functions $\mathbf{z} = (z_1, \dots, z_q), q \in \mathbb{N}$. The target of this thesis lies in the solution of systems of q partial differential equations involving q unknowns $z_l, l = 1, \dots, q$.

Definition 2.5. (System of partial differential equation) Let $\Omega \subset \mathbb{R}^d$ and let $z_l, l = 1, \dots, q$ be k times continuously differentiable functions on Ω. A system of q partial differential equations or system of PDEs for the functions $z_l(\mathbf{x})$ that depends on $\mathbf{x} = (\mathrm{x}_1, \dots, \mathrm{x}_d) \in \Omega$ is a system of equations of the form:

$$F_{l,m}(\mathbf{x}, z_l(\mathbf{x}), \frac{\partial z_l(\mathbf{x})}{\partial \mathrm{x}_1}, \dots, \frac{\partial z_l(\mathbf{x})}{\partial \mathrm{x}_d}, \frac{\partial z_l(\mathbf{x})}{\partial \mathrm{x}_1 \partial x_1}, \dots, \frac{\partial^2 z_l(\mathbf{x})}{\partial \mathrm{x}_1 \partial \mathrm{x}_d}, \dots) = 0,$$
$$m = 1, \dots, q,$$

where $F_{l,m}$ depends on $\mathbf{x}$, the value of $z_l(\mathbf{x})$ and the partial derivatives of $z_l(\mathbf{x})$ at $\mathbf{x}$.

The classifications introduced for scalar PDEs can be generalized to systems of PDEs. The term *order* for systems is used with two different meanings. The most common definition refers to the equations of which the system is composed, i.e. the order of the system is defined by the equation with highest order. We use this definition throughout this thesis. However, it is also possible to assign an order to the system as a whole. *Homogeneity* and *linearity* can be identified analogously to the definition for scalar PDEs. We focus on second-order linear systems of PDEs.

Definition 2.6. (Second-order linear systems of PDEs) Consider a linear system of q partial differential equations of second order involving q unknowns z_l,

$l = 1, \dots, q$,

$$\boldsymbol{\mathcal{A}}\mathbf{z}(\mathbf{x}) = \begin{pmatrix} \mathcal{A}_{1,1} & \dots & \mathcal{A}_{1,q} \\ \vdots & \ddots & \vdots \\ \mathcal{A}_{q,1} & \dots & \mathcal{A}_{q,q} \end{pmatrix} \begin{pmatrix} z_1(\mathbf{x}) \\ \vdots \\ z_q(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \mathrm{f}_1(\mathbf{x}) \\ \vdots \\ \mathrm{f}_q(\mathbf{x}) \end{pmatrix},$$

with

$$\mathcal{A}_{l,m} z_m(\mathbf{x}) := \sum_{i,j=1}^{d} (a_{l,m})_{i,j}(\mathbf{x}) \frac{\partial^2}{\partial \mathrm{x}_i \partial \mathrm{x}_j} z_m(\mathbf{x}) + \sum_{j=1}^{d} (b_{l,m})_j(\mathbf{x}) \frac{\partial}{\partial \mathrm{x}_j} z_m(\mathbf{x}) + c(\mathbf{x}) z_m(\mathbf{x}).$$

Definition 2.7. The symbol of the expression $\boldsymbol{\mathcal{A}}\mathbf{z}(\mathbf{x})$ as given in Definition 2.6 is

$$\boldsymbol{\mathcal{A}}^{Symb}(\boldsymbol{\xi}) := \begin{pmatrix} \mathcal{A}_{1,1}^{Symb} & \dots & \mathcal{A}_{1,q}^{Symb} \\ \vdots & \ddots & \vdots \\ \mathcal{A}_{q,1}^{Symb} & \dots & \mathcal{A}_{q,q}^{Symb} \end{pmatrix}$$

with

$$\mathcal{A}_{l,m}^{Symb}(\boldsymbol{\xi}) := \sum_{i,j=1}^{d} (a_{l,m})_{i,j}(\mathbf{x}) (\imath\boldsymbol{\xi})^{(i,j)} + \sum_{j=1}^{d} (b_{l,m})_j(\mathbf{x}) (\imath\boldsymbol{\xi})^{(j)} + c(\mathbf{x}),$$

where $(\imath\boldsymbol{\xi})^{(i,j)} := \imath\xi_i \cdot \imath\xi_j$ and $(\imath\boldsymbol{\xi})^{(j)} := \imath\xi_j$.
The principal part of the symbol is

$$\boldsymbol{\mathcal{A}}^{p}(\boldsymbol{\xi}) := \begin{pmatrix} \mathcal{A}_{1,1}^{p} & \dots & \mathcal{A}_{1,q}^{p} \\ \vdots & \ddots & \vdots \\ \mathcal{A}_{q,1}^{p} & \dots & \mathcal{A}_{q,q}^{p} \end{pmatrix}$$

with

$$\mathcal{A}_{l,m}^{p}(\boldsymbol{\xi}) := \sum_{i,j=1}^{d} (a_{l,m})_{i,j}(\mathbf{x}) (\imath\boldsymbol{\xi})^{(i,j)},$$

where $(\imath\boldsymbol{\xi})^{(i,j)} := \imath\xi_i \cdot \imath\xi_j$.

In general, we need to be careful about defining the "principal part" of a system. The naive approach of simply taking the "terms of highest order" may lead to difficulties for some systems. This can be resolved by assigning weights to the equations and unknowns of the operator $\boldsymbol{\mathcal{A}}\mathbf{z}(\mathbf{x})$. For the sake of simplicity, we do not discuss this in detail and refer to [63].

Definition 2.8. (Classification of second-order linear systems of PDEs) Let $\mathcal{A}_{l,m}z_m(\mathbf{x})$ and $\boldsymbol{\mathcal{A}}^p(\boldsymbol{\xi})$ be defined as given in Definition 2.6 and Definition 2.7. Then, systems of PDEs can be defined as *elliptic* by

$$\det \boldsymbol{\mathcal{A}}^p(\boldsymbol{\xi}) \neq 0, \quad \forall \xi \neq 0.$$

It is very rare that a real life phenomenon can be modeled by a single partial differential equation. Usually it takes a system of coupled partial differential equations to yield a complete model. Therefore, we usually have multiple equations and unknowns as formulated in Definition 2.5 and Definition 2.6.
In what follows, we list all systems of PDEs that are part of this thesis. However, we are not diving into the characteristics here, but save the presentation of details for later. One of the problems we deal with are the two-dimensional Stokes equations (2.3). For the purpose of indicating notations that are commonly used, we characterize the three unknown functions $\mathbf{z} = (z_1, z_2, z_3)$ of the two-dimensional Stokes equations by $\mathbf{u} := (z_1, z_2)^T$ and $p := z_3$. For two-dimensional Maxwell's equations (2.4), we have the unknown functions $\mathbf{z} = (z_1, z_2)$.

Definition 2.9. (The two-dimensional Stokes equations)

$$\begin{aligned} -\boldsymbol{\Delta}\mathbf{u} + \nabla p &= \mathbf{f}, \quad \text{in } \Omega, \\ \nabla \mathbf{u} &= 0, \quad \text{in } \Omega, \end{aligned} \tag{2.3}$$

where $\boldsymbol{\Delta} := \begin{pmatrix} \Delta & \\ & \Delta \end{pmatrix}$. The Stokes problem can be classified as a second-order linear elliptic system of PDEs. Here, we have a system with $q = 3$ equations and unknowns.

Definition 2.10. (The second-order definite Maxwell's equations)
The two-dimensional second-order curl-curl problem commonly referred to as Maxwell's equations is given by

$$\nabla \times \nabla \times \mathbf{z} + \beta \mathbf{z} = \mathbf{f}, \quad \text{in } \Omega, \tag{2.4}$$

where $\nabla\times$ defines the curl operator, details can be found in Chapter 5. This system consists of $q = 2$ equations and unknowns.

While a partial differential equation by itself usually has multiple solutions, uniqueness is often a desirable property for a solution of a problem. Imposing initial conditions, i.e. given values at a particular point, or boundary conditions, i.e. given values on the domain's boundary or parts of the boundary of the domain, lead to initial value problems or boundary value problems respectively.

2.1.2 Boundary conditions

Among various different boundary conditions, the following three common types on a domain $\Omega \subset \mathbb{R}^n$ will be studied.

- **Dirichlet boundary conditions**
  If the solution z_1 is given specific values at the boundaries, say

  $$z_1(\mathbf{x}) = \mathrm{d}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega,$$

  where $\mathrm{d} : \partial\Omega \to \mathbb{R}$ is a known function, we have a Dirichlet-type boundary condition.

- **Neumann boundary conditions**
  If the derivative rather than the function itself is specified, we have a Neumann-type boundary condition. Generally, Neumann conditions for a unknown function z_1 can be written in the following form:

  $$\frac{\partial z_1}{\partial \mathbf{n}}(\mathbf{x}) = \mathrm{d}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega,$$

  where $\mathrm{d} : \partial\Omega \to \mathbb{R}$ is a known function and $\partial z_1/\partial \mathbf{n}$ is the partial derivative of z_1 with respect to the normal $\mathbf{n}$ of the boundary $\partial\Omega$.

- **Periodic boundary conditions**
  Some domains do not require boundary conditions, for example if a PDE is defined on a torus. Here, the boundary conditions are termed as periodic. An examplary definition of periodic boundary conditions in one dimension on the domain $\Omega = [0, 1]$ for the unknown function z_1 can be given by:

  $$\begin{aligned} z_1(0) &= z_1(1), \\ \frac{\partial z_1(0)}{\partial \mathrm{x}_1} &= \frac{\partial z_1(1)}{\partial \mathrm{x}_1}, \quad \mathrm{x}_1 \in \Omega. \end{aligned}$$

To be formally correct, we have to prove the existence and uniqueness of the solution of a PDE. To this end, we need a few prerequisites from functional analysis. Important considerations are given in [12]. For example, the Lax-Milgram theorem provides support to tackle elliptic PDEs. Especially for systems of equations such as the Stokes equations and Maxwell's equations the proof is not straightforward. The existence of a solution for non-linear equations is even more challenging. We do not go into detail, refer to [21] and assume the existence of a unique solution of the PDEs that we consider in this thesis.

2.2 Discretization methods

An important step before numerically solving a PDE is to find a discrete algebraic replacement of the continuous problem. This is based on the discretization of the domain Ω into a discrete set of points. This set of points is denoted by Ω_h and it is called a grid or a mesh. We denominate the mesh size or grid width by h. Discretization methods convert a PDE into a system of equations, which can then be solved by linear algebra techniques. There are several discretization methods in use, such as the finite volume (FV), finite element (FE), and finite difference (FD) methods, here we focus on FD and FE discretization. Each method has its specific approach to discretization. To maintain simplicity, we introduce the methods based on scalar PDEs which can easily be generalized to systems of PDEs.

2.2.1 Finite difference method

The approximation of the finite difference method replaces the derivatives in the differential operator equation with differential quotients. For the sake of simplicity, we consider the one-dimensional case first. Here, we have one unknown function u and one equation with right-hand side f. A finite difference method is related to the definition of the derivative of a function u at a point $x \in \mathbb{R}$,

$$u'(x) := \lim_{h \to 0} \frac{u(x+h) - u(x)}{h}.$$

The definition indicates that in order to get good approximations, h must be sufficiently small (h approaches zero, without vanishing). It remains to specify the term "good approximation". Actually, the approximation is good when the error caused by replacing the derivative by the differential quotient approaches zero when h approaches zero. It is possible to quantify this discretization error using Taylor expansion,

$$u(x+h) = u(x) + u'(x)h + \mathcal{O}(h^2),$$

where the $\mathcal{O}$-notation denotes the difference between the Taylor polynomial of degree one and the original function, which indicates that the error of the approximation is proportional to h^2. We deduce an approximation for the first derivative of the function u,

$$u'(x) = \frac{u(x+h) - u(x)}{h} + \mathcal{O}(h). \tag{2.5}$$

The error created by replacing the derivative $u'(x)$ by the differential quotient is of order h. Approximation (2.5) is known as the *forward difference* approximation

of u'. In order to improve the accuracy, we define higher-order approximations of the derivative by taking the points $x - h$ and $x + h$ into account, e.g.

$$u(x+h) \approx u(x) + u'(x)h + \frac{u''(x)}{2}h^2 + \frac{u'''(x)}{6}h^3,$$

$$u(x-h) \approx u(x) - u'(x)h + \frac{u''(x)}{2}h^2 - \frac{u'''(x)}{6}h^3.$$

By simple calculation we obtain the *central difference* second-order approximation of the first and the second derivative, respectively

$$u'(x) \approx \frac{u(x+h) - u(x-h)}{2h}, \tag{2.6}$$

$$u''(x) \approx \frac{u(x+h) + u(x-h) - 2u(x)}{h^2}. \tag{2.7}$$

Discretizing the one-dimensional Poisson equation (2.1) with Dirichlet boundary conditions on the domain $\Omega = [0, 1]$ with $n+1$ equidistant grid points leads to the linear system (2.8). Here, we consider the discretization on grid points such that $i = 0, \ldots, n$ indicates the index of the grid point, $x_i = ih$ describes the location on the grid, $h := 1/n$, $u_i := u(ih)$ and $f_i := f(ih)$.

$$\begin{aligned} &-\frac{1}{h^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) = f_i, \qquad \text{for } i = 1, \ldots, n-1, \\ &u_0 = u(0) = d(0), \\ &u_n = u(1) = d(1). \end{aligned} \tag{2.8}$$

Stencil notation makes use of the locality, i.e. computation is based on values of neighboring arguments to the gridpoint ih. Using stencil notation, we are able to represent operators in a compact way:

$$\begin{aligned} &\frac{1}{h^2}\begin{bmatrix} -1 & 2 & -1 \end{bmatrix} u_i = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} f_i \quad \text{for } i = 1, \ldots, n-1, \\ &\text{with } \begin{bmatrix} \mathrm{s}_{-1} & \mathrm{s}_0 & \mathrm{s}_1 \end{bmatrix} w_i = \sum_{\kappa=-1}^{1} \mathrm{s}_\kappa w_{i+\kappa} \text{ for } i = 1, \ldots, n-1. \end{aligned} \tag{2.9}$$

Since this thesis focuses on two-dimensional problems, we consider the two-dimensional finite difference method in what follows. A second-order approximation of the second derivative in two-dimensions is given by the combination of (2.7) in both directions x_1 and x_2,

$$\begin{aligned} u''(\mathrm{x}_1, \mathrm{x}_2) \approx{}& \frac{u(\mathrm{x}_1 + h, \mathrm{x}_2) + u(\mathrm{x}_1 - h, \mathrm{x}_2)}{h^2} \\ &+ \frac{u(\mathrm{x}_1, \mathrm{x}_2 + h) + u(\mathrm{x}_1, \mathrm{x}_2 - h) - 4u(\mathrm{x}_1, \mathrm{x}_2)}{h^2}. \end{aligned}$$

Representing the operator within this equation in stencil notation leads to:

$$\frac{1}{h^2}\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}. \tag{2.10}$$

Having these representations at hand, we are able to define all components belonging to the two-dimensional Stokes equations discretized by the finite difference method. It is important to note that the Stokes system consists of three different unknowns and equations. Therefore, we establish the discretization based on the different components. More details are presented in Chapter 4. The different unknowns are denoted by $u_{i,j} = u(ih, jh)$, $v_{i,j} = v(ih, jh)$ and $p_{i,j} = p(ih, jh)$. The central finite difference approximation to the first partial derivative of p in the x_1-direction $\partial p/\partial x_1$ is labeled as $B_1 p$. The Laplace operator, Δ, applied to u is represented by $A_1 u$. In particular, the operators appear as

$$\begin{aligned} B_1 p &= \frac{p_{i+1,j} - p_{i-1,j}}{2h}, \\ A_1 u &= \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2}. \end{aligned}$$

The discrete representation of the remaining components can be obtained in a similar way. This results in the following finite difference representation of the Stokes equations in two-dimensions:

$$\begin{pmatrix} -A_1 & & B_1 \\ & -A_1 & B_2 \\ -B_1^T & -B_2^T & \end{pmatrix} \begin{pmatrix} u \\ v \\ p \end{pmatrix} = \begin{pmatrix} \mathrm{f}_1 \\ \mathrm{f}_2 \\ 0 \end{pmatrix}. \tag{2.11}$$

2.2.2 Finite element method

In this section we introduce the finite element (FE) method, an approach to approximate the unknown exact solution u of a PDE based on basis functions and the so-called weak form of a PDE. The basic idea is to subdivide the domain Ω into smaller parts and locally approximate the PDE by simple equations. This is followed by a recombination into a global system of equations which can then be solved by a numerical method. A detailed description can be found below. The basic references for this discretization method are the books of Dietrich Braess [12] and Peter Monk [55]. The method can be structured into six different steps:

1. Establish the strong formulation.
2. Obtain the weak formulation.
3. Split the domain.

4. Choose approximations for the unknown functions u.

5. Choose weight functions.

6. Solve the system.

The details, advantages and connections between each step are described and explained below. For the sake of simplicity, we introduce the method based on model problem (2.1), the Poisson equation ($-\Delta u = -\nabla^2 u = f$) in one-dimension, which can be easily adapted for higher dimensions and systems of PDEs. This general form of a PDE is called the strong formulation (step 1). Equivalently, one can formulate the so-called *variational formulation* or *weak formulation* (step 2). The weak formulation means that instead of solving a differential equation of the underlying problem, an integral equation is solved. In many cases, the original formulation of the PDE is already a weak formulation. Then, we can skip the first step of the method. Let us introduce a formal definition of the weak formulation. We start by defining some standard spaces of functions:

$C^k(\Omega)$: the set of k times continuously differentiable functions on Ω,
$C_0^k(\Omega)$: the set of functions $\phi \in C^k(\Omega)$ having compact support in Ω,
$C_0^k(\Omega)'$: the dual space of $C_0^k(\Omega)$,
$L^p(\Omega), 1 \leq p < \infty$: the set of functions ϕ on Ω for which $|\phi|^p$ is Lebesgue integrable.

The fundamental Sobolev spaces are denoted $W^{s,p}(\Omega)$, where $s \in \mathbb{Z}_+, 1 \leq p < \infty$, and defined as:

$$W^{s,p}(\Omega) := \{\phi \in L^p(\Omega) \mid \partial^\alpha \phi \in L^p(\Omega) \text{ for all } |\alpha| \leq s\}.$$

We set $p = 2$ in what follows. An alternative definition of Sobolev spaces for $p = 2$ is to define the Hilbert spaces $H^s(\Omega), s \in \mathbb{Z}_+$, with

$$H^s(\Omega) := \left\{ u \in C_0^\infty(\Omega)' \middle| u = U|_\Omega \text{ for some } U \in W^{s,2}(\mathbb{R}^n) \right\}.$$

We stick to the L^2-inner product $\langle u, v \rangle := \int_\Omega uv$. Combining these informations leads us to the variational formulation for the Poisson equation by assuming that the solution u is in $H^1(\Omega)$. We multiply the partial differential equation with an arbitrary *test function* or *weight-function* v and compute an integral over the whole domain Ω,

$$-\int_\Omega v\nabla^2 u = \int_\Omega fv.$$

Applying Green's identity $\int_\Omega v\nabla w = -\int_\Omega \nabla v w + \int_{\partial\Omega} vwv_i$ with $w = \nabla u$ and assuming $v = 0$ on $\partial\Omega$ results in the weak formulation of the Poisson equation,

$$\int_\Omega \nabla v \nabla u = \int_\Omega fv \quad \text{for all } v \in H^1(\Omega). \tag{2.12}$$

The finite element discretization requires the reformulation into the weak form (2.12), since a weak form lowers the continuity requirements on the approximation functions in contrast to the strong formulation. Note that the strong form contains two separate partial derivatives of u. Therefore, it requires u to be continuously differentiable until at least the second partial derivative. The new formulation has lowered this requirement to only first partial derivatives by incorporating one of the partial derivatives into the weight-function v. Thereby, the weak form allows the use of easy-to-construct and implement polynomials, which are used to construct the approximation functions u as shown in step four. In addition, the power of the weak formulation results from the fact that singular solutions - solutions of the differential equation that cannot be obtained from the general solutions - are allowed. This is due to the lower continuity requirement. Assuming that the solution u of the Poisson problem is in the Hilbert space $H^1(\Omega)$ leads us to the properties of the weak form, namely, continuously differentiable until first partial derivative only.
Next, in step three, we split the whole domain into smaller parts so-called elements. A finite element can be formally described as given in the next definition.

Definition 2.11. (**Finite element**) A finite element is given by three components (T, P_T, Σ_T), where

- the element $T \in \mathcal{T}_h$ is a discrete set while $\mathcal{T}_h$ is a geometric domain,
- the space of functions P_T (usually polynomials) is defined over the set T, and
- Σ_T is a set of linear functionals defined over the space P_T. These linear functionals are called the *degrees of freedom* (dofs) of the finite element.

Even though there are many possibilities, the particular choice of the three components of this general formulation has to fulfill some conditions. The element T has to be chosen such that it is non-degenerate; P_T has to be a finite-dimensional vector space of functions that are convenient to implement, e.g. polynomials; and the dofs have to be chosen so that they can uniquely determine a function in P_T. For the one-dimensional example (2.1), we make the simple choice of a discretization into equidistant points $x_i = ih$, with $i = 1, \ldots, n$, being the so-called nodes, c.f. Figure 2.1. In addition, we define the linear finite elements, s.t.

- $T = [x_i, x_{i+1}], i \in \mathbb{N}$,

- P_T = polynomials $\mathrm{p}(x)$ of degree at most one, i.e. $\mathrm{p}(x) = a + bx$, with $a, b \in \mathbb{R}$,
- $\Sigma_T = \{\alpha_k, 1 \leq k \leq 2\}$, the set of degrees of freedom for the element T.

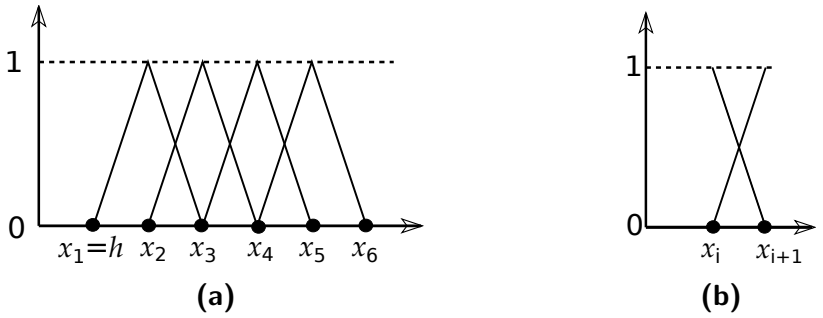


Figure 2.1: Basis functions for the six-node discretization (a) and two basis functions on one exemplary element (b).

We define the so-called *basis functions* or *shape functions*, $\phi_i(x) \in P_T, x \in \Omega$, consistent with the set of dofs α_m, where

$$\alpha_m(\phi_i) := \delta_{mi} = \begin{cases} 1, & \text{if } m = i, \\ 0, & \text{otherwise.} \end{cases} \tag{2.13}$$

We can define the dofs α_m, such that

$$\alpha_1(u) = u(x_i), \quad \alpha_2(u) = u(x_{i+1}), \tag{2.14}$$

holds for each element. Instead of using the combination of (2.13) and (2.14), we equivalently define the basis functions $\phi_i(x)$ globally such that

$$\phi_i(x_j) := \delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{2.15}$$

The choice of basis functions $\phi_i(x)$ as in (2.15) means that they are equal to one only at node x_i and zero at all the other nodes. It follows that every $\phi_i(x)$ is only non-zero at elements that share node x_j, as can be seen in Figure 2.1a. Figure 2.1b demonstrates that two of these basis functions can approximate an arbitrary linear polynomial on one element with an appropriate choice of weights. Given these facts about shape functions, we can write an approximation function $\tilde{u}$ of u as

$$\tilde{u}(x) = \sum_{i=1}^{n} \alpha_i \phi_i(x),$$

i.e., as a linear combination of the piecewise continuous linear polynomials $\phi_i(x)$ so that the dofs α_i serve as coefficients. Furthermore, we express the weight function v (step 5) as

$$v(x) = \sum_{j=1}^{n} v_j \phi_j(x).$$

Now let us rewrite the weak form (2.12) by using the approximation function $\tilde{u}(x) \in H^1(\Omega)$ and the weight function $v(x) \in H^1(\Omega)$. First, we define the bilinear and linear forms a and l

$$a(u,v) := \int_\Omega \nabla v \nabla u \quad \text{and} \quad l(v) := \int_\Omega fv.$$

Then, we can rewrite the weak form (2.12) as

$$a(u,v) = l(v) \quad u, v \in H^1(\Omega).$$

We insert the two approximations in our weak form and obtain

$$a\left(\sum_{j=1}^{N} v_j \phi_j(x), \sum_{i=1}^{N} \alpha_i \phi_i(x)\right) = l\left(\sum_{j=1}^{N} v_j \phi_j(x)\right).$$

Using the properties of a bilinear form and the underlying function space, after some transformation steps we eventually obtain the following equation:

$$\sum_{j=1}^{N} v_j \sum_{i=1}^{n} a\left(\phi_j(x), \phi_i(x)\right) \alpha_i = \sum_{j=1}^{N} v_j l(\phi_j(x)).$$

This can be written in the compact form,

$$\mathbf{v}^T A \mathbf{u} = \mathbf{v}^T \mathbf{f} \quad \Rightarrow \quad A\mathbf{u} = \mathbf{f}, \tag{2.16}$$

where

$$\mathbf{v}^T = (v_1, \ldots, v_n), \quad \mathbf{u}^T = (\alpha_1, \ldots, \alpha_n), \quad \mathbf{f}^T = (l(\phi_1(x)), \ldots, l(\phi_n(x))),$$
$$A = \begin{pmatrix} a\left(\phi_1(x), \phi_1(x)\right) & \ldots & a\left(\phi_1(x), \phi_n(x)\right) \\ \vdots & & \vdots \\ a\left(\phi_n(x), \phi_1(x)\right) & \ldots & a\left(\phi_n(x), \phi_n(x)\right) \end{pmatrix}.$$

Step six consists of solving the resulting system (2.16), whereby this thesis focuses on multigrid methods as the solution method.

Finite element method for Maxwell's equations

In Chapter 5 we focus on Maxwell's equations, therefore we briefly describe the finite element discretization of these equations into the so-called Nédélec elements here. Consequently, we obtain elements that are suitable for the discretization of Maxwell's equations. More details can be found in [55].
We have seen a simple form of FEM for the Poisson equation where it is appropriate to define simple basis functions in the space $H^1(\Omega)$. However, for Maxwell's equations we need to derive edge based finite elements (curl conforming elements) which are based on the function space

$$H(\mathrm{curl};\Omega) := \left\{ u \in \left(L^2(\Omega)\right)^3 | \nabla \times u \in \left(L^2(\Omega)\right)^3 \right\}.$$

The strong formulation of Maxwell's equations, as introduced in (2.4), can be equivalently formulated by the following weak form,

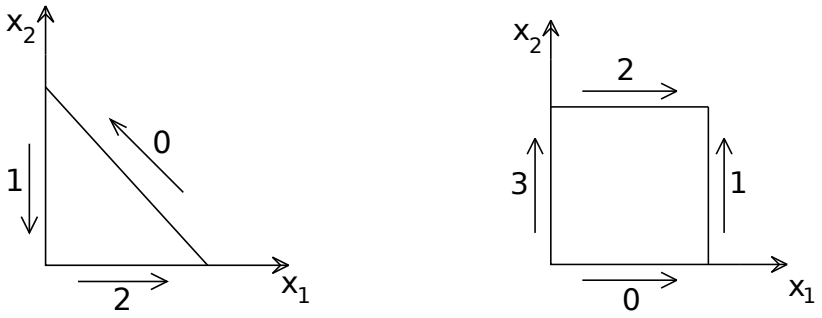
$$(\nabla \times u, \nabla \times v) + (\beta u, v) = (f, v) \quad \text{for all } v \in H(\mathrm{curl};\Omega). \tag{2.17}$$

Assuming that Ω is discretized by a quadrilateral or triangular mesh, $\mathcal{T}$, in two dimensions, Nédélec elements represent basis functions in $H(\mathrm{curl}, \mathcal{T})$ spaces. The three components (T, P_T, Σ_T) that we use in this thesis are defined as given in Definition 2.11. For Nédélec elements we specify T by the splitting of $\mathcal{T}$ into squares or triangles s.t. one element, T, is a square or triangle. Moreover, the degrees of freedom given in Σ_T are associated with the edges e_i of T. Figure 2.2 illustrates a reference element for the choice of discretization into squares or triangles. One dof for u relates to each edge. Therefore, there are $\mathrm{c} = 4$ dofs related to each element on a square and $\mathrm{c} = 3$ dofs related to each element on a triangle, s.t.

$$\alpha_i(u) = \int_{\mathrm{e}_i} t_i \cdot u, \quad i \in \{1, \ldots, \mathrm{c}\},$$

for every edge, e_i, in the reference element. The vector t_i is the tangential unit vector of the edge e_i. Therefore, the degrees of freedom α_i for the edge elements are the average value of tangential components of the vector field on each edge. One exemplary choice of the direction for the unit tangential vectors is illustrated in Figure 2.2.
Let ϕ denote the global edge basis function and let $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2)^T \in \Omega$ be a point. The reference basis functions of the Nédélec elements are given by the requirement $\alpha_i(\phi_j) = \delta_{ij}$ [4, 56, 66], s.t.

$$\phi_0(\mathbf{x}) = \begin{pmatrix} -\mathrm{x}_2 \\ \mathrm{x}_1 \end{pmatrix}, \quad \phi_1(\mathbf{x}) = \begin{pmatrix} -\mathrm{x}_2 \\ \mathrm{x}_1 - 1 \end{pmatrix}, \quad \phi_2(\mathbf{x}) = \begin{pmatrix} 1 - \mathrm{x}_2 \\ \mathrm{x}_1 \end{pmatrix},$$



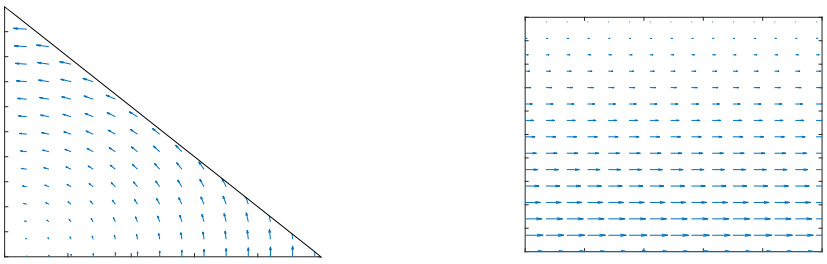
(a) triangular element **(b)** quadrilateral element

Figure 2.2: The degrees of freedom of two-dimensional Nédélec elements in a reference configuration on a triangular grid (a) and a quadrilateral grid (b).

for triangular elements, and

$$\phi_0(\mathbf{x}) = \begin{pmatrix} 1 - \mathrm{x}_2 \\ 0 \end{pmatrix}, \quad \phi_1(\mathbf{x}) = \begin{pmatrix} 0 \\ \mathrm{x}_1 \end{pmatrix}, \quad \phi_2(\mathbf{x}) = \begin{pmatrix} \mathrm{x}_2 \\ 0 \end{pmatrix}, \quad \phi_3(\mathbf{x}) = \begin{pmatrix} 0 \\ 1 - \mathrm{x}_1 \end{pmatrix},$$

for quadrilateral elements. A visualization of one exemplary basis function ϕ_j for triangular and quadrilateral elements is given in Figure 2.3.



(a) triangular element **(b)** quadrilateral element

Figure 2.3: Reference basis functions of the Nédélec discretization on a reference element.

In order to preserve tangential continuity in the global setting, we need to use the so-called *Piola mappings* to obtain the global basis functions [4].

Definition 2.12. (**Affine element transformation**) Let $\hat{K}$ be the reference element and $\hat{x}$ be a variable on the reference element. An affine element K (triangle or quadrilateral) is obtained by the affine element transformation

$$K \ni x = F_K(\hat{x}) = B_K \hat{x} + b_K,$$

from the reference element $\hat{K}$ to an element K in the mesh. Here, B_K denotes the Jacobian (a matrix of all first-order partial derivatives of the vector-valued function) of the transformation $F_K(\hat{x})$.

Definition 2.13. (**Piola mapping**) Let $\Omega_0 \in \mathbb{R}^n$, let F be a nondegenerate mapping from Ω_0 onto $F(\Omega_0) = \Omega$ and let $\phi \in L^2(\Omega_0, \mathbb{R}^n)$. The covariant Piola mapping $\mathcal{F}^{\text{curl}}$ is defined by

$$\mathcal{F}^{\text{curl}}(\phi) = B^{-T}\phi \circ F^{-1}.$$

The shape function $\phi_j(\mathbf{x})$ on element $K = F_K(\hat{K})$ are transformed by

$$\phi_j(\mathbf{x}) = \left(B^{-T}\hat{\phi}\right) \circ F_K^{-1}(\mathbf{x}).$$

The curl operator in two-dimensions is transformed by

$$\operatorname{curl}\phi = \frac{1}{\det B_K} \operatorname{curl}\hat{\phi}\left(F_K^{-1}(x)\right).$$

More details can be found in [22, 24]. To guarantee global continuity with Piola-mapped elements, special care has to be taken with regard to the orientation of geometric entities. In particular the interplay between local and global orientation is significant. It is common to direct edges in a fashion that gives a consistent orientation of the boundary of each element. However, this may result in two adjacent elements with different direction of their common edge. In this setting, two adjacent elements would naturally disagree on the direction of their tangential on a common edge. To ensure global continuity, it is necessary to introduce appropriate sign changes for the mapped basis functions. That means for two corresponding basis functions: we change the sign of one function such that both basis functions that correspond to the same global degree of freedom have the same orientation. Thus, the basis functions can be obtained by first mapping the nodal basis functions from the reference element and then correcting those basis functions with a change of sign. The global Nédélec basis functions are non-zero only in the two elements who share the edge that is related to the basis function. As already described in the context of finite element methods for the Poisson equation, we can write an approximation function $\tilde{u}$ of u as linear combination of the vector polynomials $\phi_j(x)$. By using a weight function, we rewrite the weak form (2.17) in the same manner as described in the last subsection. A more detailed treatment can be found in [56, 57]. In this way, we are able to construct a compact form of Maxwell's equations based on Nédélec elements, such that we obtain a linear system of the form

$$A\mathbf{u} = \mathbf{f},$$

with $A = N + Z$, where N is the discrete approximation of the weak form of the curl-curl term in (2.4), and Z is the discrete approximation to the weak form of the β term in (2.4).

2.3 Classification of the linear systems arising from systems of PDEs

As we have already seen, after the discretization of a linear PDE of second order by finite difference or finite elements, we obtain linear systems of equations. The matrices of these systems and the systems themselves can be classified with regard to several aspects. Since the subject of this thesis is the solution of so-called *saddle point systems*, we define such systems in the next definition.

Definition 2.14. (**Saddle point systems**) We consider a block linear system of the form

$$\begin{pmatrix} A & B_1^T \\ B_2 & -C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{2.18}$$

where $A \in \mathbb{R}^{t\times t}$, $B_1, B_2 \in \mathbb{R}^{s\times t}$ and $C \in \mathbb{R}^{s\times s}$ with $s \leq t$. It is obvious that, under suitable partitioning, any linear system can be transferred in the form (2.18). We exclude the case where A, B_1 or B_2 are zero. A saddle point system, is a block system of form (2.18) that satisfies one or more of the following conditions.

1. A is symmetric, i.e. $A = A^T$
2. the symmetric part of A, namely $H = \frac{1}{2}(A + A^T)$, is positive semidefinite
3. $B_1 = B_2 = B$
4. C is symmetric ($C = C^T$) and positive semidefinite
5. $C = \Theta$, where Θ is the zero matrix

Note that condition 5 implies condition 4. The most basic case is obtained when all the above conditions are satisfied. In this case A is symmetric positive semidefinite and we have a symmetric linear system of the form

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \tag{2.19}$$

This system arises as the first-order optimality conditions for the following equality-constrained quadratic programming problem

$$\begin{aligned} &\min\ J(x) = x^T Ax - f^T x \\ &\text{subjected to } Bx = g. \end{aligned}$$

In this case the variable y represents the vector of Lagrange multipliers. Any solution (x^*, y^*) of (2.19) is a saddle point for the Lagrangian

$$L(x, y) = x^T Ax - f^T x + (Bx - g)^T y,$$

hence the name "saddle point system" given to Definition 2.14.

3 Multigrid methods

This chapter focuses on solving linear systems of equations resulting from the discretization of PDEs. Although, the core of this thesis lies in the solution of systems of PDEs, due to simplicity, we introduce the basic principles of multigrid methods with regard to scalar PDEs. In addition, we restrict ourselves to the standard model problem, the Poisson problem (2.1) in the course of Chapter 3. Subsequently, we deal with the difficulties of linear systems that arise from systems of PDEs and adapt multigrid components appropriately. We stress that the term "multigrid" does not refer to a single algorithm, but rather categorizes a whole class of methods.
We motivate the fundamental components of the geometric and algebraic multigrid methods, establish analysis techniques for these methods and reflect upon parallelization aspects of multigrid methods. In the remainder of this thesis we refer to geometric multigrid methods as *multigrid methods* and use the term *algebraic multigrid methods* wherever those occur. The content of this chapter is mainly based on the textbooks [23, 71].

3.1 Basic iterative methods

Consider solving the linear system

$$A\mathbf{u} = \mathbf{f}, \tag{3.1}$$

where $A \in \mathbb{R}^{n\times n}$ is a real positive definite matrix and $\mathbf{u}, \mathbf{f} \in \mathbb{R}^n$ are n-dimensional vectors. Solving such linear systems has been a topic of research for many years. When solving partial differential equations in scientific or engineering applications large sparse matrices often appear. A sparse matrix is defined by a matrix of which most of the elements are zero. We distinguish between direct and iterative solution approaches for solving such systems. The drawback of direct methods like Gaussian elimination lies in the computational cost and the amount of storage required [70]. For this reason, in this work we solve sparse linear systems by iterative algorithms.
The idea of an iterative method is to start with an initial guess $\mathbf{u}_0$ and improve this guess through a sequence of updating steps. Given that the sparse linear system

of equations (3.1) has a unique solution $\mathbf{u} = A^{-1}\mathbf{f}$ and $\tilde{\mathbf{u}}$ is an approximation to the solution, we can define the error, $\mathbf{e}$, by

$$\mathbf{e} = \mathbf{u} - \tilde{\mathbf{u}}. \tag{3.2}$$

Unfortunately, we cannot compute the error since we do not know the solution $\mathbf{u}$. Therefore, we introduce a computable measure that indicates how well the current solution fulfills the system of equations, the *residual* $\mathbf{r}$. It is defined by

$$\mathbf{r} = \mathbf{f} - A\tilde{\mathbf{u}}.$$

Using the definition of $\mathbf{r}$ and $\mathbf{e}$ and that $A\mathbf{u} = \mathbf{f}$, we see the following relationship between the error and the residual,

$$A\mathbf{e} = \mathbf{r}. \tag{3.3}$$

To improve the approximation, $\tilde{\mathbf{u}}$, we solve (3.3) for $\mathbf{e}$ and then compute a new approximation using the definition of the error,

$$\tilde{\mathbf{u}} \leftarrow \tilde{\mathbf{u}} + \mathbf{e}.$$

This idea of residual correction is the basis of almost all iterative methods to solve systems of linear equations. In what follows, we use the notation $\mathbf{u}^{(k)}$ to denote the current approximation, while the new, updated approximation is denoted $\mathbf{u}^{(k+1)}$. In practice, $\mathbf{u}^{(k+1)}$ will be computed, and afterwards $\mathbf{u}^{(k+1)}$ plays the role of $\mathbf{u}^{(k)}$. This procedure is continued until convergence to the solution is obtained. We will specify the convergence of a method in Theorem 3.1. Next, we consider the splitting,

$$A = D - L - U,$$

where D is the diagonal of A, and $-L$ and $-U$ are the strictly lower and upper triangular parts of A, respectively. One classical stationary linear iterative method that can be introduced using this splitting is the so-called *Jacobi method*. The term *stationary linear* refers to the fact that the update rule is linear in the unknown u and does not change from one iteration to the next. First, we rewrite (3.1) in the following way,

$$A\mathbf{u} = \mathbf{f} \quad \Leftrightarrow \quad D\mathbf{u} = (L + U)\mathbf{u} + \mathbf{f}.$$

Thus, a new approximation is computed by the iteration

$$\mathbf{u}^{(k+1)} = D^{-1}(L + U)\mathbf{u}^{(k)} + D^{-1}\mathbf{f} = (I - D^{-1}A)\mathbf{u}^{(k)} + D^{-1}\mathbf{f} = \mathbf{u}^{(k)} + D^{-1}\mathbf{r}^{(k)}.$$

By assuming this new iteration $\mathbf{u}^{(k+1)}$ as intermediate solution and then improving through a weighted average, we get the ω-Jacobi method,

$$\mathbf{u}^{(k+1)} = (1 - \omega)\mathbf{u}^{(k)} + \omega(\mathbf{u}^{(k)} + D^{-1}\mathbf{r}^{(k)}) = \mathbf{u}^{(k)} + \omega D^{-1}\mathbf{r}^{(k)}, \tag{3.4}$$

where $\omega \in \mathbb{R}$, $\omega > 0$. In contrast to the Jacobi method, the Gauss-Seidel method uses components of the new approximation as soon as they are computed, which changes the iteration to

$$\mathbf{u}^{(k+1)} = (I - (D-L)^{-1}A)\mathbf{u}^{(k)} + (D-L)^{-1}\mathbf{f} = \mathbf{u}^{(k)} + (D-L)^{-1}\mathbf{r}^{(k)}.$$

A weighted version of the Gauss-Seidel method is called successive over-relaxation (SOR) and has the form,

$$\mathbf{u}^{(k+1)} = (I - (\frac{1}{\omega}D - L)^{-1}A)\mathbf{u}^{(k)} + (\frac{1}{\omega}D - L)^{-1}\mathbf{f} = \mathbf{u}^{(k)} + (\frac{1}{\omega}D - L)^{-1}\mathbf{r}^{(k)}. \tag{3.5}$$

The order in which the components of $\mathbf{u}^{(k)}$ are updated is significant for the performance of Gauss-Seidel and SOR methods, while the performance of the Jacobi method is independent of the order. Changing the order results in variations of the GS method. Instead of sweeping through the components in ascending order, so-called lexicographic Gauss-Seidel (GS-LEX), we might sweep through the components in descending order, alternate between ascending and descending orders or update all the even components first and then update all the odd components. The latter procedure is called the *red-black Gauss-Seidel method* (GS-RB).
The general form of stationary iterative methods such as Jacobi, ω-Jacobi, Gauss-Seidel and SOR can be written as

$$\mathbf{u}^{(k+1)} = (I - M^{-1}A)\mathbf{u}^{(k)} + M^{-1}\mathbf{f} = \mathbf{u}^{(k)} + M^{-1}\mathbf{r}^{(k)}, \tag{3.6}$$

where M^{-1} can be seen as an approximation to A^{-1}, with $M = D, M = \frac{1}{\omega}D, M = (D-L)$ and $M = (\frac{1}{\omega}D - L)$ for Jacobi, ω-Jacobi, Gauss-Seidel and SOR, respectively. The convergence of such a method is established by the spectral radius in the following way.

Theorem 3.1. *Let $\mathcal{S} = I - M^{-1}A$ be the so-called iteration matrix. A linear iterative method is convergent (i.e. $\lim_{m\to\infty}(\mathcal{S})^m = 0$) iff the spectral radius of the iteration matrix $\mathcal{S}$ is strictly less than one, i.e.*

$$\rho(\mathcal{S}) < 1.$$

$\rho(\mathcal{S})$ is called the convergence factor.

3.2 Geometric multigrid methods

Multigrid (MG) methods have been proven useful in a variety of applications. The benefit results from the fast h-independent convergence in combination with

the computational work of order $\mathcal{O}(n)$; meaning that the convergence factor of the method is small and bounded, independent of the number of grid points, n, and that the number of arithmetic operations is proportional to the number of grid points. These facts imply what is often called the *optimality of multigrid methods*. In the 1960s, Fedorenko [33] and Bakhvalov [7] were the first to show the optimality of multigrid methods. Fedorenko proved convergence of multigrid methods for the Poisson equation in the unit square discretized by the finite difference method, while Bakhvalov extended his investigations to general elliptic boundary value problems with variable coefficients. Since the 1980s multigrid methods became a fastly growing field of research, due to their efficiency in a wide range of applications. For example, systems of PDEs can usually be treated by multigrid with efficiency similar to that of scalar equations. Nevertheless, for more challenging equations like saddle point systems, the solution process is more complicated. A crucial point is the selection of optimal components and parameters. We start with the basic principles of multigrid methods, which is followed by some adjustments with respect to more challenging systems of equations.

3.2.1 Stationary iterative methods

Next, we analyze the behavior of stationary iterative methods to motivate the use of these methods in multigrid. We gain valuable insight by applying stationary iterations to the discretized version of the two-dimensional Poisson problem (2.1). A rectangular domain is discretized with $(n-1)^2$ interior grid points. For each gridpoint (ih, jh) the discretized Poisson equation appears as

$$\frac{1}{h^2}\left(-u_{i,j-1} - u_{i-1,j} + 4u_{i,j} - u_{i,j+1} - u_{i+1,j}\right) = f_{i,j}, \quad 1 \leq i,j \leq n-1. \tag{3.7}$$

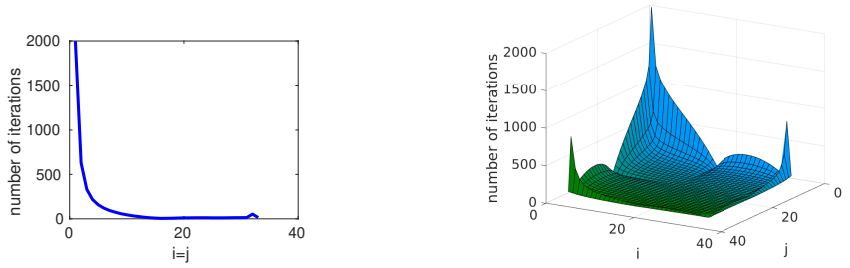
This leads to the linear system

$$A_h \mathbf{u}_h = \mathbf{f}_h,$$

with $A_h \in \mathbb{R}^{(n-1)^2 \times (n-1)^2}$, $h = 1/n$ and $\mathbf{u}_h, \mathbf{f}_h \in \mathbb{R}^{(n-1)^2}$. If the right-hand side $f_{i,j}$ of (3.7) is set to zero it is called *homogeneous Poisson equation* or *Laplace equation*. For the purpose of motivating the use of iterative methods, it is sufficient to work with the homogeneous system, since the exact solution is known ($\mathbf{u} = \mathbf{0}$) and the error of the approximation $\tilde{\mathbf{u}}$ is simply $-\tilde{\mathbf{u}}$. The Poisson problem serves as model problem, however the following observations can be made for a large class of problems.
We apply stationary iterations to the homogeneous system of equations with an initial guess consisting of the vectors (or Fourier modes)

$$\tilde{\mathbf{u}} = \sin\left(k\pi ih\right)\sin\left(l\pi jh\right), \quad i,j = 1,\ldots,n-1, \quad k,l = 1,\ldots,n-1. \tag{3.8}$$

The integers k, l are called frequencies. Notice that small values of k, l correspond to long, smooth waves, which are called low frequencies; while large values correspond to highly oscillatory waves, which are called high frequencies. We now observe how different modes behave under iteration. We call the following procedure *Fourier mode study*. It is based on the introduction given in [23]. We first apply the weighted Jacobi iteration with $\omega = 4/5$ to problem (3.7).



(a) **(b)**

Figure 3.1: Number of iterations needed to reduce the error of $v_{i,j}$ as defined in (3.9) by a factor of 10^3 on a grid of size 33×33 using ω-Jacobi with $\omega = 4/5$.

In Figure 3.1 the number of iterations needed to reduce the error by a factor of 10^3 is given. We observe that the stationary iterations are able to reduce high frequencies much better than low frequencies. This reduction of oscillatory error components is called the "smoothing property"; for this reason, this sort of an iterative method is called a *smoother* or *relaxation method*. Figure 3.1 shows two possibilities to visualize the number of iterations. Figure 3.1b illustrates a complete representation including all combinations of indices i and j. A two-dimensional view is given in Figure 3.1a such that the number of iterations corresponding to indices $i = j$ is illustrated. Here, both figures serve as good representations of the number of iterations. This might not always be the case. The two-dimensional view could neglect combinations of indices i and j that possess a different behavior. Whereas the three-dimensional representation may be less descriptive and clear. By way of illustration, we use the two-dimensional representation within the course of this thesis.
We now turn to a more analytical approach. Again, we consider the weighted Jacobi iteration applied to the model problem (3.7) and we recall that $\mathcal{S} = I - M^{-1}A$ is the iteration matrix of the Jacobi method. It follows that the eigenvalues of $\mathcal{S}$ and A are related by

$$\lambda_{\mathcal{S}}^{(k,l)} = 1 - \frac{\omega}{4}\lambda_A^{(k,l)}, \quad k, l = 1, \ldots, n-1.$$

The eigenvalues of A are given by

$$\lambda_A^{(k,l)} = 4 - 2\left(\cos\left(k\pi h\right) + \cos\left(l\pi h\right)\right), \quad k, l = 1, \ldots, n-1,$$

with corresponding eigenvectors $\mathbf{v}^{(k,l)}$ given by

$$v_{i,j}^{(k,l)} = \sin(k\pi ih)\sin(l\pi jh), \quad i,j,k,l = 1,\dots,n-1. \tag{3.9}$$

We see that the eigenvectors of A are the Fourier modes in (3.8). These results imply that the eigenvalues of $\mathcal{S}$ are

$$\lambda_{\mathcal{S}}^{(k,l)} = 1 - \frac{\omega}{2}\Big(2 - \cos(k\pi h) - \cos(l\pi h)\Big), \quad k,l = 1,\dots,n-1,$$

while the eigenvectors of $\mathcal{S}$ are the same as the eigenvectors of A. Note that, if $0 < \omega \leq 1$, then $|\lambda_{\mathcal{S}}^{(k,l)}| < 1$ and the weighted Jacobi iteration converges. We discuss these convergence properties in more detail in Section 3.3. Let $\mathbf{e}^{(0)}$ be the error of an initial guess used in the weighted Jacobi method. Since it is possible to expand arbitrary vectors in terms of a set of eigenvectors, it is also possible to represent $\mathbf{e}^{(0)}$ using the eigenvectors of A in the form

$$\mathbf{e}^{(0)} = \sum_{k,l=1}^{n-1} c_{k,l}\mathbf{v}^{(k,l)},$$

where the coefficients $c_{k,l} \in \mathbb{R}$ give the "amount" of each mode in the error. After m sweeps of the iteration, the error is given by

$$\mathbf{e}^{(m)} = (I - M^{-1}A)^m\mathbf{e}^{(0)}.$$

Then, we have

$$\mathbf{e}^{(m)} = (I - M^{-1}A)^m\mathbf{e}^{(0)} = \sum_{k,l=1}^{n-1} c_{k,l}(I - M^{-1}A)^m\mathbf{v}^{(k,l)} = \sum_{k,l=1}^{n-1} c_{k,l}\lambda_{\mathcal{S}}^{(k,l)}\mathbf{v}^{(k,l)}.$$

The last equality follows because the eigenvectors of A and $(I - M^{-1}A)$ are the same and it holds that $(I - M^{-1}A)\mathbf{v}^{(k,l)} = \lambda_{\mathcal{S}}^{(k,l)}\mathbf{v}^{(k,l)}$. This expansion shows that after m iterations, the kth and lth mode of the initial error has been reduced by a factor of $\lambda_{\mathcal{S}}^{(k,l)}$. We will see that this property is not shared by all stationary iterations.

The complement to the smoother is the *coarse-grid correction*, which constitute the process of eliminating the low frequency components. The idea is as follows: assume that a relaxation method has been applied until only smooth error components remain, as seen in Figure 3.2. This smooth error looks more oscillatory on a coarser grid. Therefore, it is possible to smooth again on the coarser grid and thereby eliminate more error frequencies. This procedure may be repeated multiple times using grids that become more and more coarse. Incorporating transfer operators and a coarse-grid solver leads to the approximate solution of the system. These low and high frequency eliminations are used in multigrid methods. For the sake of simplicity, we focus on the description of a two-grid method which can be easily expanded by recursion to a multigrid method.

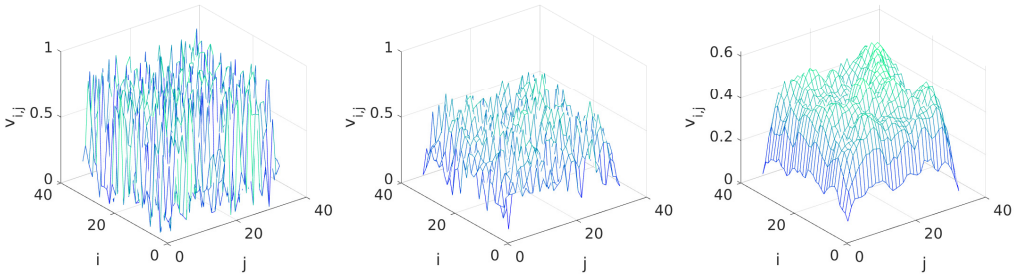


Figure 3.2: Error of an arbitrarily chosen initial guess and right-hand side zero on a grid of size 33×33. Before (left), after application of one (center) and three iteration steps using GS-LEX.

3.2.2 Two-grid methods

The combination of a relaxation method and a coarse-grid correction (which consists of restriction, a direct solver and prolongation) results in a two-grid method. We define the method in a formal way, starting with a detailed view on the different components, followed by a description of the entire two-grid method.

Smoother

In the previous section, we have observed the error smoothing effect of iterative methods such as the Jacobi method. This is due to the elimination of high frequency error components. The following outlines how the smoothing factor can be introduced after the formal definition of low and high modes. First, we introduce the so-called *error propagation operator* $\mathcal{S}_h$ resulting of a reformulation of the linear iterative method (3.6). In order to address the influence of the operators due to the mesh size h, we add the subscript notation. In addition, we specify $\mathcal{S}_h(\omega)$ to distinguish between weighted iterative methods with different weighting factors ω. Using Equations (3.3) and (3.2) it follows that

$$\mathbf{e}_h^{(k+1)} = \left(I - M_h^{-1} A_h\right) \mathbf{e}_h^{(k)} = \mathcal{S}_h(\omega)\mathbf{e}_h^{(k)} = \mathcal{S}_h^k(\omega)\mathbf{e}_h^{(0)}. \tag{3.10}$$

From this equation it becomes clear that the iteration converges if and only if the spectral radius $\rho(\mathcal{S}_h(\omega)) = \rho(I - M_h^{-1}A_h)$ (also called the convergence factor of $\mathcal{S}_h(\omega)$) is strictly smaller than one. Now, we describe how the operator $\mathcal{S}_h(\omega)$ influences the error, starting with the formal definition of low and high frequencies.

Definition 3.2. Let $\mathbf{v}_h^{(k,l)}$ be an eigenvector of A_h as given in (3.9). Then

$$\begin{aligned} \mathbf{v}_h^{(k,l)} \text{ is a low frequency component} &\quad :\Longleftrightarrow \quad 1 \leq \max(k,l) < \frac{n}{2}, \\ \mathbf{v}_h^{(k,l)} \text{ is a high frequency component} &\quad :\Longleftrightarrow \quad \frac{n}{2} \leq \max(k,l) < n. \end{aligned}$$

An iterative method is called a smoother if it eliminates the high frequency components of the error. For this reason, the definition of the smoothing factor rests on high frequencies only.

Definition 3.3. Let $\lambda_h^{(k,l)}$ be an eigenvalue of $\mathcal{S}_h(\omega)$. The *smoothing factor*

$$\mu(\mathcal{S}_h(\omega)) := \max\left\{\lambda_h^{(k,l)} : \frac{n}{2} \leq \max(k,l) \leq n\right\}$$

of $\mathcal{S}_h(\omega)$ represents the worst factor by which high-frequency error components are reduced per relaxation step.

The smoothing factor indicates the smoothing properties of the weighted Jacobi method as introduced in Section 3.2.1

$$\begin{aligned}\mu_h(\mathcal{S}_h(\omega)) &= \max\left\{\left|1 - \frac{\omega}{2}\left(2 - \cos(k\pi h) + \cos(l\pi h)\right)\right| : \frac{n}{2} \leq \max(k,l) \leq n\right\} \\ &= \max\left\{\left|1 - \frac{\omega}{2}\right|, |1 - 2\omega|\right\}.\end{aligned}$$

This shows that the method has a good smoothing effect on the error for the Poisson problem with the optimal choice of $\omega = 4/5$. Since the eigenvectors of the iteration matrix of the Gauss-Seidel and SOR methods are not the same as the eigenvectors of A_h, the analysis of these smoothers is not straightforward. A detailed analysis can be found in Section 3.3, where we introduce an analysis tool called *local Fourier analysis*. In that section, we also compare different orderings of the grid points. It turns out that the smoothing properties depend on the right choice of relaxation parameters and, in the case of the Gauss-Seidel iteration, also on the ordering of grid points. In the following paragraph we assume that an appropriate relaxation method was chosen and we denote the error propagation operator S_h as the *relaxation* or *smoothing operator*.

Coarse-grid correction

After eliminating the high frequency error components through a relaxation method, the coarse-grid correction is applied consisting of these five consecutive steps:

1. Compute the residual: $\mathbf{r}_h = \mathbf{f}_h - A_h\mathbf{u}_h$.
2. Restrict the residual to the coarse grid: $\mathbf{r}_H = R_h^H\mathbf{r}_h$.
3. Compute the error on the coarse grid by solving the system: $A_H\mathbf{e}_H = \mathbf{r}_H$.
4. Prolongate the error to the fine grid: $\mathbf{e}_h = P_H^h\mathbf{e}_H$.
5. Update the approximation: $\mathbf{u}_h = \mathbf{u}_h + \mathbf{e}_h$.

This results in the coarse-grid correction operator B_h^H, which influences an error as follows:

$$\mathbf{e}_h^{(k+1)} = B_h^H \mathbf{e}_h^{(k)} := (I - P_H^h A_H^{-1} R_h^H A_h)\mathbf{e}_h^{(k)}. \tag{3.11}$$

Next, we specify and list some standard examples of the multigrid components: restriction, prolongation, choices of coarse grids as well as coarse-grid operators.

Choices of coarse grids and coarse-grid operators

In this paragraph, we present some common choices for the coarse grid. The simplest choice is *standard coarsening*, where the coarse grid has twice the grid spacing of the next finest grid. If the mesh size h is doubled in one direction only, it is denoted by *semicoarsening*, i.e., $H = (2h, h)$ (x-semicoarsening) or $H = (h, 2h)$ (y-semicoarsening). If the coarse-grid points are distributed in the fine grid in a checkerboard manner, it is denoted by *red-black coarsening*. The corresponding grids are based on the fact that the coarse-grid points are a subset of the fine-grid points. In Chapter 4, we consider coarsening hierarchies where this is not the case.
We define two different coarse-grid operators A_H. One choice is to use the direct analog of A_h on the coarse grid. For our model problem, this means

$$A_H = \frac{1}{H^2} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}_H .$$

Another choice is the so-called *Galerkin coarse-grid operator* which is defined by

$$A_H = R_h^H A_h P_H^h, \tag{3.12}$$

where R_h^H and P_H^h are appropriate transfer operators.

Restriction and prolongation operators

The idea of multigrid methods only works if we find sufficient restriction operators R_h^H and prolongation operators P_H^h to transfer the error to the coarse grid and fine grid, respectively. In what follows, we introduce some standard operators for the Poisson problem. In Chapters 4 and 5 we introduce some additional operators that are useful for problems like the Stokes equations and Maxwell's equations. The choice of intergrid transfer operators R_h^H and P_H^h is closely related to the choice of the coarse grid. Here, we restrict ourselves to standard coarsening.
A restriction operator R_h^H maps h-grid functions to H-grid functions. For the sake of simplicity, we represent the restriction operator with the stencil notation introduced in Equation (2.9). Here, the point at the center denotes the fine-grid

point that corresponds to the current coarse-grid point. The entry at the center, as well as the surrounding entries, demonstrate the impact of the fine-grid error values on the coarse-grid error values. We use h on the bottom right and H on the top right of the stencil to demonstrate that the operator maps the error from the fine grid, with grid width h, to the coarse grid, with width H. We define two well-known restriction operators.

Definition 3.4. The *injection operator*

$$\begin{bmatrix} & & \\ & 1 & \\ & & \end{bmatrix}_h^H,$$

is the simplest example for a restriction operator since it identifies grid functions at coarse-grid points with corresponding grid functions at fine-grid points.

Improving the representation of the error on the coarse grid while generating a computationally more expensive restriction operator leads to the full weighting (FW) operator.

Definition 3.5. The *full weighting (FW) restriction operator* is given by the stencil

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^H.$$

A prolongation operator P_H^h maps H-grid functions to h-grid functions. Again, we represent the prolongation with a stencil notation, where the point at the center denotes the current fine-grid point. The labels h and H are reversed and we use open brackets to demonstrate that the operator works in the opposite direction than the restriction operator. We define the bilinear interpolation operator, which is the adjoint of the full weighting operator up to a constant factor.

Definition 3.6. The *bilinear interpolation operator* is given by the stencil

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_H^h.$$

Two-grid cycle

Having defined all components, we are able to set up a two-grid method. The full two-grid cycle consists of a certain number of pre-smoothing steps ν_1, followed by

coarse-grid correction, and terminates with additional post-smoothing steps ν_2. This leads to the two-grid operator E_{TG},

$$E_{TG} := \mathcal{S}_h^{\nu_2} B_h^H \mathcal{S}_h^{\nu_1} = \mathcal{S}_h^{\nu_2} (I - P_H^h A_H^{-1} R_h^H A_h) \mathcal{S}_h^{\nu_1}.$$

The two-grid cycle in algorithmic form is presented in Algorithm 3.1.

Algorithm 3.1: Two-grid cycle

1 **Presmoother**
  for $j = 1 : \nu_1$ **do**
    $\mathbf{u}_h \leftarrow \mathbf{u}_h + M_h^{-1}\mathbf{r}_h$
  end for
2 **Coarse-grid correction**
  $\mathbf{r}_h \leftarrow \mathbf{f}_h - A_h\mathbf{u}_h$
  $\mathbf{r}_H \leftarrow R_h^H\mathbf{r}_h$
  $\mathbf{e}_H \leftarrow A_H^{-1}\mathbf{r}_H$
  $\mathbf{e}_h \leftarrow P_H^h\mathbf{e}_H$
  $\mathbf{u}_h \leftarrow \mathbf{u}_h + \mathbf{e}_h$
3 **Postsmoother**
  for $j = 1 : \nu_2$ **do**
    $\mathbf{u}_h \leftarrow \mathbf{u}_h + M_h^{-1}\mathbf{r}_h$
  end for

So far, we have not discussed the influence of choosing different components. This will be shown by experiments and an analysis in the latter part of this thesis. However, there are no simple rules of how to choose sufficient components, especially for more complicated problems. An additional question we have to answer is: how to solve the coarse-grid equation (step three in the coarse-grid correction process, well-known as the *residual equation*) efficiently. The answer is an iterative method, namely the two-grid method. This leads to a recursive procedure, the so-called *multigrid method*.

3.2.3 Multigrid methods

We can easily expand two-grid methods by recursion to multigrid methods to solve the coarse-grid equation efficiently. Once we transfer the error to the coarser grid, the situation is basically the same as before, since the coarse-grid problem is not much different from the original problem. Therefore, we can apply the two-grid method to the coarse-grid problem, which does not differ from applying the two-grid method to the original problem. We just introduce more grids and recursively apply this idea until a direct solution of the residual equation is possible. This leads to a multigrid method. Depending on the number of recursive two-grid steps on each grid level, we distinguish different types of multigrid methods, the

V-cycle and the *W-cycle*. The names are motivated by the resulting hierarchy structure of the grids, as seen in Figure 3.3. The V-cycle repeats each recursive step on each grid once, while for the W-cycle the number of consecutive steps is set to two.

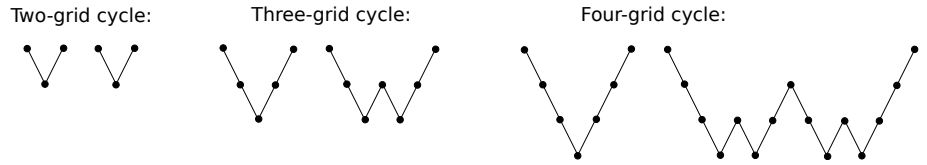


Figure 3.3: Structure of one multigrid V-cycle (left) and W-cycle (right) for different numbers of grid levels.

Having defined two-grid methods and multigrid methods, we are interested in the efficiency of these procedures. We want to know why these methods are considered efficient and how they perform compared to other iterative methods. Therefore, we prove convergence results, demonstrate computational work and explain the efficiency of two-grid methods and multigrid methods.

Multigrid efficiency

Two facts, h-independent fast convergence and a computational cost of $\mathcal{O}(n)$, constitute that multigrid methods are considered optimal methods. We will derive estimates for the multigrid convergence factor by proving the h-independent fast convergence of two-grid methods. In addition, a computational cost of $\mathcal{O}(n)$ per multigrid cycle is obtained. We start with two-grid convergence estimates. There are several possibilities to prove the h-independent convergence of the two-grid method; we use the smoothing and approximation properties to prove convergence. A detailed explanation of this theoretical approach is given in [35].

Definition 3.7. Smoothing property
The error propagation operator S_h possesses the smoothing property, if there exists a function $\vartheta(\nu)$, that is independent of h, such that for sufficiently large $\nu \in \mathbb{Z}^+$

$$\|A_h \mathcal{S}_h^\nu\|_2 \leq \vartheta(\nu) h^{-\gamma},$$

for a number $\gamma > 0$ and $\vartheta(\nu) \to 0$ as $\nu \to \infty$.

The smoothing property states that the smoother reduces the high frequency components of the error. One can measure the smoothness of an error $\mathbf{e}_h$ by a norm involving differences of the value of this error on different grid points. We choose the A_h^2-norm and recall the relation $\mathbf{e}_h^\nu = \mathcal{S}_h^\nu \mathbf{e}_h^0$ which results in $\|\mathcal{S}_h^\nu\|_{A_h^2} = \|A_h \mathcal{S}_h^\nu\|_2$ as a measure for the efficiency after ν smoothing steps.

Definition 3.8. Approximation property
The approximation property holds if there exists a constant C_1, independent of h, such that

$$\|B_h^H A_h^{-1}\|_2 = \|(A_h^{-1} - P_H^h A_H^{-1} R_h^H)\|_2 \leq C_1 h^\gamma,$$

with the same γ as in the smoothing property, Def. 3.7.

Notice that this estimate measures how well the coarse-grid solution approximates the fine-grid solution. This means that we measure the accuracy between $\mathbf{u}_h = A_h^{-1}\mathbf{f}_h$ and $P_H^h \mathbf{u}_H$, where $\mathbf{u}_H = A_H^{-1} R_h^H \mathbf{f}_h$. The combination of these two properties results in the h-independent convergence of two-grid methods with $\nu_1 = \nu, \nu_2 = 0$.

Theorem 3.9. ***Convergence of the two-grid method***
Suppose the smoothing property and the approximation property hold and let $\rho \in (0,1)$ be a fixed number. Then there exists a number of smoothing steps ν such that

$$\|E_{TG}\|_2 = \|\mathcal{S}_h^0 B_h^H \mathcal{S}_h^\nu\|_2 \leq \|B_h^H A_h^{-1}\|_2 \cdot \|A_h \mathcal{S}_h^\nu\|_2 \leq C_1 \vartheta(\nu) \leq \rho < 1.$$

Proof. Choose $\tilde{\nu}$ such that $\vartheta(\nu) \leq \frac{\rho}{C_1}$ for all $\nu > \tilde{\nu}$. Then we have

$$\begin{aligned}
\|E_{TG}\|_2 &= \|\mathcal{S}_h^0 B_h^H \mathcal{S}_h^\nu\|_2 = \|B_h^H \mathcal{S}_h^\nu\|_2 = \|B_h^H A_h^{-1} A_h \mathcal{S}_h^\nu\|_2 \\
&= \|(A_h^{-1} - P_H^h A_H^{-1} R_h^H)(A_h \mathcal{S}_h^\nu)\|_2 \\
&\leq \|A_h^{-1} - P_H^h A_H^{-1} R_h^H\|_2 \|A_h \mathcal{S}_h^\nu\|_2 \\
&\leq C_1 h^\gamma \vartheta(\nu) h^{-\gamma} = C_1 \vartheta(\nu) \leq \rho.
\end{aligned}$$

□

The convergence theorem indicates that the two-level method converges at a rate that is independent of h if sufficiently many smoothing steps are applied. If a given two-grid method converges sufficiently well, i.e., with a small convergence factor and independent of h, then the corresponding multigrid method will have similar convergence properties due to the recursive definition. It is sufficient to analyze pre-smoothing for a two-grid method ([67], L.4.4). We know that the positions of the presmoothing and postsmoothing can be interchanged or cumulated at the beginning or at the end of the whole two-grid method without changing the asymptotical rate of convergence.
In the next paragraph, we demonstrate the computational cost of multigrid methods. A reasonable measure for the computational cost is the number of arithmetic operations. We define the grid width h_k and the grid Ω_{h_k} on level k, whereby Ω_{h_0} denotes the coarsest grid. Due to the recursive definition of the multigrid

method, the computational work W_l per multigrid cycle on Ω_l can recursively be given by

$$W_1 := W_1^0 + W_0, \quad W_{k+1} := W_{k+1}^k + \gamma^k W_k \quad (k = 1, \ldots, l),$$

where W_0 denotes the computational cost of the solution on the coarsest grid Ω_{h_0} and W_{k+1}^k stands for the computational cost needed for one two-grid cycle with fine grid $\Omega_{h_{k+1}}$ and coarse grid Ω_{h_k}, excluding the cost for the solution of the residual equation on the coarser level $k+1$. This leads to the definition of computational cost with l grid levels,

$$W_l := \sum_{k=1}^{l} \gamma^{l-k} W_k^{k-1} + \gamma^{l-1} W_0, \quad \text{for } l \geq 1.$$

We focus on standard coarsening. Therefore, we have $n_k \approx 4n_{k-1}$, where n_k denotes the number of grid points on grid Ω_{h_k}. Due to boundary effects n_k is only approximately equal to $4n_{k-1}$, therefore "$\approx$" means equality up to lower order terms. We assume that multigrid components require a number of arithmetic operators per point on each grid which is bounded by a small constant C_2. This constant is independent of k, s.t.

$$W_k^{k-1} \lessapprox C_2 n_k \quad (k = 1, \ldots, l).$$

Again, " $\lessapprox$ " means equality up to lower order terms. Additionally, we assume that the coarsest grid Ω_0 is chosen such that W_0 is negligibly small, i.e., of order $\mathcal{O}(1)$. Straightforward calculation leads to the following computational cost for the V- and W-cycles.

$$\text{V-cycle } (\gamma = 1) : W_l = \sum_{k=1}^{l} W_k^{k-1} + W_0 = \frac{4}{3} C_2 n_l,$$

$$\text{W-cycle } (\gamma = 2) : W_l = \sum_{k=1}^{l} 2^{l-k} W_k^{k-1} + 2^{l-1} W_0 = 2 C_2 n_l.$$

This estimate shows that the number of arithmetic operations needed for one multigrid V- or W-cycle is proportional to the number of grid points on the finest grid for standard coarsening. Together with the existence of the small and h-independent upper bound for the convergence of the multigrid cycle, this means that multigrid methods achieve an h-independent reduction of the error in $\mathcal{O}(n)$ operations. This is why we call multigrid methods optimal methods.

3.3 Local Fourier analysis

In this section, we introduce the local Fourier analysis (LFA), which is considered the core of the theoretical understanding of multigrid methods [15]. However, we use LFA not as a theory, but as a tool to measure the quality of geometric multigrid components. It is used for the design and comparison of concrete algorithms for challenging problems. We introduce the basic ideas and formalism of LFA and show its influence on the convergence of the multigrid method.
LFA serves as a successful analysis tool since it often provides accurate performance predictions and it is applicable to a variety of problems (including systems of PDEs), while being based on reasonable assumptions. In Section 3.2, we used eigenvectors to show that stationary iterative methods, including, e.g., the Jacobi iteration, are smoothers. For the Jacobi method, the eigenvalues give us information about the smoothing property. This complete analysis of convergence properties is not possible for all kinds of problems and operators because it can be difficult to find eigenvectors and eigenvalues of the operators. This is where the key idea of LFA takes effect. We gain insight into the convergence and smoothing properties of operators by applying them to so-called Fourier modes. The following outlines how and why this simplification works. We write the operator as a constant stencil and neglect boundary conditions by analyzing on an infinite grid. Due to these assumptions, eigenfunctions of operators are given by simple exponential functions, so-called Fourier modes. We denote them by $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ and define:

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) := e^{\imath \boldsymbol{\theta} \cdot \boldsymbol{x}/\mathbf{h}}.$$

Here, $\boldsymbol{x}$ varies in the given infinite grid G_h and $\boldsymbol{\theta}$ is a vector of parameters that characterizes the frequency of the grid function (more details are given below). These modes serve as basis functions and span the whole space of bounded infinite-grid functions. Therefore, each infinite-grid function can be represented by a linear combination of Fourier modes and it is sufficient to consider the effect of an operator A_h on Fourier modes rather than on the real eigenfunctions. We are able to predict the smoothing behavior with the formal eigenvalue or Fourier symbol $\tilde{A}_h(\boldsymbol{\theta})$, i.e.,

$$A_h \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) = \tilde{A}_h(\boldsymbol{\theta}) \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}).$$

As we will see in the course of this thesis, a symbol with values smaller than one indicates that the error is damped, while values significantly smaller than one refer to a good smoothing behavior. Values that are considered to be small and therefore indicate a good smoothing behavior differ from problem to problem. LFA analyzes not only the smoothing methods but also the effect of the whole two-grid process, which increases the predictive power of the convergence behavior. In the

latter part of this section, we discuss difficulties that arise when analyzing the two-grid method as well as more sophisticated smoothers. Therefore, we introduce the invariance property to overcome these difficulties. We establish some notation first.

3.3.1 Terminology

Again, we restrict ourselves to two-dimensional problems and standard coarsening. We write $\boldsymbol{x} = (x_1, x_2)$ and assume a fixed grid width $\mathbf{h} = (h_1, h_2)$. LFA is based on the simplification of neglecting boundary conditions; therefore, we extend the problem to an infinite grid, i.e.,

$$G_h = \{\boldsymbol{x} = \boldsymbol{k}\mathbf{h} := (k_1 h_1, k_2 h_2), \mathbf{k} \in \mathbb{Z}^2\}. \tag{3.13}$$

The so-called *Fourier modes* or *Fourier functions* are the fundamental quantities of LFA. They are defined as follows.

Definition 3.10.

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}} := e^{\imath\theta_1 x_1/h_1} e^{\imath\theta_2 x_2/h_2} \quad \text{for } \boldsymbol{x} \in G_h,$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2)$ and $\imath^2 = -1$.

Since $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ is periodic in $\boldsymbol{\theta}$ with period 2π, we consider θ_i to vary continuously in an interval of length 2π. Here, we use the interval $\left[-\frac{\pi}{2}, \frac{3\pi}{2}\right)^2$. The coarse grid G_H with $\mathbf{H} = (2h_1, 2h_2)$ is defined similarly to G_h by

$$G_H = \{\boldsymbol{x} = \boldsymbol{k}\mathbf{H}, \boldsymbol{k} \in \mathbb{Z}^2\}.$$

On the grid G_h, as well as on G_H, we consider discrete operators in stencil notation, as defined in (2.9). In what follows, let A_h be a stencil operator and not a matrix anymore. For a scalar Toeplitz operator A_h, i.e., a matrix in which each descending diagonal from left to right is constant, we have

$$\begin{aligned} A_h &\hat{=} [\mathrm{s}_{\boldsymbol{\kappa}}]_h, \quad (\boldsymbol{\kappa} = (\kappa_1, \kappa_2) \in \mathbb{Z}^2), \\ A_h w_h(\boldsymbol{x}) &= \sum_{\boldsymbol{\kappa}\in\mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} w_h(\boldsymbol{x} + \boldsymbol{\kappa}\mathbf{h}), \end{aligned}$$

with constant coefficients $\mathrm{a}_{\boldsymbol{\kappa}} \in \mathbb{R}$ and $\mathbf{V}$ taken to be a finite index set. In (2.9) we recognize the index set $\mathbf{V}$ for the one-dimensional Poisson problem reaches

from -1 to 1, i.e., $\mathbf{V} = \{-1, 0, 1\}$, since the stencil consists of three values. For a two-dimensional stencil, i.e.,

$$[\mathrm{s}_{\boldsymbol{\kappa}}]_h = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & \mathrm{s}_{-1,1} & \mathrm{s}_{0,1} & \mathrm{s}_{1,1} & \cdots \\ \cdots & \mathrm{s}_{-1,0} & \mathrm{s}_{0,0} & \mathrm{s}_{1,0} & \cdots \\ \cdots & \mathrm{s}_{-1,-1} & \mathrm{s}_{0,-1} & \mathrm{s}_{1,-1} & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix},$$

the index set $\mathbf{V}$ consists of 2-tuples. The size of the stencil depends on the number and location of non-zero values in the corresponding system matrix.

Definition 3.11. We call $\tilde{A}_h(\boldsymbol{\theta}) := \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} e^{\imath \boldsymbol{\theta} \cdot \boldsymbol{\kappa}}$ the symbol of A_h.

Lemma 3.12. *For all modes $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ it holds that*

$$A_h \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) = \tilde{A}_h(\boldsymbol{\theta}) \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}).$$

Proof. By Definition 3.10 and Definition 3.11, it holds that $\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = e^{\imath \boldsymbol{\theta} \cdot \boldsymbol{x}/\mathbf{h}} := e^{\imath \theta_1 x_1/h_1} e^{\imath \theta_2 x_2/h_2}$ and $\tilde{A}_h(\boldsymbol{\theta}) := \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} e^{\imath \boldsymbol{\theta} \cdot \boldsymbol{\kappa}}$. Therefore,

$$\begin{aligned} A_h \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) &= \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} \varphi_h(\boldsymbol{\theta}, \boldsymbol{x} + \boldsymbol{\kappa}\mathbf{h}) = \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} e^{\imath \boldsymbol{\theta} \cdot \frac{\boldsymbol{x} + \boldsymbol{\kappa} \boldsymbol{h}}{\mathbf{h}}} \\ &= \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} e^{\imath \theta_1 \frac{x_1 + \kappa_1 h_1}{h_1}} e^{\imath \theta_2 \frac{x_2 + \kappa_2 h_2}{h_2}} = \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} e^{\imath \theta_1 \left(\frac{x_1}{h_1} + \kappa_1 \right)} e^{\imath \theta_2 \left(\frac{x_2}{h_2} + \kappa_2 \right)} \\ &= \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} e^{\imath \theta_1 \frac{x_1}{h_1}} e^{\imath \theta_1 \kappa_1} e^{\imath \theta_2 \frac{x_2}{h_2}} e^{\imath \theta_2 \kappa_2} = \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} e^{\imath \theta_1 \kappa_1} e^{\imath \theta_2 \kappa_2} e^{\imath \theta_1 \frac{x_1}{h_1}} e^{\imath \theta_2 \frac{x_2}{h_2}} \\ &= \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{a}_{\boldsymbol{\kappa}} e^{\imath \boldsymbol{\theta} \cdot \boldsymbol{\kappa}} e^{\imath \boldsymbol{\theta} \cdot \frac{\boldsymbol{x}}{\mathbf{h}}} \\ &= \tilde{A}_h(\boldsymbol{\theta}) \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) \end{aligned}$$

□

Example 3.13. The symbol of the two-dimensional Laplace stencil, $-\Delta_h$, defined in (2.10), is given by $\tilde{A}_h(\theta_1, \theta_2) = \frac{4 - 2\cos\theta_1 - 2\cos\theta_2}{h^2}$.

For the smoothing and two-grid analysis, we again have to distinguish high and low frequency components on G_h with respect to G_H. The definition is based on the phenomenon that only those modes $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ with $\boldsymbol{\theta} \in [-\frac{\pi}{2}, \frac{\pi}{2})^2$ are distinguishable on G_H. One recognizes that

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) \equiv \varphi(\boldsymbol{\theta}', \boldsymbol{x}) \text{ for } \boldsymbol{x} \in G_H \text{ iff } \boldsymbol{\theta} = \boldsymbol{\theta}' (\mathrm{mod} \pi). \tag{3.14}$$

This leads to the definition of high and low frequencies for standard coarsening:

Definition 3.14. Let $\varphi_h(\boldsymbol{\theta}, \cdot)$ be a Fourier mode. Then,

$$\varphi_h(\boldsymbol{\theta}, \cdot) \text{ is a low frequency mode} \quad :\Longleftrightarrow \boldsymbol{\theta} \in T^{\text{low}} := \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2,$$
$$\varphi_h(\boldsymbol{\theta}, \cdot) \text{ is a high frequency mode} \quad :\Longleftrightarrow \boldsymbol{\theta} \in T^{\text{high}} := \left[-\frac{\pi}{2}, \frac{3\pi}{2}\right)^2 \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2.$$

3.3.2 Smoothing analysis

Lemma 3.15. *Let the error-propagation symbol, $\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta})$, for a smoother $\mathcal{S}_h(\omega)$ on the infinite grid G_h satisfy*

$$\mathcal{S}_h(\omega)\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) = \tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta})\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}), \quad \boldsymbol{\theta} \in \left[-\frac{\pi}{2}, \frac{3\pi}{2}\right)^2,$$

for all $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x})$, with

$$\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta}) = I - \tilde{M}_h^{-1}(\boldsymbol{\theta})\tilde{A}_h(\boldsymbol{\theta}).$$

Then, the corresponding smoothing factor for $\mathcal{S}_h(\omega)$ is given by

$$\mu(\omega) = \mu\left(\mathcal{S}_h(\omega)\right) = \max_{\boldsymbol{\theta} \in T^{high}} \left\{ \left| \tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta}) \right| \right\}. \tag{3.15}$$

Remark 3.16. If A_h is not a scalar operator, i.e., it corresponds to the discretization of a system of PDEs, then $\left|\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta})\right|$ in (3.15) can be modified to be the absolute value of the eigenvalues of $\mathcal{S}_h(\omega, \boldsymbol{\theta})$, i.e., $|\lambda_{S_h}|$.

Example 3.17. We consider the weighted Jacobi relaxation (3.4) applied to the two-dimensional Poisson problem, i.e., $M_h := \omega D_h$, where D_h is given by

$$D_h = \frac{1}{h^2} \begin{bmatrix} & 0 & \\ 0 & 4 & 0 \\ & 0 & \end{bmatrix},$$

with symbol $\tilde{D}_h(\boldsymbol{\theta}) = \frac{4}{h^2}$. The error propagation symbol of the weighted Jacobi relaxation for the Poisson problem is

$$\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta}) = I - \omega \tilde{D}_h^{-1}(\boldsymbol{\theta})\tilde{A}_h(\boldsymbol{\theta}) = 1 - \omega \frac{4 - 2\cos\theta_1 - 2\cos\theta_2}{4}.$$

Using Definition 3.3, we have

$$\mu(\omega) = \max\left\{ |1 - 2\omega|, \left|1 - \frac{\omega}{2}\right| \right\}. \tag{3.16}$$

The choice $\omega = 0.8$ is optimal [71].

The symbol $\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta})$ indicates the influence of a relaxation method on the error. As a result, the smoothing factor $\mu(\omega)$ in (3.16) identifies the smoothing properties of the weighted Jacobi method. We reach the best possible smoothing factor with $\omega = 0.8$, namely $\mu(0.8) = 0.6$. The smoothing behavior of the weighted Jacobi method with $\omega = 0.8$ corresponding to different frequencies $\boldsymbol{\theta}$ is visualized in Figure 3.4. The LFA smoothing analysis gives a good prediction for the actual multigrid performance (if we assume that we have a coarse-grid correction operator that annihilates low-frequency error components and leaves high-frequency components unchanged). Figure 3.4 illustrates that low frequencies, i.e., $\boldsymbol{\theta} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$, are not eliminated by the relaxation method. Lemma 3.15 indicates that the smoothing factor is based on high frequencies only, i.e., $\boldsymbol{\theta} \in [-\frac{\pi}{2}, \frac{3\pi}{2})^2 \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$. In addition, the smoothing factor is based on the least damped high frequencies. Consequently, a lower value implies a better smoothing behavior, while $\mu(\omega) = 1$ indicates that the error is not damped and values larger than one predict that the relaxation method enlarges the error.

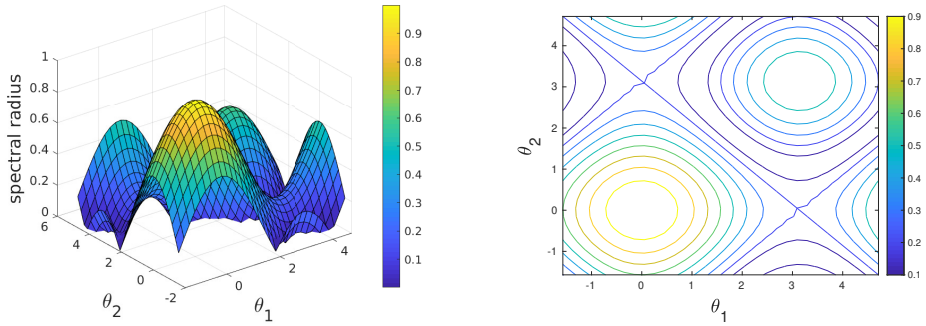


Figure 3.4: The spectral radius of the smoothing error-propagation symbol for weighted Jacobi relaxation with $\omega = 0.8$ and different frequencies $\boldsymbol{\theta}$.

The next fundamental example shows how LFA is used to analyze the smoothing behavior of GS-LEX. GS-LEX is a classical relaxation method which has been used, generalized and analyzed for a variety of PDEs [71]. GS-LEX has natural smoothing properties, but the complete analysis presented in Section 3.2 cannot be applied. LFA, however, is applicable and shows the smoothing properties of GS-LEX.

Example 3.18. We consider the weighted GS-LEX relaxation (3.5) applied to the two-dimensional Poisson problem, i.e., $M_h := (\frac{1}{\omega} D_h - L_h)$ is given by

$$M_h = \frac{1}{h^2} \begin{bmatrix} & 0 & \\ -1 & 4/\omega & 0 \\ & -1 & \end{bmatrix},$$

with symbol $\frac{1}{h^2}\left(\frac{4}{\omega} - e^{-\imath\theta_1} - e^{-\imath\theta_2}\right)$. The error propagation symbol of the weighted GS relaxation for the Poisson problem is

$$\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta}) = 1 - \frac{4 - e^{-\imath\theta_1} - e^{-\imath\theta_2} - e^{\imath\theta_1} - e^{\imath\theta_2}}{4/\omega - e^{-\imath\theta_1} - e^{-\imath\theta_2}}.$$

One can show that the optimal value of ω is not exactly but very close to one [71]. Therefore, we assume $\omega = 1$, which results in

$$\mu(\omega) = 0.5.$$

The smoothing behavior of the weighted GS method with $\omega = 1$ corresponding to different frequencies $\boldsymbol{\theta}$ is visualized in Figure 3.5.

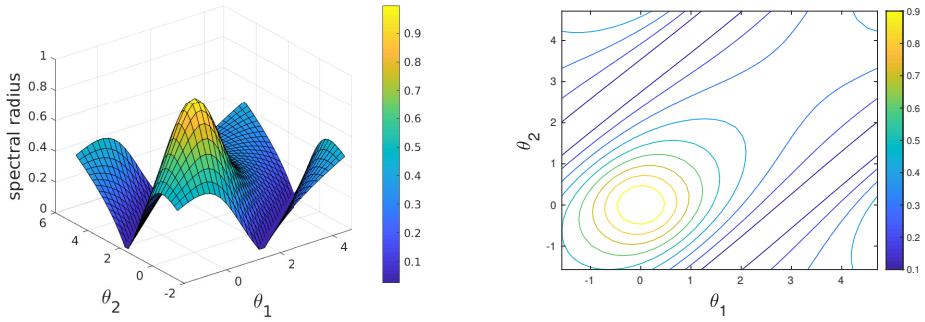


Figure 3.5: The spectral radius of the smoothing error-propagation symbol for weighted GS-LEX with $\omega = 1$ and different frequencies $\boldsymbol{\theta}$.

A natural question is: what is considered to be a good smoothing factor? A first answer is: it depends on the problem. Problems that are considered to be easier problems are expected to have smaller smoothing factors than harder problems. Indeed, there is not a definite answer, however, we compare different factors in the course of this thesis to generate an understanding of good smoothing factors, especially for the Stokes equations.

3.3.3 Two-grid analysis

The basis for the efficient performance of a multigrid method is the interplay between smoothing and coarse-grid correction. Thus, it is appropriate to perform at least a two-grid analysis which takes both the smoother and the coarse-grid correction into account. To perform a two-grid local Fourier analysis, we consider

the fine and coarse grids G_h and G_H. In the transition from the fine to the coarse grid, each low-frequency $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$ is coupled with three high-frequencies, meaning that those three frequencies coincide with the low-frequency on the coarse grid, that is,

$$\varphi_h(\boldsymbol{\theta}^{(\mathbf{0,0})}, \boldsymbol{x}) = \varphi_h(\boldsymbol{\theta}^{(\mathbf{1,1})}, \boldsymbol{x}) = \varphi_h(\boldsymbol{\theta}^{(\mathbf{1,0})}, \boldsymbol{x}) = \varphi_h(\boldsymbol{\theta}^{(\mathbf{0,1})}, \boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in G_H.$$

Interpreting the Fourier components as coarse-grid functions gives

$$\varphi_h(\boldsymbol{\theta}^{(\mathbf{0,0})}, \boldsymbol{x}) = \varphi_H(2\boldsymbol{\theta}^{(\mathbf{0,0})}, \boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in G_H.$$

Because of this, it is appropriate to subdivide the Fourier space into the corresponding four-dimensional subspaces as follows:

Definition 3.19. The four-dimensional space of harmonics is defined by

$$\begin{aligned} \mathcal{F}_h := \operatorname{span} &\left\{ \varphi_h\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}, \cdot\right) : \boldsymbol{\xi} = (\xi_1, \xi_2), \xi_j \in \{0,1\} \right\}, \\ &\text{with } \boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)} \in T^{\text{low}} := \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2 \\ &\text{and } \boldsymbol{\theta}^{\boldsymbol{\xi}} = \boldsymbol{\theta} + \boldsymbol{\xi}\pi, \text{ where } \boldsymbol{\xi} = (0,0), (1,1), (1,0), (0,1). \end{aligned}$$

The four modes $\varphi\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}, \cdot\right)$ (and sometimes also the corresponding frequencies $\boldsymbol{\theta}^{\boldsymbol{\xi}}$) are called harmonics (of each other). We use the ordering of $\boldsymbol{\xi} = (0,0), (1,1), (1,0), (0,1)$ for the four harmonics.

The operator B_h^H, defined in Equation (3.11), intermixes Fourier components with each other. This is a consequence of the fact that the two different grids, G_h and G_H , are involved. However, specifying the symbols for each operator of the two-grid method leads to the invariance property with respect to the space of harmonics $\mathcal{F}_h$. Due to this property, a block-matrix representation of the two-grid operator on the Fourier space can be obtained, which simplifies the computation of the spectral radius of the iteration matrix of the method.

Definition 3.20. Fourier representation of the fine-grid operator
The Fourier representation of the fine-grid operator $A_h \hat{=} [\mathrm{s}_{\boldsymbol{\kappa}}]_h$ w.r.t. $\mathcal{F}_h$ is given by

$$\hat{A}_h(\boldsymbol{\theta}) = \begin{pmatrix} \tilde{A}_h\left(\boldsymbol{\theta}^{(0,0)}\right) & 0 & 0 & 0 \\ 0 & \tilde{A}_h\left(\boldsymbol{\theta}^{(1,1)}\right) & 0 & 0 \\ 0 & 0 & \tilde{A}_h\left(\boldsymbol{\theta}^{(1,0)}\right) & 0 \\ 0 & 0 & 0 & \tilde{A}_h\left(\boldsymbol{\theta}^{(0,1)}\right) \end{pmatrix},$$

with Fourier symbols $\tilde{A}_h\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}\right)$ as defined in Definition 3.11.

Definition 3.21. Fourier representation of the restriction operator
The restriction operator R_h^H maps the four Fourier harmonics onto one coarse-grid function. The Fourier representation of the restriction $R_h^H \hat{=} [r_{\boldsymbol{\kappa}}]_h^H$ w.r.t. $\mathcal{F}_h$ is given by

$$\hat{R}_h^H(\boldsymbol{\theta}) = \begin{pmatrix} \tilde{R}_h^H\left(\boldsymbol{\theta}^{(0,0)}\right) & \tilde{R}_h^H\left(\boldsymbol{\theta}^{(1,1)}\right) & \tilde{R}_h^H\left(\boldsymbol{\theta}^{(1,0)}\right) & \tilde{R}_h^H\left(\boldsymbol{\theta}^{(0,1)}\right) \end{pmatrix},$$

with Fourier symbols $\tilde{R}_h^H\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}\right) := \sum_{\boldsymbol{\kappa}\in\mathbf{V}} r_{\boldsymbol{\kappa}} e^{\imath\boldsymbol{\theta}^{\boldsymbol{\xi}}\cdot\boldsymbol{\kappa}}$, such that
$R_h^H \varphi_h\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}, \cdot\right) = \tilde{R}_h^H\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}\right) \varphi_H\left(2\boldsymbol{\theta}^{(0,0)}, \cdot\right)$.

Definition 3.22. Fourier representation of the prolongation operator
The prolongation operator P_H^h maps one coarse-grid function onto the four Fourier harmonics. The Fourier representation of the prolongation $P_H^h \hat{=}]p_{\boldsymbol{\kappa}}[_H^h$ w.r.t. $\mathcal{F}_h$ is given by

$$\hat{P}_h^H(\boldsymbol{\theta}) = \begin{pmatrix} \tilde{P}_H^h\left(\boldsymbol{\theta}^{(0,0)}\right) \\ \tilde{P}_H^h\left(\boldsymbol{\theta}^{(1,1)}\right) \\ \tilde{P}_H^h\left(\boldsymbol{\theta}^{(1,0)}\right) \\ \tilde{P}_H^h\left(\boldsymbol{\theta}^{(0,1)}\right) \end{pmatrix}$$

with Fourier symbols $\tilde{P}_H^h\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}\right) := \sum_{\boldsymbol{\kappa}\in\mathbf{V}} p_{\boldsymbol{\kappa}} e^{\imath\boldsymbol{\theta}^{\boldsymbol{\xi}}\cdot\boldsymbol{\kappa}}$, such that
$P_H^h \varphi_H\left(2\boldsymbol{\theta}^{(0,0)}, \cdot\right) = \sum_{\boldsymbol{\xi}} \tilde{P}_H^h\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}\right) \varphi_H\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}, \cdot\right)$.

Definition 3.23. Fourier representation of the coarse-grid operator
We distinguish two types of coarse-grid operators. Let $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)}$ and $\boldsymbol{\theta} \in T^{low}$.

1. The Fourier representation of coarse-grid operator $A_H \hat{=} [\mathrm{a}_{\boldsymbol{\kappa}}]_H$ w.r.t. $\mathcal{F}_h$ is given by

$$\tilde{A}_H\left(2\boldsymbol{\theta}^{(0,0)}\right) = \sum_{\boldsymbol{\kappa}\in\mathbf{V}} \mathrm{s}_{\boldsymbol{\kappa}} e^{\imath 2\boldsymbol{\theta}\cdot\boldsymbol{\kappa}},$$

   such that $A_H \varphi_H\left(2\boldsymbol{\theta}^{(0,0)}, \cdot\right) = \tilde{A}_H\left(2\boldsymbol{\theta}^{(0,0)}\right) \varphi_H\left(2\boldsymbol{\theta}^{(0,0)}, \cdot\right)$.

2. The Fourier representation of the Galerkin coarse-grid operator $A_H = R_h^H A_h P_H^h$ w.r.t. $\mathcal{F}_h$ is given by

$$\tilde{A}_H\left(2\boldsymbol{\theta}^{(0,0)}\right) = \tilde{R}_h^H\left(2\boldsymbol{\theta}^{(0,0)}\right) \tilde{A}_h\left(2\boldsymbol{\theta}^{(0,0)}\right) \tilde{P}_H^h\left(2\boldsymbol{\theta}^{(0,0)}\right).$$

Definition 3.24. Invariance property of the two-grid operator
The two-grid operator E_{TG} leaves the harmonic space $\mathcal{F}_h$ invariant for an arbitrary

frequency $\boldsymbol{\theta} \in T^{\text{low}} = [-\frac{\pi}{2}, \frac{\pi}{2})^2$, which means $\mathcal{F}_h$ remains unchanged under E_{TG}. This invariance property results from the following relations:

$$\begin{aligned}
I &: \mathcal{F}_h(\boldsymbol{\theta}) \to \mathcal{F}_h(\boldsymbol{\theta}), \\
A_h &: \mathcal{F}_h(\boldsymbol{\theta}) \to \mathcal{F}_h(\boldsymbol{\theta}), \\
R_h^H &: \mathcal{F}_h(\boldsymbol{\theta}) \to \operatorname{span}\left\{\varphi_H\left(2\boldsymbol{\theta}^{(0,0)}, \cdot\right)\right\}, \\
A_H &: \operatorname{span}\left\{\varphi_H\left(2\boldsymbol{\theta}^{(0,0)}, \cdot\right)\right\} \to \operatorname{span}\left\{\varphi_H\left(2\boldsymbol{\theta}^{(0,0)}, \cdot\right)\right\}, \\
P_H^h &: \operatorname{span}\left\{2\varphi_H\left(\boldsymbol{\theta}^{(0,0)}, \cdot\right)\right\} \to \mathcal{F}_h(\boldsymbol{\theta}), \\
\mathcal{S}_h &: \mathcal{F}_h(\boldsymbol{\theta}) \to \mathcal{F}_h(\boldsymbol{\theta}).
\end{aligned}$$

Summarizing, the Fourier space $\mathcal{F}$ turns out to be invariant under the two-grid operator,

$$E_{TG} : \mathcal{F}_h(\boldsymbol{\theta}) \twoheadrightarrow \mathcal{F}_h(\boldsymbol{\theta}).$$

Definition 3.25. Two-grid convergence factor
The spectral radius of the two-grid iteration matrix and thus the asymptotic two-grid convergence factor can be approximated by

$$\rho(E_{TG}) := \sup_{\boldsymbol{\theta} \in T^{\text{low}}} \rho\left(\hat{E}_{TG}(\boldsymbol{\theta})\right),$$

where

$$\begin{aligned}
\hat{E}_{TG}(\boldsymbol{\theta}) &= \left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_2} \hat{B}_h^H(\boldsymbol{\theta}) \left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_1} \\
&= \left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_2} \left(\hat{I} - \hat{P}_H^h(\boldsymbol{\theta}) \left(\hat{A}_H(2\boldsymbol{\theta})\right)^{-1} \hat{R}_h^H(\boldsymbol{\theta}) \hat{A}_h(\boldsymbol{\theta})\right) \left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_1}
\end{aligned}$$

Example 3.26. We perform a two-grid LFA for the two-dimensional Poisson equation solved by a multigrid method with bilinear interpolation, full-weighting restriction and a Galerkin coarse-grid operator. We perform the two-grid analysis with the GS-LEX smoother. From Example 3.13, we have

$$\tilde{A}_h\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}\right) = \frac{4 - 2\cos(\theta_1 + \xi_1\pi) - 2\cos(\theta_2 + \xi_2\pi)}{h^2},$$

which results in

$$\hat{A}_h(\boldsymbol{\theta}) = \begin{pmatrix}
\frac{4-2\cos\theta_1-2\cos\theta_2}{h^2} & 0 & 0 & 0 \\
0 & \frac{4-2\cos(\theta_1+\pi)-2\cos(\theta_2+\pi)}{h^2} & 0 & 0 \\
0 & 0 & \frac{4-2\cos(\theta_1+\pi)-2\cos\theta_2}{h^2} & 0 \\
0 & 0 & 0 & \frac{4-2\cos\theta_1-2\cos(\theta_2+\pi)}{h^2}
\end{pmatrix}.$$

In addition, we obtain for the bilinear interpolation operator P_H^h (Def. 3.6) the symbols

$$\tilde{P}_H^h\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}\right) = \sum_{\boldsymbol{\kappa}\in\mathbf{V}} p_{\boldsymbol{\kappa}} e^{\imath\boldsymbol{\theta}^{\boldsymbol{\xi}}\cdot\boldsymbol{\kappa}} = (1+\cos(\theta_1+\xi_1\pi))(1+\cos(\theta_2+\xi_2\pi))$$

and thereby

$$\hat{P}_H^h = \begin{pmatrix} (1+\cos\theta_1)(1+\cos\theta_2) \\ (1+\cos(\theta_1+\pi))(1+\cos(\theta_2+\pi)) \\ (1+\cos(\theta_1+\pi))(1+\cos\theta_2) \\ (1+\cos\theta_1)(1+\cos(\theta_2+\pi)) \end{pmatrix}.$$

The symbol for the full-weighting restriction operator (Def. 3.5) is

$$\tilde{R}_h^H\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}\right) = \sum_{\boldsymbol{\kappa}\in\mathbf{V}} r_{\boldsymbol{\kappa}} e^{\imath\boldsymbol{\theta}^{\boldsymbol{\xi}}\cdot\boldsymbol{\kappa}} = \frac{1}{4}\left(1+\cos(\theta_1+\xi_1\pi)\right)\left(1+\cos(\theta_2+\xi_2\pi)\right).$$

We obtain $\hat{R}_h^H$ as introduced in Definition 3.21. From Example 3.18 we know the Fourier symbol of weighted GS-LEX with weight $\omega = 1$ can be represented by

$$\tilde{\mathcal{S}}_h(\boldsymbol{\theta}) = \frac{e^{\imath\theta_1}+e^{\imath\theta_2}}{4-e^{-\imath\theta_1}-e^{-\imath\theta_2}},$$

and the corresponding smoothing factor is $\mu(\omega) = 0.5$. From the representations of $\hat{A}_h$, $\hat{P}_H^h$, $\hat{R}_h^H$, $\hat{I}$ and the solution on the coarse grid $\hat{A}_H = \hat{R}_h^H \hat{A}_h \hat{P}_H^h$, we obtain the representation of $\hat{B}_h^H$, i.e.,

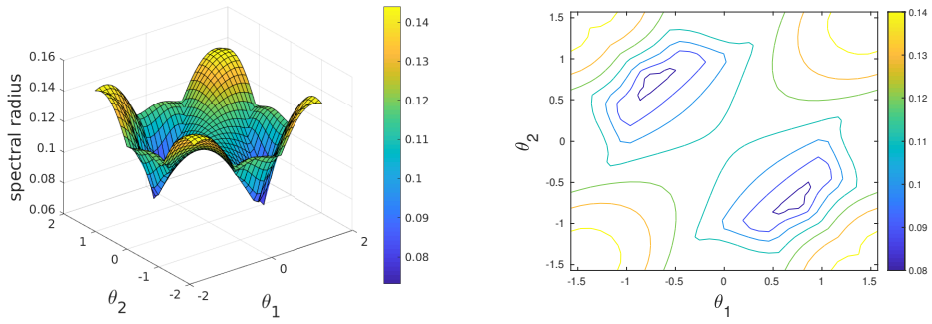
$$\hat{B}_h^H = \hat{I} - \hat{P}_H^h(\boldsymbol{\theta})\left(\hat{A}_H(2\boldsymbol{\theta})\right)^{-1}\hat{R}_h^H(\boldsymbol{\theta})\hat{A}_h(\boldsymbol{\theta}).$$

In order to extend it to $\hat{E}_{TG}$, we include the representations of $\hat{\mathcal{S}}_h$ in the analysis and obtain the two-grid convergence factor

$$\rho(E_{TG}) = \sup_{\boldsymbol{\theta}\in T^{\text{low}}} \rho\left(\hat{E}_{TG}(\boldsymbol{\theta})\right) = \sup_{\boldsymbol{\theta}\in T^{\text{low}}} \rho\left(\left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_2}\hat{B}_h^H(\boldsymbol{\theta})\left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_1}\right) = 0.15.$$

Figure 3.6 shows the convergence behavior of the two-grid method as a function of the frequencies $\boldsymbol{\theta} \in T^{\text{low}}$.

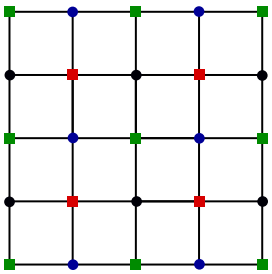
For the GS-RB smoother, the definition of a smoothing factor is not straightforward. The GS-LEX smoother from Example 3.18 has the property that all $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ are eigenfunctions of the smoothing operator. This is not the case for GS-RB. In the two-grid analysis in the last section, we made the more general assumption that only the invariance property is fulfilled for the smoothing operator under consideration, see Definition 3.24. In order to analyze the two-grid method using GS-RB relaxation, we introduce some basic definitions and relations that are crucial to perform a Fourier analysis.



(a) **(b)**

Figure 3.6: The spectral radius of the two-grid symbol with GS-LEX as a function of the frequencies $\boldsymbol{\theta} \in T^{\text{low}}$ for the two-dimensonal Poisson equation.

Lemma 3.27. *We use the ordering of* $\boldsymbol{\xi} = (0,0),(1,1),(1,0),(0,1)$ *for the four harmonics* $\varphi(\boldsymbol{\theta}^{\boldsymbol{\xi}}, \boldsymbol{x})$*. We distinguish four types of grid points within the infinite grid* G_h*, namely*



$$G_h^{(0,0)} := \left\{ \boldsymbol{x} = \boldsymbol{\kappa h} | \boldsymbol{\kappa} \in \mathbb{Z}^2, \kappa_1, \kappa_2 \ \textit{even} \right\},$$
$$G_h^{(1,1)} := \left\{ \boldsymbol{x} = \boldsymbol{\kappa h} | \boldsymbol{\kappa} \in \mathbb{Z}^2, \kappa_1, \kappa_2 \ \textit{odd} \right\},$$
$$G_h^{(1,0)} := \left\{ \boldsymbol{x} = \boldsymbol{\kappa h} | \boldsymbol{\kappa} \in \mathbb{Z}^2, \kappa_1 \ \textit{odd}, \kappa_2 \ \textit{even} \right\},$$
$$G_h^{(0,1)} := \left\{ \boldsymbol{x} = \boldsymbol{\kappa h} | \boldsymbol{\kappa} \in \mathbb{Z}^2, \kappa_1 \ \textit{even}, \kappa_2 \ \textit{odd} \right\}.$$

Due to the relation

$$e^{\imath\theta q} e^{\imath\pi q} = \begin{cases} e^{\imath\theta q} & \textit{for } q \textit{ even}, \\ -e^{\imath\theta q} & \textit{for } q \textit{ odd}, \end{cases}$$

the following relations are valid:

$$\begin{aligned}
\varphi_h\left(\boldsymbol{\theta}^{(1,1)}, \boldsymbol{x}\right) &= \begin{cases} \varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \textit{for } \boldsymbol{x} \in G_h^{(0,0)} \cup G_h^{(1,1)}, \\ -\varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \textit{for } \boldsymbol{x} \in G_h^{(1,0)} \cup G_h^{(0,1)}. \end{cases} \\
\varphi_h\left(\boldsymbol{\theta}^{(1,0)}, \boldsymbol{x}\right) &= \begin{cases} \varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \textit{for } \boldsymbol{x} \in G_h^{(0,0)} \cup G_h^{(0,1)}, \\ -\varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \textit{for } \boldsymbol{x} \in G_h^{(1,0)} \cup G_h^{(1,1)}. \end{cases} \\
\varphi_h\left(\boldsymbol{\theta}^{(0,1)}, \boldsymbol{x}\right) &= \begin{cases} \varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \textit{for } \boldsymbol{x} \in G_h^{(0,0)} \cup G_h^{(1,0)}, \\ -\varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \textit{for } \boldsymbol{x} \in G_h^{(0,1)} \cup G_h^{(1,1)}. \end{cases}
\end{aligned} \tag{3.17}$$

In addition, we define four functions $\Psi_1, \ldots, \Psi_4$*:*

$$\begin{aligned}
\Psi_1(\boldsymbol{x}) &:= \frac{1}{4}\left(\varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) + \varphi_h\left(\boldsymbol{\theta}^{(1,1)}, \boldsymbol{x}\right) + \varphi_h\left(\boldsymbol{\theta}^{(1,0)}, \boldsymbol{x}\right) + \varphi_h\left(\boldsymbol{\theta}^{(0,1)}, \boldsymbol{x}\right)\right), \\
\Psi_2(\boldsymbol{x}) &:= \frac{1}{4}\left(\varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) + \varphi_h\left(\boldsymbol{\theta}^{(1,1)}, \boldsymbol{x}\right) - \varphi_h\left(\boldsymbol{\theta}^{(1,0)}, \boldsymbol{x}\right) - \varphi_h\left(\boldsymbol{\theta}^{(0,1)}, \boldsymbol{x}\right)\right), \\
\Psi_3(\boldsymbol{x}) &:= \frac{1}{4}\left(\varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) - \varphi_h\left(\boldsymbol{\theta}^{(1,1)}, \boldsymbol{x}\right) - \varphi_h\left(\boldsymbol{\theta}^{(1,0)}, \boldsymbol{x}\right) + \varphi_h\left(\boldsymbol{\theta}^{(0,1)}, \boldsymbol{x}\right)\right), \\
\Psi_4(\boldsymbol{x}) &:= \frac{1}{4}\left(\varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) - \varphi_h\left(\boldsymbol{\theta}^{(1,1)}, \boldsymbol{x}\right) + \varphi_h\left(\boldsymbol{\theta}^{(1,0)}, \boldsymbol{x}\right) - \varphi_h\left(\boldsymbol{\theta}^{(0,1)}, \boldsymbol{x}\right)\right).
\end{aligned} \tag{3.18}$$

Using (3.17) *it can be easily verified that*

$$\begin{aligned}
\Psi_1(\boldsymbol{x}) &= \begin{cases} \varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \text{for } \boldsymbol{x} \in G_h^{(0,0)}, \\ 0 & \text{for } \boldsymbol{x} \notin G_h^{(0,0)}. \end{cases} \\
\Psi_2(\boldsymbol{x}) &= \begin{cases} \varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \text{for } \boldsymbol{x} \in G_h^{(1,1)}, \\ 0 & \text{for } \boldsymbol{x} \notin G_h^{(1,1)}. \end{cases} \\
\Psi_3(\boldsymbol{x}) &= \begin{cases} \varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \text{for } \boldsymbol{x} \in G_h^{(1,0)}, \\ 0 & \text{for } \boldsymbol{x} \notin G_h^{(1,0)}. \end{cases} \\
\Psi_4(\boldsymbol{x}) &= \begin{cases} \varphi_h\left(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}\right) & \text{for } \boldsymbol{x} \in G_h^{(0,1)}, \\ 0 & \text{for } \boldsymbol{x} \notin G_h^{(0,1)}. \end{cases}
\end{aligned} \tag{3.19}$$

The general definition of the red-black Gauss-Seidel relaxation operator $\mathcal{S}_h^{RB}$ *is*

$$\mathcal{S}_h^{RB} = \mathcal{S}_h^B \mathcal{S}_h^R,$$

where the corresponding grid $G_h := \{\boldsymbol{x} = \boldsymbol{\kappa}\mathbf{h} \mid \boldsymbol{\kappa} \in \mathbb{Z}^2\}$ *is divided into two disjoint subsets* G_h^R *and* G_h^B*, s.t.*

$$\begin{aligned}
G_h^R &:= \{\boldsymbol{x} = \boldsymbol{\kappa}\mathbf{h} \in G_h \mid \kappa_1 + \kappa_2 \ \textit{even}\} = G_h^{(0,0)} \cup G_h^{(1,1)}, \\
G_h^B &:= \{\boldsymbol{x} = \boldsymbol{\kappa}\mathbf{h} \in G_h \mid \kappa_1 + \kappa_2 \ \textit{odd}\} = G_h^{(1,0)} \cup G_h^{(0,1)},
\end{aligned} \tag{3.20}$$

and

$$\mathcal{S}_h^R \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) = \begin{cases} A(\boldsymbol{\theta})\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) & \text{for } \boldsymbol{x} \in G_h^R, \\ \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) & \text{for } \boldsymbol{x} \in G_h^B. \end{cases}$$

For the sake of clarity, we replace $\varphi^{\boldsymbol{\xi}}$ *by* $\varphi_h\left(\boldsymbol{\theta}^{\boldsymbol{\xi}}, \boldsymbol{x}\right)$*,* $\Psi_j(\boldsymbol{x})$ *by* Ψ_j *and* $A(\boldsymbol{\theta}^{\boldsymbol{\xi}})$ *by* $A^{\boldsymbol{\xi}}$*.*

The combination of (3.17), (3.19) *and* (3.20) *yields, for example with* $\boldsymbol{\xi} = (1,0)$,

$$\begin{aligned}
\mathcal{S}_h^R \varphi^{(1,0)} &= \begin{cases} A^{(1,0)}\varphi^{(1,0)} & \text{for } \boldsymbol{x} \in G_h^R \\ \varphi^{(1,0)} & \text{for } \boldsymbol{x} \in G_h^B \end{cases} \\
&= \begin{cases} A^{(1,0)}\varphi^{(1,0)} & \text{for } \boldsymbol{x} \in G_h^{(0,0)} \\ A^{(1,0)}\varphi^{(1,0)} & \text{for } \boldsymbol{x} \in G_h^{(1,1)} \\ \varphi^{(1,0)} & \text{for } \boldsymbol{x} \in G_h^{(1,0)} \\ \varphi^{(1,0)} & \text{for } \boldsymbol{x} \in G_h^{(0,1)} \end{cases} \\
&= \begin{cases} A^{(1,0)}\varphi^{(0,0)} & \text{for } \boldsymbol{x} \in G_h^{(0,0)} \\ -A^{(1,0)}\varphi^{(0,0)} & \text{for } \boldsymbol{x} \in G_h^{(1,1)} \\ -\varphi^{(0,0)} & \text{for } \boldsymbol{x} \in G_h^{(1,0)} \\ \varphi^{(0,0)} & \text{for } \boldsymbol{x} \in G_h^{(0,1)} \end{cases} \\
&= A^{(1,0)}\Psi_1 - A^{(1,0)}\Psi_2 - \Psi_3 + \Psi_4
\end{aligned}$$

Similar results are obtained for $\boldsymbol{\xi} = (0,0), (1,1), (0,1)$, *such that*

$$\mathcal{S}_h^R \varphi^{\boldsymbol{\xi}} = \begin{cases} A^{(0,0)}\left(\Psi_1 + \Psi_2\right) + \left(\Psi_3 + \Psi_4\right) & \boldsymbol{\xi} = (0,0), \\ A^{(1,1)}(\Psi_1 + \Psi_2) + (-\Psi_3 - \Psi_4) & \boldsymbol{\xi} = (1,1), \\ A^{(1,0)}(\Psi_1 - \Psi_2) + (-\Psi_3 + \Psi_4) & \boldsymbol{\xi} = (1,0), \\ A^{(0,1)}(\Psi_1 - \Psi_2) + (\Psi_3 - \Psi_4) & \boldsymbol{\xi} = (0,1). \end{cases}$$

Inserting $\Psi_1, \ldots, \Psi_4$ *as defined in* (3.18) *yields:*

$$\mathcal{S}_h^R \varphi^{\boldsymbol{\xi}} = \frac{1}{2} \begin{cases} A^{(0,0)}\left(\varphi^{(0,0)} + \varphi^{(1,1)}\right) + \left(\varphi^{(0,0)} - \varphi^{(1,1)}\right) & \boldsymbol{\xi} = (0,0), \\ A^{(1,1)}\left(\varphi^{(0,0)} + \varphi^{(1,1)}\right) + \left(-\varphi^{(0,0)} + \varphi^{(1,1)}\right) & \boldsymbol{\xi} = (1,1), \\ A^{(1,0)}\left(\varphi^{(1,0)} + \varphi^{(0,1)}\right) + \left(\varphi^{(1,0)} - \varphi^{(0,1)}\right) & \boldsymbol{\xi} = (1,0), \\ A^{(0,1)}\left(\varphi^{(1,0)} + \varphi^{(0,1)}\right) + \left(-\varphi^{(1,0)} + \varphi^{(0,1)}\right) & \boldsymbol{\xi} = (0,1). \end{cases}$$

Next, we perform steps that lead us to the representation of the coefficient matrix $\hat{\mathcal{S}}_h^R(\boldsymbol{\theta})$ *as shown in* (3.21). *Therefore, we convert the representation in the following way*

$$\begin{aligned}
A^{(0,0)}\left(\varphi^{(0,0)} + \varphi^{(1,1)}\right) + \left(\varphi^{(0,0)} - \varphi^{(1,1)}\right) &\to A^{(0,0)}\left(\begin{pmatrix}1\\0\\0\\0\end{pmatrix} + \begin{pmatrix}0\\1\\0\\0\end{pmatrix}\right) + \left(\begin{pmatrix}1\\0\\0\\0\end{pmatrix} - \begin{pmatrix}0\\1\\0\\0\end{pmatrix}\right) \\
&= \begin{pmatrix} A^{(0,0)} + 1 \\ A^{(0,0)} - 1 \\ 0 \\ 0 \end{pmatrix}.
\end{aligned}$$

This can be done in a similar way for $\boldsymbol{\xi} = (1,1),(1,0),(0,1)$. *We obtain*

$$\hat{\mathcal{S}}_h^R(\boldsymbol{\theta}) = \frac{1}{2}\begin{pmatrix} A^{(0,0)}+1 & A^{(1,1)}-1 & & \\ A^{(0,0)}-1 & A^{(1,1)}+1 & & \\ & & A^{(1,0)}+1 & A^{(0,1)}-1 \\ & & A^{(1,0)}-1 & A^{(0,1)}+1 \end{pmatrix}, \tag{3.21}$$

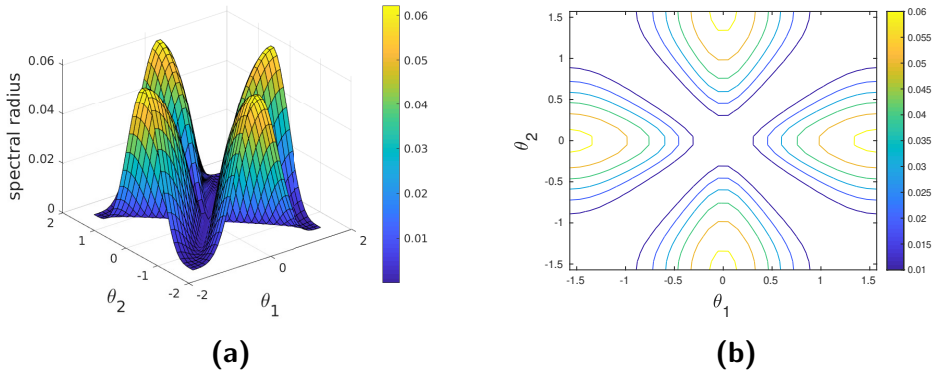
with $A^{\boldsymbol{\xi}} = I - \frac{\omega h^2}{4}\hat{A}_h(\boldsymbol{\theta}^{\boldsymbol{\xi}})$. *Similar steps lead to the representation of the symbol* $\mathcal{S}_h^R$ *corresponding to the performance of the smoother on "black" grid points as shown in* (3.22).

$$\hat{\mathcal{S}}_h^B(\boldsymbol{\theta}) = \frac{1}{2}\begin{pmatrix} A^{(0,0)}+1 & -A^{(1,1)}+1 & & \\ -A^{(0,0)}+1 & A^{(1,1)}+1 & & \\ & & A^{(1,0)}+1 & -A^{(0,1)}+1 \\ & & -A^{(1,0)}+1 & A^{(0,1)}+1 \end{pmatrix}. \tag{3.22}$$

Example 3.28. We perfom a two-grid analysis for the two-dimensional Poisson equation with the red-black Gauss-Seidel smoother (GS-RB). We use the coarse-grid operator $\hat{B}_h^H$ from Example 3.26, namely bilinear interpolation, full-weighting restriction and the Galerkin coarse-grid operator.
From the representation of the GS-RB symbol as given in Lemma 3.27 in combination with the coarse-grid correction symbols, we can compute the two-grid convergence factor. We obtain,

$$\rho(E_{TG}) = 0.0625.$$

Figure 3.7 shows the convergence behavior of the two-grid method in as a function of the frequencies $\boldsymbol{\theta} \in T^{\text{low}}$.



(a) **(b)**

Figure 3.7: The spectral radius of the two-grid symbol with GS-RB as a function of the frequencies $\boldsymbol{\theta} \in T^{\text{low}}$ for the two-dimensonal Poisson equation.

3.3.4 Practical guidelines

A first introduction into practical guidelines on the use of LFA can be found in [23, 71, 75]. An extension of this theory is obtained through the so-called *LFA Lab* by Rittich [64]. The LFA Lab is a software package that provides automated computations of Fourier matrix symbols to simplify the usage of LFA. The automated LFA (aLFA) by Kahl and Kintscher [46] provides a new way of applying LFA that simplifies its application even further. It is carried out in position space and allows the treatment of operators on arbitrary repetitive structures, i.e. it is not limited to simple systems on rectangular grids. Thereby aLFA fully automates the process of LFA and can be used in a wide range of applications and for different kinds of operators. We employ the aLFA tool in Chapter 4 for the analysis of the Stokes equations.
In the following paragraph, we explain the results that are produced by these tools, with a focus on the generated graphics. Furthermore, we evaluate the advantages and disadvantages of available alternatives to visualize the smoothing and two-grid convergence factors without using one of these tools. We highlight a basic fact of practical LFA in Remark 3.29 that we already used in the previous subsections.

Remark 3.29. For practical use, a discrete form of the frequencies $\boldsymbol{\theta}$ resulting from sampling over only a finite set is considered. In this thesis, we discretize the frequency interval $\left[-\frac{\pi}{2}, \frac{3\pi}{2}\right)^2$ into a finite set with 33×33 values.

We recapitulate the main facts of LFA. The smoothing and two-grid convergence factors are indicators for the performance of the multigrid method. The smoother has to eliminate high frequencies. As a result, the analysis of the smoothing behavior is based on the quality of high frequency elimination. An illustration can be found in Figure 3.4. As expected, frequencies $\boldsymbol{\theta} \in T^{\text{low}} = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$ are not damped by the smoother. The corresponding modes are left out of the computation of the smoothing factor, as seen in Lemma 3.15. Visualization of the smoothing behavior is very consistent throughout different tools and papers. The only inconsistency is the frequency interval. In classical LFA theory, the required interval range is 2π. We may shift the interval in $\mathbb{R}^2$, where common choices are $[-\pi, \pi)^2$ and $\left[-\frac{\pi}{2}, \frac{3\pi}{2}\right)^2$.
Some researchers neglect the smoothing factor and focus on the two-grid method as a whole. The entire method has to eliminate high and low frequencies. Therefore we consider an interval range of 2π for the analysis of the two-grid method. As stated previously, c.f. Section 3.3.3, applying the two-grid operator to the modes $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ with $\boldsymbol{\theta} \in \left[-\frac{\pi}{2}, \frac{3\pi}{2}\right)^2$ intermixes Fourier components with each other; i.e., we can no longer distinguish the Fourier modes. This means the modes corresponding to the frequencies $\boldsymbol{\theta} \in T^{\text{low}} := \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$ coincide with

$\boldsymbol{\theta} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left[\frac{\pi}{2}, \frac{3\pi}{2}\right)$, $\boldsymbol{\theta} \in \left[\frac{\pi}{2}, \frac{3\pi}{2}\right) \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)$ and $\boldsymbol{\theta} \in \left[\frac{\pi}{2}, \frac{3\pi}{2}\right)^2$. In particular, for an exemplary choice of $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ with $\boldsymbol{\theta} = \begin{pmatrix}\theta_1 \\ \theta_2\end{pmatrix} = \begin{pmatrix}-\frac{\pi}{2} \\ -\frac{\pi}{2}\end{pmatrix}$, it holds:

$$\varphi\left(\begin{pmatrix}-\frac{\pi}{2} \\ -\frac{\pi}{2}\end{pmatrix}, \boldsymbol{x}\right) = \varphi\left(\begin{pmatrix}-\frac{\pi}{2} \\ \frac{\pi}{2}\end{pmatrix}, \boldsymbol{x}\right) = \varphi\left(\begin{pmatrix}\frac{\pi}{2} \\ -\frac{\pi}{2}\end{pmatrix}, \boldsymbol{x}\right) = \varphi\left(\begin{pmatrix}\frac{\pi}{2} \\ \frac{\pi}{2}\end{pmatrix}, \boldsymbol{x}\right).$$

Due to this reason, we visualize the convergence behavior in the low mode interval $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$, i.e., we compute and plot the maximum value over the four smoothing values for each frequency $\boldsymbol{\theta} \in T^{\text{low}}$.
Alternatively, it is possible to plot the results on the entire interval, i.e., $\boldsymbol{\theta} \in \left[-\frac{\pi}{2}, \frac{3\pi}{2}\right)^2$. Since we cannot distinguish the modes within this interval, it is not an accurate presentation. However, it shows exactly what we are interested in: the convergence property of the multigrid method. To this end, we have to put more effort into the visualization: we apply the operator to each of the four coinciding basis Fourier modes of the harmonic space $\mathcal{F}_h$. From there, we compute the norm of the resulting vector. This presentation highlights the performance of the operator on the basis modes, while the presentation in this thesis focuses on the "intermixture" of those.
A significant factor for LFA is the Fourier basis that we choose (in this thesis $\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}}$). The performance prediction of the two-grid method that we obtain by LFA is the same for different choices of bases. However, either the same Fourier basis must be used at all times throughout the analysis or we use a simple transformation to combine the analysis of operators that are based on different bases. We discuss more details and further aspects in Chapter 4.

3.4 Classical Algebraic multigrid methods

Geometric multigrid methods were critical to the early development of multigrid methods and still play an important role today. Nevertheless, there are classes of problems for which geometric techniques are too difficult to apply or cannot be used at all. These classes can be addressed by algebraic multigrid (AMG) methods as introduced in [23, 29, 69]. The development of AMG methods started in the 1980s to solve linear systems. They are based on multigrid concepts, but in a way that requires no explicit knowledge of the problem geometry. Therefore, we should use the term multi*level* rather than multi*grid*. For historical reasons, we stick to the term multigrid. One of the motivations for AMG methods is the fact that they are solely based on matrix coefficients. Due to this fact, these methods serve as efficient methods for solving unstructured grid problems and they have been shown to work well over a wide range of applications [71].
The execution of AMG methods is a two-part process. The first part, the so-called *setup* phase, is followed by the *solution* phase. The second part uses the

components constructed in the setup phase in order to solve the original matrix equation in a multigrid-like process. This second part is straightforward and requires no additional description. The setup phase comprises determining a sequence of smaller matrices that serve as coarse-level matrices. In addition, the associated inter-level transfer operators are determined automatically. A detailed description about this classical algebraic multigrid method can be found in one of the next paragraphs.
Since the coarse levels, inter-level transfer operators and coarse-level equations are determined based solely on matrix entries, the coarsening process is fully automatic. This is the main reason for AMG's flexibility in adapting itself to specific requirements of the problem to be solved. Nevertheless, the setup phase causes extra overhead, which makes AMG less efficient than geometric multigrid. Having said that, algebraic multigrid should not be regarded as a competitor for geometric multigrid. Instead, AMG should be regarded as a complementary method that is applicable to solve problems which are out of reach of geometric multigrid. The main conceptual difference between geometric and algebraic multigrid results from the fact that geometric multigrid employs fixed grid hierarchies. Geometric multigrid methods enforce an efficient interplay between smoothing and coarse-grid correction by selecting an appropriate smoothing procedure. On the other hand, AMG methods focus on the selection of appropriate coarser levels and interpolation operators, while using a simple relaxation method.
One part of this thesis concentrates on algebraic multigrid smoothing methods for saddle point systems. The focus on smoothers stems from challenging applications that need more sophisticated smoothers, unlike classical AMG theory which attempts to maintain simple smoothers. We introduce the classical AMG method for scalar elliptic PDEs in the next paragraph, which can easily be modified for systems of PDEs since the ideas behind AMG are more general. This is followed by some insights into developments which expand the classical AMG theory. In Chapter 5 we use this knowledge to construct automatic AMG smoothers for systems of PDEs.

3.4.1 Terminology

In this thesis, we always define the restriction as the transpose of interpolation in AMG settings. In addition, we neglect the superscript and subscript notation, i.e.,

$$R = P^T,$$

and we stick to the Galerkin coarse-grid operator

$$A_H = P^T A P.$$

A smoother is denoted by M, where the smoothing iteration is given in (3.10),

$$\mathbf{e}^{(k+1)} = \left(I - M^{-1}A\right)\mathbf{e}^{(k)} = \mathcal{S}(\omega)\mathbf{e}^{(k)} = \mathcal{S}^k(\omega)\mathbf{e}^{(0)}.$$

We assume that $M^T + M - A$ is symmetric and positive definite (s.p.d.). The following result is well known (and easily seen),

$$M^T + M - A \text{ is s.p.d.} \quad \Leftrightarrow \quad \|I - M^{-1}A\|_A < 1.$$

The operator

$$\tilde{M} = M^T(M^T + M - A)^{-1}M,$$

will be frequently used in the definitions and analysis later on. Note that $(I - \tilde{M}^{-1}A) = (I - M^{-1}A)(I - M^{-T}A)$, hence $\tilde{M}$ is a symmetrized version of the smoother M. The iteration matrix E_{TG} of the standard two-grid method is given by

$$E_{TG} = (I - M^{-T}A)(I - P(P^TAP)^{-1}P^TA)(I - M^{-1}A).$$

3.4.2 Algebraic smoothness

Before we construct an algebraic multigrid hierarchy, we first need a good characterization of smooth error. In AMG, error not eliminated by the smoother is called smooth error, i.e., the error $\mathbf{e}$ fulfills $S\mathbf{e} \approx \mathbf{e}$ and must be eliminated by coarse-grid correction. In contrast to the geometric definition, in the AMG setting, the smooth error may be geometrically oscillatory. We should replace the term *smooth* by *slow to converge*. For historical reasons, we stick to the terms *smooth* and *algebraically smooth* for multigrid and algebraic multigrid. A good example to illustrate the difference between algebraic and geometric smooth error is the diffusion equation (2.2). This example shows that for coefficients $a \ll 1$ the error after some iteration steps of a simple pointwise smoother is geometrically smooth, but for $a = 1$ the error is geometrically oscillatory. It makes perfect sense that such an error can only be effectively reduced by means of a coarser grid if that grid is obtained by coarsening in directions in which the error really changes smoothly in the geometric sense. This coarsening procedure is denoted by coarsening in the direction of geometric smoothness. In addition, interpolation has to treat the discontinuities correctly. Indeed, this is exactly what AMG does. Next, we need to characterize the geometric smoothness in some algebraic way. It is easy to verify that the eigenmodes of the system matrix A corresponding to the smallest eigenvalues are those which typically cause the slowest convergence of relaxation and, therefore, correspond to what we defined as an algebraically smooth error. We call those eigenmodes *small eigenmodes* for short. That means

that smoothing damps large eigenmodes, while coarse-grid correction has to handle the remaining small eigenmodes of A. The smallest of the eigenmodes of A, the so-called *near null-space*, is of particular importance. The assumption in AMG is that only geometrically smooth functions are in the near null-space. This is easy to see for the diffusion equation. Any linear function is in the kernel of the differential operator. The same is true for the discrete differential operator A (away from boundaries). That means that the near null-space of A for this problem consists of any vector that is almost linear. This gives us an algebraic way to characterize geometric smoothness, as we see in the the next section. For applications where the near null-space contains geometrically oscillatory functions (such as electromagnetics), the approach of coarsening in directions of geometric smoothness is not sufficient. More details about such applications are covered in Chapter 5.

3.4.3 Classical AMG

In the following, we introduce the classical AMG algorithm of Brandt, McCormick, Ruge, and Stüben [23, 65, 69]. We highlight the main facts to understand the construction of the classical version and show how we use knowledge of the near null-space to design an AMG algorithm. However, we do not go into detail, since in this thesis we focus on an extension of the classical theory to develop AMG smoothers.
The classical AMG method is based on the assumption that geometrically smooth functions are in the near null-space of A. Geometric smoothness can be characterized in an algebraic way as follows: we assume that A is symmetric and positive definite and has been scaled so that its largest eigenvalue equals 1. Furthermore, let $\mathbf{v}$ be a normalized eigenvector of A (i.e., $||\mathbf{v}|| = 1$) that corresponds to a small eigenvalue of A. By multiplying the eigenvalue problem $A\mathbf{v} = \lambda\mathbf{v}$ with $\mathbf{v}^T$, we see that a small eigenmode (i.e., an eigenmode corresponding to an eigenvalue $\lambda \ll 1$) satisfies

$$\lambda = \mathbf{v}^T A\mathbf{v} \ll 1. \tag{3.23}$$

Due to the assumption that geometrically smooth functions are in the near null-space of A, we can assume that A has row sum zero, which leads to the following estimate

$$\begin{aligned}\mathbf{v}^T A\mathbf{v} &= \sum_{i,j} a_{i,j} v_i v_j = \frac{1}{2}\sum_{i,j}(-a_{i,j})(v_i - v_j)^2 + \sum_i v_i^2 \sum_j a_{i,j} \\ &= \frac{1}{2}\sum_{i,j}(-a_{i,j})(v_i - v_j)^2 = \sum_{i<j}(-a_{i,j})(v_i - v_j) \ll 1.\end{aligned}$$

In addition, we assume that $a_{i,j} > 0$ (which is true for M-matrices [71]). We obtain an algebraic way to detect geometrically smooth error due to the fact that smooth error varies slowly in the direction of relatively large negative coefficients of the matrix. Large coefficients are defined with the so-called *strength of connection* term (Definition 3.30), which measures large coefficients relative to the largest off-diagonal entry in a row.

Definition 3.30. Given a threshold $0 < \zeta \leq 1$, we say that variable u_i strongly depends on variable u_j if

$$-a_{i,j} \geq \zeta \max_{k \neq i} -a_{i,k}.$$

This measure determines which variables are strong representatives of the errors left by relaxation. Definition 3.30 guides the construction of coarse grids. In classical AMG, a coarse grid is a subset of the fine grid and the points are chosen such that the grid is coarsened in the direction of strong connections of the matrix A. The coarsening procedure partitions the grid into *C-points* (points on the coarse grid) and *F-points* (points not on the coarse grid). The algorithm is split into two phases. We determine a strength matrix by assigning weights to each point i based on the strength of connection as defined in Definition 3.30. The first phase consists of creating an independent set of fine-grid points based on the strength matrix. In the second phase additional C-points are choosen to satisfy interpolation requirements. The interpolation scheme as described below requires each pair of strongly connected F-points to be strongly connected to a common C-point.
Since the geometrically smooth error $\mathbf{e}$ is identified by small eigenmodes and the residual $\mathbf{r} = A\mathbf{e}$, we obtain from (3.23) that smooth error can be identified by small residuals,

$$\lambda^2 = \mathbf{e}^T A\mathbf{e} = \mathbf{r}^T\mathbf{r} \ll 1.$$

We assume that $\mathbf{r}$ is not only small but also that,

$$r_i = (A\mathbf{e})_i = \sum_j a_{i,j}e_j = 0. \tag{3.24}$$

We can then define interpolation by rewriting Equation (3.24) at F-point i in terms of the coefficients of A,

$$a_{i,i}e_i = -\sum_{j \in C_i} a_{i,j}e_j - \sum_{j \in F_i} a_{i,j}e_j - \sum_{j \in N_i} a_{i,j}e_j,$$

where the sets C_i, F_i, N_i are defined as follows,

$$\begin{aligned} C_i &: C\text{-points strongly connected to } i, \\ F_i &: F\text{-points strongly connected to } i, \\ N_i &: \text{all points weakly connected to } i. \end{aligned}$$

The set C_i contains the points that the F-point i will interpolate from. Therefore, we just need to rewrite the e_j's in the last two terms with regard to either the interpolatory points in C_i or the F-point i. We obtain an equation that involves only the F-point and its interpolatory points, which we can use directly to define interpolation weights.
The definition of the coarse-grid correction $B = I - P(P^TAP)^{-1}P^TA$ and the interpolation operator P yields the setup phase. We do not describe the solve phase here as it is similar to geometric multigrid.

3.5 New class of algebraic multigrid methods

In this section we concentrate on AMG methods and theory that extend the classical AMG theory introduced in the last section. Although the classical method works remarkably well for a wide variety of problems, some of the assumptions, made in its derivation, limit its applicability. There have been many other AMG algorithms that have been developed to extend the applicability of AMG to new classes of problems, e.g. [20]. Here, we introduce the theory that we need in Chapter 5 to construct AMG components for challenging equations such as second-order definite Maxwell's equations (2.4), [53, 77].
AMG theory drives AMG algorithm development. A basic theoretical concept is the so-called *weak approximation property* that, if satisfied by interpolation, implies convergence of the two-grid algorithm. As shown in Definition 3.31, this approximation property relates the accuracy of interpolation to the spectrum of the system matrix. In other words, eigenmodes with small associated eigenvalues must be interpolated well. Let $Q : \mathbb{R}^n \to \mathbb{R}^n$ be a projection onto range(P), i.e.,

$$Q = PR,$$

holds for some restriction-like operator $R : \mathbb{R}^n \to \mathbb{R}^{n_c}$ such that RP is equal to the identity on $\mathbb{R}^{n_c}$ which we denote by I_c. We think of R as defining the coarse-grid variables, i.e., $\mathbf{u}_c = R\mathbf{u}$. For any vector $\mathbf{v} \in$ range(P), we have $Q\mathbf{v} = \mathbf{v}$. Thus, $I - Q$ can be used to measure the defect of interpolation.

Definition 3.31. (**Weak approximation property**)
Let $Q = PR$ be any projection onto range(P). Then the weak approximation property is satisfied if there is a constant K such that

$$\frac{\langle (I-Q)\mathbf{v}, (I-Q)\mathbf{v}\rangle}{\langle A\mathbf{v}, \mathbf{v}\rangle} = \frac{\|(I-Q)\mathbf{v}\|_2^2}{\|\mathbf{v}\|_A^2} \leq K \quad \forall \mathbf{v} \in \mathbb{R}^n \backslash \{0\}. \tag{3.25}$$

An important characteristic of inequality (3.25) is that an eigenvector $\mathbf{v}$ of matrix A corresponding to a small eigenvalue causes a small denominator. Therefore, the

weak approximation property, i.e., the inequality (3.25) is fulfilled for some K if the numerator is also small. The weak approximation property, as introduced in Definition 3.31, implies that the elimination of small eigenmodes is solely based on the interpolation operator. That means it is limited to simple pointwise smoothers and a particular type of coarse grids. Due to this limitation, a new approximation theory was developed in [30] and holds for a broader class of problems and algorithms. The development of this theory is motivated by Maxwell's equations for which more sophisticated smoothers are needed. The extended approximation property is given in the next definition.

Definition 3.32. Let $Q = PR$ be any projection onto range(P) and let $\tilde{M}$ be a smoothing operator. Then the weak approximation property is satisfied if there is a constant K such that

$$\frac{\|(I-Q)\mathbf{v}\|_{\tilde{M}}^2}{\|\mathbf{v}\|_A^2} \leq K \quad \forall \mathbf{v} \in \mathbb{R}^n \backslash \{0\}.$$

Theorem 3.33 gives a convergence result by means of this definition.

Theorem 3.33. *Let $Q = PR$ be any projection onto range(P) and let $\tilde{M}$ be a smoothing operator. Assume that the approximation property in Definition 3.32 is satisfied for some constant*

$$K = \sup_{\mathbf{v}} \frac{\|(I-Q)\mathbf{v}\|_{\tilde{M}}^2}{\|\mathbf{v}\|_A^2}.$$

Then $K \geq 1$ and

$$\|E_{TG}\|_A^2 \leq 1 - \frac{1}{K}.$$

The proof of Theorem 3.33 can be found in [30]. Theorem 3.33 gives conditions that the interpolation operator P must satisfy in order to achieve fast convergence of the multigrid method. Here, small eigenmodes $\mathbf{v}$ must either be interpolated well by P or they must be handled by the smoother. The goal in practice is to minimize K while maintaining sparsity in P. The "best" P possible that fulfills Theorem 3.33 and therefore gives good AMG convergence is called *ideal interpolation operator* and is denoted by P_0. P_0 as given in Lemma 3.34 is the absolute minimizer of K. Although P_0 is usually not sparse, it often serves as a good guidance to generate practical approaches for building P. Define $S : \mathbb{R}^{n_s} \to \mathbb{R}^n$, where $n_s = n - n_c$, such that $RS = 0$. We think of range(S) as the "smoother space", i.e., the space on which the smoother must be effective. Note that S is not unique (but range(S) is). The variables, $S^T\mathbf{u}$, are analogous to F-points. Note also that S and R^T define an orthogonal decomposition of $\mathbb{R}^n$. That is, any

vector $\mathbf{v}$ can be written as $\mathbf{v} = S\mathbf{v}_s + R^T\mathbf{v}_c$ for some $\mathbf{v}_s \in \mathbb{R}^{n_s}$ and $\mathbf{v}_c \in \mathbb{R}^{n_c}$. In the classical AMG setting, the coarse variables are a subset of the fine variables. That is, in the context of the above framework,

$$R = \begin{pmatrix} 0 & I \end{pmatrix}, \quad S = \begin{pmatrix} I \\ 0 \end{pmatrix}. \tag{3.26}$$

Here, the unknowns are assumed to be ordered so that the matrix has the block form

$$A = \begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix}.$$

Lemma 3.34. *Assume we are given a coarse grid and Definition 3.32 is satisfied for some K. Then, the absolute minimizer of Theorem 3.33 is given by*

$$P_0 = (I - S(S^TAS)^{-1}S^TA)R^T = \begin{pmatrix} -A_{ff}^{-1}A_{fc} \\ I \end{pmatrix}.$$

P_0 is called the ideal interpolation operator.

The proof of Lemma 3.34 can be found in [30]. A general form of the absolute minimizer P_0 as introduced in [19] is given by

$$P_* = (I - S(S^TAS)^{-1}S^TA)R^T(RR^T)^{-1}. \tag{3.27}$$

With the assumption that R is normalized such that $RR^T = I$ it holds that $P_0 = P_*$. This assumption is fulfilled by the choice of R in (3.26). However it limits the operators that can be represented as seen in [19]. The generalized formula for P_* overcomes this limitations and enables the following theoretical considerations, c.f. [19]. A sharp theory was developed in [31] that gives additional insight for two-grid convergence theory and development of AMG methods. The sharp theory is very similar in form to Theorem 3.33,

Theorem 3.35. *Let $\tilde{M}$ be a smoother and $\Pi_{\tilde{M}}(P) := P(P^T\tilde{M}P)^{-1}P^T\tilde{M}$ be any projection onto range(P). Assume that*

$$K_\# = \sup_{\mathbf{v}} \frac{\|(I - \Pi_{\tilde{M}}(P))\mathbf{v}\|_{\tilde{M}}^2}{\|\mathbf{v}\|_A^2}.$$

Then, $K \geq 1$ and

$$\|E_{TG}\|_A^2 = 1 - \frac{1}{K_\#}.$$

The proof of Theorem 3.35 can be found in [31]. Based on this result, necessary and sufficient conditions for constructing the components of efficient algebraic multigrid (AMG) methods have been established in [31]. The main foundation for constructing efficient two-grid methods is that we need complementary smoothing and coarse-grid operators. Consequently, the so-called *optimal interpolation operator* was developed in [19]. Using Theorem 3.35 it is straightforward to derive the optimal two-grid convergence rate $||E_{TG}||_A^2$ with respect to P for a given smoother M. This general form of optimal P is derived as introduced in the following lemma:

Lemma 3.36. *Let $P : \mathbb{R}^{n_c} \to \mathbb{R}^n$ be full rank and let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ denote the eigenvalues and orthonormal eigenvectors of the generalized eigenvalue problem*

$$A\mathbf{v} = \lambda \tilde{M}\mathbf{v}.$$

Then the minimal convergence rate of the two-grid method is given by

$$\|E_{TG}(P_\#)\|_A^2 = 1 - \frac{1}{K_\#}, \quad K_\# = \frac{1}{\lambda_{n_{c+1}}},$$

where the optimal interpolation operator $P_\#$ satisfies

$$range(P_\#) = range((\mathbf{v}_1 \cdots \mathbf{v}_{n_c})).$$

For the sake of definiteness we set $P_\# = (\mathbf{v}_1 \cdots \mathbf{v}_{n_c})$ throughout the thesis.

The proof of Lemma 3.36 can be found in [19]. The ideal interpolation operator P_* is different in general from the optimal operator $P_\#$ and this difference can lead to substantial changes in convergence rates of the resulting two-grid method in certain cases. Chapter 5 shows how we use the different theoretical approaches to develop algorithms for systems of equations such as Maxwell's equations.

3.6 Parallel multigrid methods

Generally speaking, parallel programming is used to make execution of software faster. That means, we decrease simulation time by using additional resources simultaneously. This enables the solution, modeling and understanding of large real world phenomena that cannot be achieved by serial computing, due to time consuming simulations or the lack of memory. A compact overview about parallel computing can be found in [8].
In parallel computing, a computational task is typically broken down into several similar subtasks that can be processed independently. Therefore, these subtasks

can be solved simultaneously by a given number of processors. Afterwards, the results are combined to obtain the overall solution. Although parallel and serial computing demand different considerations, it is likely that efficient numerical algorithms for parallel computing will come from the parallelization of algorithms that are already very efficient in the serial environment. It is needless to say that the underlying hardware architecture influences the execution of algorithms. However, in the scope of this thesis we ignore the effect of different parallel computer architectures regardless. An important point to consider in parallel computing is the communication of data between processors. Assuming that each processor has its own memory not shared by others, there are usually several points where information must be exchanged. The same is true for some components in a multigrid algorithm, such as relaxation and residual calculation. The goal in parallel computing is to reduce communication costs as far as possible to avoid so called *communication overhead*. Therefore, it is desirable to keep the number of communications at a low level and achieve a good load-balancing between processors to avoid idle-time.
Section 3.2.3 shows that multigrid methods require $\mathcal{O}(n)$ operations to solve a sparse linear system with n unknowns. As a consequence, they have good scaling potential on parallel computers, since the work per processor can be bounded as the problem size and number of processors are proportionally increased. The parallelization approaches for multigrid algorithms can be classified into two categories. The simpler and more common approach involves no significant changes to the overall structure of the basic multigrid algorithm. Here, parallel computations are executed within each level while the different grids (or levels) are still processed in sequential order. The second approach is elementarily different and seeks to overcome the parallel limitation inherent in the sequential processing of different grids. However, in general this approach is more complicated and has been shown to be not as efficient as standard sequential processing of grid levels [44].
In this thesis, we consider solely the parallel computations within levels. To be more precise, we focus on serial implementations and discuss parallelization potentials of the specific multigrid components. Some multigrid components may be highly parallel while others are not. A critical multigrid component with respect to parallelism is usually the smoothing procedure. The computation of other components in multigrid can be performed in parallel for all grid points, since computations at different grid points are independent of each other. The parallelization property of smoothing procedures depends on the selected smoothing method. Some of the best smoothers do not parallelize well, e.g., GS-LEX. Since this thesis focuses on smoothing techniques, we compare parallel features of the well-known ω-JAC, GS-LEX and GS-RB smoothers. ω-Jacobi is fully parallel since new values are independent of each other, all updates can be computed simultaneously. GS-LEX instead has dependencies since we want to use the most

recent values of the unknown wherever possible. Therefore, GS-LEX is not satisfactorily parallel. In case of GS-RB each step of the Gauss-Seidel relaxation consists of two steps. In the first step, all red grid points are treated simultaneously and independently. In the second step, all black grid points are treated, using the updated values at the red points. For GS-RB the degree of parallelism is 1/2 times the number of grid points, i.e., it is highly parallel. Discretizations with larger stencils require multicolor Gauss-Seidel relaxation or JAC-RB to obtain good parallelization properties. More details on parallelization properties of smoothers are discussed in Chapter 4, especially in the context of the Stokes equations. We also refer to [6, 27]. The degree of parallelism of multigrid is different on different grid levels (small on coarse grids, high on fine grids) due to the number of processors that can contribute to the computation. On the coarse grids, the degree of parallelism decreases substantially. Methods for parallelizing geometric multigrid have been known for some time and most of algebraic multigrid can be parallelized using existing technology.
The potential for optimality (i.e., convergence independently of problem size) mixed with the potential for parallel implementation makes multigrid an attractive algorithm to solve large-scale problems. For AMG, reasonable scalability can be obtained if both the solve phase and the setup phase are implemented efficiently. While most parts of AMG can be parallelized in a straightforward way, the coarse-grid selection process within the AMG setup is inherently sequential.
Several software packages that contain parallel multigrid methods to solve matrix equations have been developed for both the geometric as well as the algebraic setting.
For parallel geometric multigrid software, we refer to the *ExaStencils* project [50]. Here, the focus lies in the automatic generation of stencil codes for applications with exascale performance. The domain of ExaStencils is multigrid stencil codes on (semi-)structured grids. The software is based on a domain-specific approach with languages at several layers of abstraction. At every layer, the corresponding language expresses not only computational directives but also domain knowledge of the problem.
In the algebraic multigrid setting, we refer to two well-known software packages, the *HYPRE* software library [32] and *RAPtor* [10]. Both are general, high performance algebraic multigrid solvers. HYPRE is a library of linear solvers that offers a comprehensive suite of scalable solvers for large-scale scientific simulation, featuring parallel multigrid methods for both structured and unstructured grid problems. The HYPRE library is highly portable and supports a number of languages. RAPtor is a software framework that supports adaptive mesh refinement. It attempts to provide detailed simulation data and efficiently perform parameter studies at high resolution.

4 Construction of geometric multigrid smoothers for the Stokes equations

Systems of PDEs can be treated by multigrid, usually with an efficiency similar to that of scalar equations. For scalar PDEs, we have scalar unknowns $z_1 : \Omega \to \mathbb{R}$, while systems of PDEs are composed of vector-valued unknowns, i.e., $\mathbf{z} : \Omega \to \mathbb{R}^d$. In this thesis, we focus on a special case of discretized systems of PDEs, namely saddle point systems. Large linear systems of saddle point type arise in a wide variety of applications throughout computational science and engineering. A common saddle point system is the Stokes system, which represents a slow viscous flow. In recent years, there has been a surge of interest in saddle point systems and numerous solution techniques have been proposed for this type of system.

4.1 The Stokes equations

We introduce the Stokes equations in two dimensions here. Let Ω be a domain in $\mathbb{R}^2$ with boundary $\partial\Omega$, and $\mathbf{f}$ be a given vector function that describes an external force. Solving the two-dimensional Stokes equations means finding a fluid velocity $\mathbf{u} = (u, v)$ and pressure p such that the system described in (2.3) is fulfilled. We rewrite the Stokes system in coordinate notation,

$$\Delta u + \partial_{\mathrm{x}_1} p = \mathrm{f}_1, \tag{4.1}$$

$$\Delta v + \partial_{\mathrm{x}_2} p = \mathrm{f}_2, \tag{4.2}$$

$$\partial_{\mathrm{x}_1} u + \partial_{\mathrm{x}_2} v = 0, \tag{4.3}$$

where $\partial_{\mathrm{x}_1} := {}^{\partial}/_{\partial \mathrm{x}_1}$ and $\partial_{\mathrm{x}_2} := {}^{\partial}/_{\partial \mathrm{x}_2}$. The discretization of the Stokes equations, which naturally leads to saddle point systems, causes some difficulty. Instability arises, for example, when using central differences of the first-order derivatives (2.6), if all variables are located at the grid points. One remedy for the instability in the case of the Stokes equations is to use a staggered distribution of unknowns, the so-called Marker-and-Cell (MAC) scheme. The idea of the MAC scheme is to place the unknowns (u, v, p) in different locations. More specifically, the discrete

pressure unknowns p are located in the center of each cell and the discrete values of the velocity unknowns are located on the midpoint of vertical edges (x_1-component u) and on the midpoint of horizontal edges (x_2-component v), as seen in Figure 4.1.

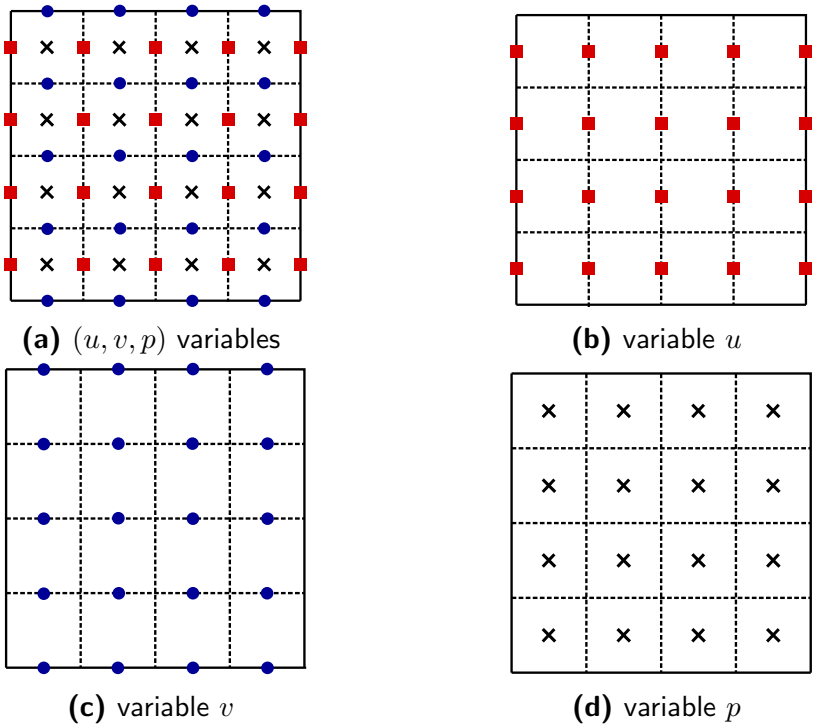


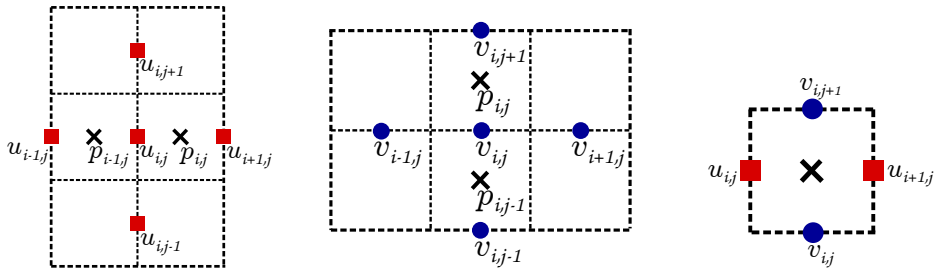
Figure 4.1: Location of unknowns (u, v, p).

The discrete analog of the x_1-coordinate momentum equation (4.1) is defined at vertical edges, the discrete x_2-coordinate momentum equation (4.2) at horizontal edges, and the continuity equation (4.3) at cell centers using central difference schemes. We introduce the following indexing system with regard to the underlying grid: i is the column index and j is the row index, ranging from $1 : n$ or $1 : n + 1$, where n is the number of cells in one direction. The discrete pressure unknowns are $p_{i,j}$, where $i = 1, \ldots, n$, $j = 1, \ldots, n$, and the velocity unknowns are $u_{i,j}$ with $i = 1, \ldots, n+1$, $j = 1, \ldots, n$ and $v_{i,j}$ with $i = 1, \ldots, n$, $j = 1, \ldots, n+1$. Using this indexing system, the MAC scheme can be written as

$$\frac{4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} + \frac{p_{i,j} - p_{i-1,j}}{h} = f_1^{i,j} \tag{4.4}$$

$$\frac{4v_{i,j} - v_{i-1,j} - v_{i+1,j} - v_{i,j-1} - v_{i,j+1}}{h^2} + \frac{p_{i,j} - p_{i,j-1}}{h} = f_1^{i,j} \tag{4.5}$$

$$\frac{u_{i+1,j} - u_{i,j}}{h} + \frac{v_{i,j+1} - v_{i,j}}{h} = 0. \tag{4.6}$$



(a) equation for u **(b)** equation for v **(c)** equation for p

Figure 4.2: Local stencils of MAC scheme.

Then, the discretization of the entire two-dimensional Stokes system with staggered distribution of unknowns has the form

$$\begin{pmatrix} A_1 & & B_1 \\ & A_1 & B_2 \\ -B_1^T & -B_2^T & \end{pmatrix} \begin{pmatrix} u \\ v \\ p \end{pmatrix} = \begin{pmatrix} \mathrm{f}_1 \\ \mathrm{f}_2 \\ 0 \end{pmatrix}, \tag{4.7}$$

where A_1 denotes the negative discrete Laplace operator, $A_1 = -\Delta_h$, and B_1^T and B_2^T are the discrete divergence operators, $B_1^T = (\partial_{\mathrm{x}_1})_{h/2}$ and $B_2^T = (\partial_{\mathrm{x}_2})_{h/2}$. The corresponding system matrix thus reads

$$A = \begin{pmatrix} A_1 & & B_1 \\ & A_1 & B_2 \\ -B_1^T & -B_2^T & \end{pmatrix} = \begin{pmatrix} -\Delta_h & & (\partial_{\mathrm{x}_1})_{h/2} \\ & -\Delta_h & (\partial_{\mathrm{x}_2})_{h/2} \\ -(\partial_{\mathrm{x}_1})_{h/2} & -(\partial_{\mathrm{x}_2})_{h/2} & \end{pmatrix}. \tag{4.8}$$

One of our interests in this work is to develop efficient multigrid methods to solve the Stokes equations discretized by the MAC scheme. We discuss how the multigrid components of smoothing, restriction, interpolation and solution on the coarsest grid are generalized to saddle point systems. Again, a key point of an efficient multigrid method is the right choice of the relaxation procedure. The design of efficient smoothers for solving systems of PDEs often requires special attention. The relaxation method should smooth the error for all unknowns in the equations, which is not an easy task. Therefore, we focus on constructing appropriate relaxation schemes to smooth the error of the unknown functions. The other multigrid components can immediately be extended to systems of PDEs.
Although this gives us a stable discretization, a zero diagonal block appears in the discrete system. For multigrid, this zero block hampers a basic numerical treatment of the problem, which would be to relax the discrete equations directly in a decoupled way. More specifically, decoupled iterative solution methods use blocks on the main diagonal in an equation-wise fashion. Unfortunately, this equation-wise relaxation is not possible due to the zero diagonal block. Moreover, a first obvious choice for these so-called *strong off-diagonal operators* in the differential system is collective smoothing: it updates all unknowns in the system at

a certain grid point simultaneously. However, for staggered grids the unknowns are not defined at the same locations. It is thus not immediately clear how to define standard collective relaxation. In summary, we obtain a challenging system of equations that is not straightforward to solve. Due to the properties of the Stokes equations, we need to put more effort into the construction of multigrid smoothers.
We present a generalization of collective relaxations, so-called *box relaxation*, in Section 4.2.2. Another approach is to use *distributive relaxation*, as described in Section 4.2.2, which can be regarded as a generalization of decoupled relaxation. Other discretization strategies for the Stokes system that are based on non-staggered grids have been developed [71]; these are beyond the scope of this thesis. We start the discussion on multigrid methods for staggered discretizations with a description of some appropriate transfer operators. Afterwards, we return to our main interest: suitable smoothing procedures.

4.2 Multigrid components for the Stokes equations on staggered grids

4.2.1 Transfer operators

Transfer operators on staggered grids depend on the relative locations of the unknowns with respect to the fine grid and the coarse grid, as seen in Figures 4.3 and 4.4. We use transfer operators that are well-known for the Stokes system. The restriction operator R_h^H for each of the three equations in the system can be obtained separately in a straightforward generalization of the scalar case. It is dictated by the staggered grid as follows: at u- and v-grid points, we consider six-point restrictions, and at p-grid points, four-point cell-centered restrictions defined as follows:

Definition 4.1. The restriction operators for the Stokes equations on staggered grids are given by

$$R_h^H\big|_u = \begin{bmatrix} \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ & * & \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \end{bmatrix}, \quad R_h^H\big|_v = \begin{bmatrix} \frac{1}{8} & & \frac{1}{8} \\ \frac{1}{4} & * & \frac{1}{4} \\ \frac{1}{8} & & \frac{1}{8} \end{bmatrix}, \quad R_h^H\big|_p = \begin{bmatrix} \frac{1}{4} & & \frac{1}{4} \\ & * & \\ \frac{1}{4} & & \frac{1}{4} \end{bmatrix},$$

where $*$ denotes the resulting coarse-grid point.

Let us motivate the choice of the restriction operator $R_h^H\big|_u$ that transfers a degree of freedom corresponding to x-component u, as visualized in Figure 4.3. Point

0 corresponds to a degree of freedom u on the coarse grid, and points $1, \ldots, 6$ correspond to degrees of freedom on the fine grid. The restriction operator maps the degrees of freedom that correspond to the fine-grid points to the coarse-grid point with the weighting factors from Definition 4.1.

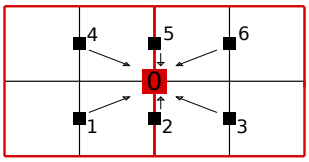


Figure 4.3: Sketch of a restriction operator for a degree of freedom that corresponds to the velocity x-component u denoted by 0.

The derivations of the restriction operators $R_h^H\big|_v$ and $R_h^H\big|_p$ are similar. Note that $R_h^H\big|_u$ and $R_h^H\big|_v$ are only defined for interior edges. When considering Dirichlet boundary conditions, we have prescribed values for the dofs on boundary edges. For the prolongation operator P_H^h, there are two appropriate interpolation schemes: the transpose of the restriction operator and bilinear interpolation. Let us focus again on the degrees of freedom that correspond to the velocity x-component u. Then, bilinear interpolation of neighboring coarse-grid unknowns in the staggered grid can be applied as described in Definition 4.2. We have four types of fine-grid dofs in relation to coarse-grid dofs. A visualization can be found in Figure 4.4. Point 0 is a degree of freedom of u on the fine grid and points $1, \ldots, 4$ or $1, 2$ are degrees of freedom on the coarse grid. The prolongation operator maps the dofs that correspond to coarse-grid points to one fine-grid point with the weighting factors from Definition 4.2.

Definition 4.2. Let u^H be the coarse-grid function that approximates the fine-grid function u^h. The bilinear interpolation operator for the Stokes equations on staggered grids for the degrees of freedom u is defined by

$$
\begin{aligned}
u^h &= \frac{3}{8}u_1^H + \frac{3}{8}u_2^H + \frac{1}{8}u_3^H + \frac{1}{8}u_4^H \\
u^h &= \frac{1}{8}u_1^H + \frac{1}{8}u_2^H + \frac{3}{8}u_3^H + \frac{3}{8}u_4^H \\
u^h &= \frac{3}{4}u_1^H + \frac{1}{4}u_2^H \\
u^h &= \frac{1}{4}u_1^H + \frac{3}{4}u_2^H .
\end{aligned}
$$

The derivation for the prolongation operators for function v is similar. We use piecewise constant interpolation for the pressure unknown p. Note that for the prolongation, we must take boundary condition corrections, at the domain boundaries into account [26].

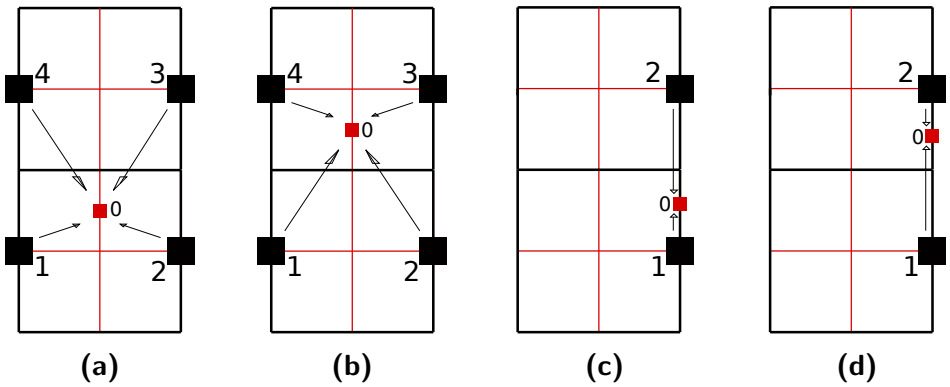


Figure 4.4: Sketch of a prolongation operator for a degree of freedom corresponding to x-component u denoted by 0.

Analogously to the scalar case, the solution on the coarsest grid can be obtained with any suitable solver. However, the discrete system on the coarsest grid may be much larger than in the scalar case. Therefore, the efficiency of a numerical algorithm used for solving the coarsest grid problem may be more important than for scalar equations. We compare rediscretization and the Galerkin coarse-grid operator defined in (3.12) in the course of this chapter.

4.2.2 Relaxation methods

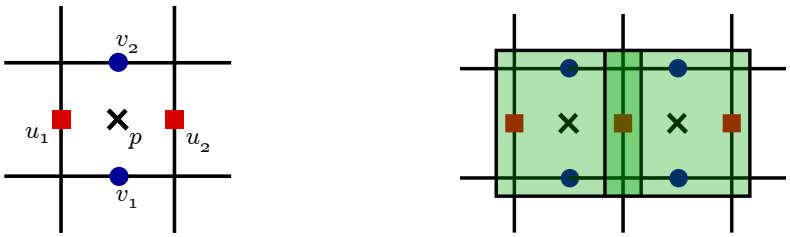
A general form of a linear relaxation or smoothing method for scalar PDEs is given by the error propagation operator

$$\mathcal{S}_h(\omega) := I - M_h^{-1} A_h, \tag{4.9}$$

where M_h represents the interation matrix of the relaxation scheme, c.f. (3.10). This representation may not always be feasible for systems of PDEs. For the Stokes equations discretized on staggered grids, two general types of relaxation methods can be distinguished, *box relaxation* and *decoupled relaxation*.

Box relaxation

The basic idea of box relaxation is to solve the discrete Stokes equations locally "cell by cell" (or "box by box"). This means all five unknowns of one box are updated collectively, involving the respective four momentum equations at the cell boundaries and one continuity equation in the center of the box, c.f. Figure 4.5 and 4.2 and Equations (4.4)-(4.6). This results in an overlapping updating process of 5×5 systems of equations. This method was first introduced in [73].



(a) All five unknowns of one box **(b)** Two overlapping boxes

Figure 4.5: Unknowns updated collectively by box relaxation.

Using this scheme, each velocity component is updated twice and the pressure once per relaxation. Here, one relaxation or one relaxation step refers to relaxing each box once, while one box relaxation refers to one relaxation of one box. A well-known process is relaxation in a Gauss-Seidel manner (lexicographically or with an appropriate coloring). That means, after smoothing all unknowns of one cell collectively, we use the updated values for the relaxation of the next cell. Due to the overlap of the collectively updated unknowns, it is not possible to describe this relaxation scheme in the form given in (4.9). More details are given in Section 4.4.2. This relaxation method is commonly referred to as *overlapping Schwarz smoother* or *symmetric coupled Gauss-Seidel* (SCGS). Especially in the context of the Stokes equations, we use the term *Vanka smoother*. Box relaxation turns out to have robust smoothing properties [71].

Decoupled relaxation

While the box smoothing method is clearly collective in type, it differs from another general class of relaxation methods on staggered grids, so-called *decoupled* or *noncollective relaxation* schemes. For these kind of smoothers, the error propagation operator $S_h(\omega)$ is given in a slightly different form than introduced in (4.9). The so-called *distributive relaxation* approach is a well-known class of noncollective relaxation schemes as described in [17, 74, 76]. It consists of a predictor-corrector sequence of updating steps starting with a transformation of the original system such that a decoupled smoother can be applied afterwards. A general way to "transform" smoothers can be described in the following way: Instead of solving $A\mathbf{u} = \mathbf{f}$, one may solve

$$Z_1 A Z_2 \hat{\mathbf{u}} = \mathbf{f}.$$

Here, we choose Z_1 and Z_2 such that the splitting

$$Z_1 A Z_2 = M - N \tag{4.10}$$

leads to a convergent iterative method. As introduced in Section 3.1, an iterative method with a splitting $A = \tilde{M} - \tilde{N}$ for a system $A\mathbf{u} = \mathbf{f}$ can be given by

$\tilde{M}\mathbf{u}^{(k+1)} = \tilde{N}\mathbf{u}^{(k)} + \mathbf{f}$. With the splitting (4.10) we have $A = Z_1^{-1}MZ_2^{-1} - Z_1^{-1}NZ_2^{-1}$, which leads to

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - Z_2M^{-1}Z_1(A\mathbf{u}^{(k)} - \mathbf{f}).$$

If $Z_1 = I$, we have an *r-transforming iteration* which is a general form of distributive relaxation. Here we have $AZ_2\hat{\mathbf{u}} = \mathbf{f}$ instead of $A\mathbf{u} = \mathbf{f}$, i.e. we use $\mathbf{u} = Z_2\hat{\mathbf{u}}$. An algorithmic formulation of the distributive relaxation can be found in Algorithm 4.1. The distributive Gauss-Seidel (DGS) relaxation scheme for saddle point systems is given by

$$A = \begin{pmatrix} A_1 & A_3 \\ A_2 & \end{pmatrix}, \quad Z_2 = \begin{pmatrix} I & A_1^{-1}A_3(A_2A_1^{-1}A_3)^{-1}V \\ & -(A_2A_1^{-1}A_3)^{-1}V \end{pmatrix} \quad \Rightarrow AZ_2 = \begin{pmatrix} A_1 & \\ A_2 & V \end{pmatrix}.$$

In general, we choose V such that the resulting operator AZ_2 is suited for noncollective relaxation. For the Stokes equations, the zero block in A should disappear.

Algorithm 4.1: General Form of distributive relaxation

1 **Relax transformed system**
  $\hat{\mathbf{u}}^{(k+1)} = \hat{\mathbf{u}}^{(k)} + M^{-1}(\mathbf{f} - AZ_2\hat{\mathbf{u}}^{(k)})$
2 **Update**
  $\mathbf{u}^{(k+1)} = Z_2\hat{\mathbf{u}}^{(k+1)} = \mathbf{u}^{(k)} - Z_2M^{-1}(A\mathbf{u}^{(k)} - \mathbf{f})$

Therefore, we use $V = \Delta_h$, where the operator has to be taken with respect to the pressure unknown p. This leads to the operator

$$A = \begin{pmatrix} -\Delta_h & & (\partial_x)_{h/2} \\ & -\Delta_h & (\partial_y)_{h/2} \\ -(\partial_x)_{h/2} & -(\partial_y)_{h/2} & \end{pmatrix}, \quad Z_2 = \begin{pmatrix} I & & (\partial_x)_{h/2} \\ & I & (\partial_y)_{h/2} \\ & & \Delta_h \end{pmatrix}$$

$$\Rightarrow AZ_2 = \begin{pmatrix} -\Delta_h & & \\ & -\Delta_h & \\ -(\partial_x)_{h/2} & -(\partial_y)_{h/2} & -\Delta_h \end{pmatrix}.$$

We use operator AZ_2 to obtain the distributive relaxation method for the Stokes equations on staggered grids, as described in Algorithm 4.1. Note that M is chosen to be an approximation of AZ_2. Here we use a standard GS-LEX type relaxation with respect to the operator AZ_2, for details see [3]. The SIMPLE algorithm [61] (Semi-Implicit Method for Pressure-Linked equations) is another example of a distributive scheme, but it does not serve as a good smoother [14]. A variant of the pressure correction steps in SIMPLE type algorithms leads to Braess-Sarazin smoothing iterations [13, 14, 79], which can be described as follows: we assume the Stokes system has the form,

$$\mathcal{L}\bar{u} = \bar{f}$$
$$\begin{pmatrix} A & B^T \\ B & \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \tag{4.11}$$

We notice that the solution of a system similar to the original Stokes system is more easily determined,

$$\begin{aligned} \mathcal{W}\bar{v} &= \bar{b} \\ \begin{pmatrix} \alpha D & B^T \\ B & \end{pmatrix} \begin{pmatrix} \hat{u} \\ \hat{p} \end{pmatrix} &= \begin{pmatrix} b \\ d \end{pmatrix}. \end{aligned} \tag{4.12}$$

The similarity between $\mathcal{W}$ and $\mathcal{L}$ guides the following considerations, where we assume that $\mathcal{W} \approx \mathcal{L}$.

$$\begin{aligned} \bar{u} &= \mathcal{L}^{-1}\bar{f} \\ \leftrightarrow \bar{u} &= \bar{u} - \mathcal{L}^{-1}\mathcal{L}\bar{u} + \mathcal{L}^{-1}\bar{f} \\ \leftrightsquigarrow \bar{u} &= \bar{u} - \mathcal{W}^{-1}(\mathcal{L}\bar{u} - \bar{f}) \end{aligned}$$

Therefore, (4.11) can be solved by iterations of the form,

$$\begin{pmatrix} u^{(k+1)} \\ p^{(k+1)} \end{pmatrix} := \begin{pmatrix} u^{(k)} \\ p^{(k)} \end{pmatrix} - \begin{pmatrix} \alpha D & B^T \\ B & \end{pmatrix}^{-1} \left[\begin{pmatrix} A & B^T \\ B & \end{pmatrix} \begin{pmatrix} u^{(k)} \\ p^{(k)} \end{pmatrix} - \begin{pmatrix} f \\ g \end{pmatrix} \right]. \tag{4.13}$$

Note that

$$\begin{pmatrix} \alpha D & B^T \\ B & \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{\alpha} D^{-1}(I - B^T Q B D^{-1}) & D^{-1}B^T Q \\ QBD^{-1} & -\alpha Q \end{pmatrix},$$

where we assume D to be the diagonal of the system matrix A and $Q = (BD^{-1}B^T)^{-1}$. Then, the vectors in (4.13) are obtained by implementing the formulas

$$BD^{-1}B^T\hat{p} = BD^{-1}b - \alpha d \quad \text{and} \quad \hat{u} = \frac{1}{\alpha} D^{-1}(b - B^T\hat{p}),$$

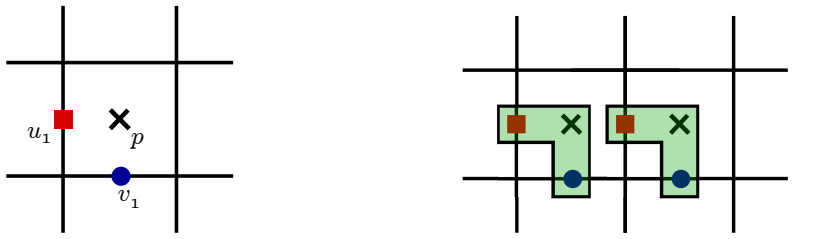
where $b = Au^{(k)} + Bp^{(k)} - f$ and $d = Bu^{(k)} - g$.

Box, DGS and Braess-Sarazin smoothers work well and serve as efficient smoothing methods for the Stokes system [14, 73, 76]. In contrast to the Vanka smoother, the Braess-Sarazin smoother is non-local, i.e., a linear saddle point system for all degrees of freedom has to be solved. For this reason, we call it a global smoother, which might not be as efficient as local coupled methods like the Vanka smoother. The numerical tests presented in Section 4.5 will confirm this statement for the Braess-Sarazin smoother. The main reason for the larger computing times of the Braess-Sarazin smoother are the higher computational costs of one smoothing step. The Vanka smoother is, however, more expensive than DGS. The use of box relaxation may be considered as more straightforward and more convenient

than that of distributive relaxation. Additional smoothers in multigrid methods for solving the Stokes equations as for example the well-known *Uzawa-type smoother* or the *MINRES-type method* have been studied in a number of papers [9, 43, 49, 59, 60, 78]. The analysis of several smoothers based on different discretization techniques can be found in [28, 36, 37, 51, 54]. In the light of the above arguments, we exploit the potential of the Vanka relaxation method and focus on the development of a box smoother that is less expensive but still efficient.

4.2.3 The Triad smoother

The basic idea of the Triad smoother is to update three unknowns collectively, involving two momentum equations at cell boundaries and the continuity equation in the center of the box. This results in a non-overlapping updating process of 3×3 systems of equations, as seen in Figure 4.6.



(a) All three unknowns of one Triad box **(b)** Two non-overlapping boxes

Figure 4.6: Unknowns updated collectively by Triad relaxation.

Using this scheme, only one velocity component in each direction and one pressure component are updated simultaneously. Due to the non-overlapping boxes, the corresponding error-propagation operator $\mathcal{S}_h$ can be represented as introduced in (4.9),

$$\mathcal{S}_h(\omega) := I - M_h^{-1} A_h,$$

where A_h in stencil representation is given by

$$A_h := D_h - L_h - U_h = \begin{bmatrix} -\frac{1}{h^2} & & -\frac{1}{h} \\ & -\frac{1}{h^2} & \end{bmatrix} \begin{bmatrix} \frac{4}{h^2} & & \frac{1}{h} \\ & \frac{4}{h^2} & \frac{1}{h} \\ \frac{1}{h} & \frac{1}{h} & \end{bmatrix} \begin{bmatrix} -\frac{1}{h^2} & & \\ & -\frac{1}{h^2} & \\ -\frac{1}{h} & & \end{bmatrix} - \begin{bmatrix} -\frac{1}{h^2} & & \\ & -\frac{1}{h^2} & -\frac{1}{h} \\ & & \end{bmatrix} \tag{4.14}$$

and M_h can be represented by

$$M_h := \frac{1}{\omega} D_h - L_h = \begin{bmatrix} -\frac{1}{h^2} & & -\frac{1}{h} \\ & -\frac{1}{h^2} & \end{bmatrix} \begin{bmatrix} \frac{4}{h^2} & & \frac{1}{h} \\ & \frac{4}{h^2} & \frac{1}{h} \\ \frac{1}{h} & \frac{1}{h} & \end{bmatrix} - \begin{bmatrix} -\frac{1}{h^2} & & \\ & -\frac{1}{h^2} & -\frac{1}{h} \\ & & \end{bmatrix} \tag{4.15}$$

for a Gauss-Seidel-type updating process and

$$M_h := \frac{1}{\omega} D_h = \frac{1}{\omega} \begin{bmatrix} \frac{4}{h^2} & & \frac{1}{h} \\ & \frac{4}{h^2} & \frac{1}{h} \\ \frac{1}{h} & \frac{1}{h} & \end{bmatrix}, \tag{4.16}$$

for Jacobi-type updating steps. The Triad smoother is also referred to as *unsymmetric coupled Gauss-Seidel* (UCGS) [73]. We stick to the term Triad smoother in this thesis.

For the Stokes system, the Vanka smoother is the most efficient smoother among all of the known ones. However, due to the overlap, computational costs are high and parallel implementation aspects are not satisfying. The Triad smoother seems to be a good alternative to the Vanka smoother, since it has no overlap and therefore reduces computational costs. In addition, this relaxation method has better parallelization properties. These characteristics are examined in detail in the next section. Before developing the symmetric coupled Gauss-Seidel (Vanka) smoother, the unsymmetric coupled (Triad) smoother was initially tried, but it was observed to have poor smoothing and often led to divergence for the Stokes equations [73]. In this thesis we describe the conditions that lead to poor smoothing of the Triad smoother. In addition, we modify the relaxation method to generate an efficient smoother with good smoothing properties.

4.3 Computational work

We start with a comparison of the Vanka smoother and the Triad smoother with regard to computational effort and parallel implementation aspects. Since the Braess-Sarazin smoother is a decoupled smoother, it is easy to see that it requires considerably more communication on parallel computers than coupled multigrid methods with Vanka-type smoothers. Due to this fact, we exclusively focus on the comparison of the Vanka and Triad smoothers.
The obvious difference between these two smoothers is the number of unknowns and equations that are included in one box, while the number of boxes that have to be updated is the same for both smoothers per relaxation. To be more specific, we compute the arithmetic operations that we need for one relaxation step. In addition, we compare some parallel implementation aspects such as the required coloring and the communication effort that is needed for computations in parallel.

Number of arithmetic operations

One box relaxation step consists of several arithmetic operations for updating the corresponding residual and for solving a local system of equations. We give an example of one Vanka box smoothing step. Here, we solve a system of equations that consists of a 5×5 matrix and five unknowns, c.f. Figure 4.5,

$$\begin{pmatrix} \frac{4}{h^2} & -\frac{1}{h^2} & 0 & 0 & \frac{1}{h} \\ -\frac{1}{h^2} & \frac{4}{h^2} & 0 & 0 & -\frac{1}{h} \\ 0 & 0 & \frac{4}{h^2} & -\frac{1}{h^2} & \frac{1}{h} \\ 0 & 0 & -\frac{1}{h^2} & \frac{4}{h^2} & -\frac{1}{h} \\ -\frac{1}{h} & \frac{1}{h} & -\frac{1}{h} & \frac{1}{h} & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ v_1 \\ v_2 \\ p \end{pmatrix} = \begin{pmatrix} f_{11} \\ f_{12} \\ f_{21} \\ f_{22} \\ f_{33} \end{pmatrix}. \tag{4.17}$$

The right-hand side has to be updated to include values that are not included in the matrix system on the left-hand side of the equation. For example, the right-hand side of the first equation of system (4.17) is updated as follows: we indicate unknown u_1 with $u_{i,j}$ to use the indexing system introduced in (4.4) and Figure 4.2a. Then, we have that unknowns $u_{i,j}, u_{i+1,j}$ and $p_{i,j}$ are contained in the left-hand side, and the right-hand side appears as

$$f_{11}^{(k+1)} = f_{11}^{(k)} - \frac{1}{h^2}\left(-u_{i-1,j} - u_{i,j-1} - u_{i,j+1}\right) - \frac{1}{h}\left(-p_{i-1,j}\right).$$

Thus, six arithmetic operations are needed to update the right-hand side of the first equation of system (4.17). One obtains the remaining number of arithmetic operations for the other equations analogously. We use the following well-known

formula from [1] to compute the number of arithmetic operations to solve a system of n linear equations,

$$\text{Total number of arithmetic operations} = \left(\frac{n^3}{3} + \frac{n^2}{2} - \frac{5}{6}n\right) + \left(\frac{n^3}{3} + n^2 - \frac{1}{3}n\right).$$

Table 4.1 shows the number of arithmetic operations that are needed to perform one relaxation step of the Vanka and Triad smoother. N^2 denotes the number of boxes that are included in the considered grid with periodic boundaries.

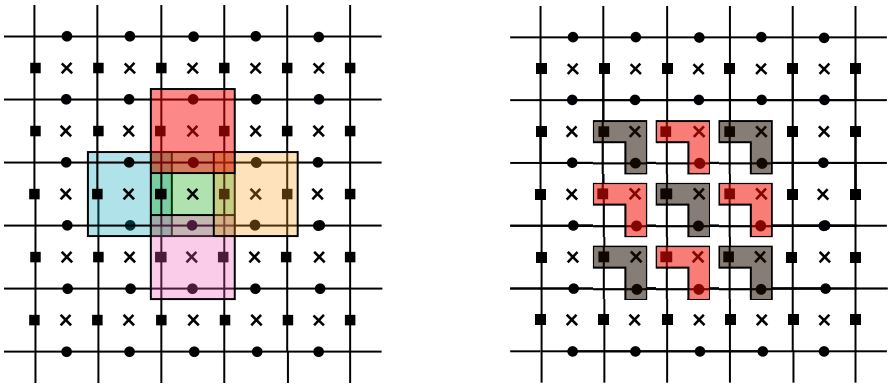
# operations	Vanka smoother	Triad smoother
compute rhs	$24 \cdot N^2$	$16 \cdot N^2$
solve system	$115 \cdot N^2$	$28 \cdot N^2$
Total number	$139 \cdot N^2$	$44 \cdot N^2$

Table 4.1: Number of arithmetic operations for one relaxation step.

We notice that one smoothing step of the Vanka smoother requires approximately as many operations as three smoothing steps of the Triad smoother.

Parallel implementation aspects

The required colorings to perform the smoothing steps in parallel are visualized in Figure 4.7. The Vanka smoother needs five different colors, while the Triad smoother requires a red-black coloring (i.e. two colors) only.



(a) The Vanka smoother **(b)** The Triad smoother

Figure 4.7: Coloring schemes.

An important parallelization aspect is the "dependencies", i.e., all neighboring unknowns that are needed to update an unknown. To perform a method in parallel without read and write conflicts it is necessary to avoid simultaneous updates of dependent unknowns. The coloring scheme is based on depenencies. The unknowns of one Triad box depend on the unknowns of adjacent cells. Consequently, a red-black coloring is sufficient. That means all unknowns of the red boxes can be updated simultaneously and the unknowns of the black Triad boxes can be updated simultaneously. The coloring scheme of the Vanka smoother is illustrated in Figure 4.7a. Due to the additional unknowns that are included in one Vanka box compared to the Triad box, the unknowns do not solely depend on the adjacent boxes. Due to the MAC scheme as given in Equations (4.4) - (4.6) a five coloring is sufficient to perform the Vanka smoother in parallel.
The repetition of the red-black pattern of the Triad smoother is visualized in Figure 4.7b. The five color Vanka pattern is illustrated in Figure 4.8.

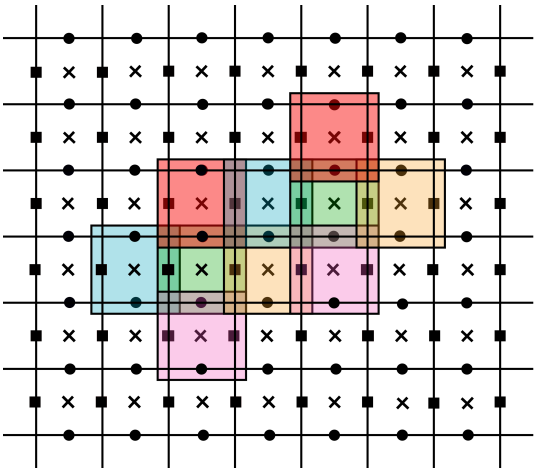


Figure 4.8: Repetition of the coloring pattern of the Vanka smoother.

The communication effort differs due to the different numbers of unknowns that are updated within each box smoothing step. Moreover, the dependence on updated unknowns of neighboring cells leads to further differences. For the Vanka smoother, 12 neighboring cells depend on five updates of the current box smoothing step, c.f. Table 4.2.

	Vanka smoother	Triad smoother
Total number of communications	$60 \cdot N^2$	$12 \cdot N^2$

Table 4.2: Number of communication steps for one relaxation step.

We notice that one smoothing step of the Vanka smoother requires as many communication steps as five smoothing steps of the Triad smoother. These results show the low computational cost in combination with good parallelization properties of the Triad smoother. This motivates us to consider the Triad smoother within a multigrid cycle to solve the Stokes equations. In what follows, we compare the Triad smoother and the Vanka smoother. We analyze both smoothers by means of local Fourier analysis (LFA). In addition, we show numerical results and thereby highlight the impact of boundary conditions on the efficiency of the smoothers. This is followed by an adjustment of the Triad smoother to improve its convergence behavior.

4.4 LFA for the Stokes equations

We employ LFA to analyze the convergence properties of multigrid methods for the Stokes equations and thereby compare the Triad and Vanka smoother.

4.4.1 Terminology

In order to describe LFA for staggered grids, we first introduce some terminology. We consider two-dimensional infinite uniform grids $G_h = G_h^1 \cup G_h^2 \cup G_h^3$ where

$$G_h^j = \{\boldsymbol{x}^j := \boldsymbol{kh} + \delta^j, \boldsymbol{k} \in \mathbb{Z}^2\}, \text{with } \delta^j = \begin{cases} (0, h/2) & \text{if } j = 1, \\ (h/2, 0) & \text{if } j = 2, \\ (h/2, h/2) & \text{if } j = 3, \end{cases} \tag{4.18}$$

c.f. Equation (3.13). Here, the grids that correspond to the velocity unknowns u and v, depicted in Figures 4.1b and 4.1c, are denoted by G_h^1 and G_h^2. Moreover, the grid that corresponds to pressure unknown p is denoted by G_h^3, as visualized in Figure 4.1d. The coarse grid, G_H, is defined similarly. As a reminder, the functions $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ are called Fourier modes or Fourier functions with frequency $\boldsymbol{\theta}$ and form the basis of the Fourier space $\mathcal{F}_h$, see Definition 3.19. For staggered grids, the Fourier basis is given by the span$\{\varphi^1(\boldsymbol{\theta}, \boldsymbol{x}), \varphi^2(\boldsymbol{\theta}, \boldsymbol{x}), \varphi^3(\boldsymbol{\theta}, \boldsymbol{x})\}$ for $\boldsymbol{\theta} \in \left[-\frac{\pi}{2}, \frac{3\pi}{2}\right)^2$, with

$$\varphi^1(\boldsymbol{\theta}, \boldsymbol{x}) = \begin{pmatrix} e^{\imath \boldsymbol{\theta} \boldsymbol{x}^1 / \boldsymbol{h}} & 0 & 0 \end{pmatrix}^T, \varphi^2(\boldsymbol{\theta}, \boldsymbol{x}) = \begin{pmatrix} 0 & e^{\imath \boldsymbol{\theta} \boldsymbol{x}^2 / \boldsymbol{h}} & 0 \end{pmatrix}^T,$$
$$\varphi^3(\boldsymbol{\theta}, \boldsymbol{x}) = \begin{pmatrix} 0 & 0 & e^{\imath \boldsymbol{\theta} \boldsymbol{x}^3 / \boldsymbol{h}} \end{pmatrix}^T.$$

The frequencies $\boldsymbol{\theta}$ do not have to be the same for each basis component φ^j, $j = 1, 2, 3$. However, throughout the course of this thesis we use identical values

for each Fourier basis component. We use the stencil notation and consider the operator A_h of the Stokes system,

$$A_h = \begin{pmatrix} -\Delta_h & & (\partial_{\mathrm{x}_1})_{h/2} \\ & -\Delta_h & (\partial_{\mathrm{x}_2})_{h/2} \\ -(\partial_{\mathrm{x}_1})_{h/2} & -(\partial_{\mathrm{x}_2})_{h/2} & \end{pmatrix},$$

with stencils

$$-\Delta_h = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}, \quad (\partial_{\mathrm{x}_1})_{h/2} = \frac{1}{h}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \quad (\partial_{\mathrm{x}_2})_{h/2} = \frac{1}{h}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}.$$

Each entry in the symbol $\tilde{A}_h$ is computed as the (scalar) symbol of the corresponding block of A_h, following Definition 3.11. Since A_h is a 3×3 block operator, its symbol is a 3×3 matrix,

$$\tilde{A}_h(\boldsymbol{\theta}) = \begin{pmatrix} \frac{1}{h^2}(4 - 2\cos\theta_1 - 2\cos\theta_2) & 0 & 2\imath h \sin\frac{\theta_1}{2} \\ 0 & \frac{1}{h^2}(4 - 2\cos\theta_1 - 2\cos\theta_2) & 2\imath h \sin\frac{\theta_2}{2} \\ -2\imath h \sin\frac{\theta_1}{2} & -2\imath h \sin\frac{\theta_1}{2} & 0 \end{pmatrix}. \tag{4.19}$$

The error-propagation symbol $\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta})$ for a relaxation scheme M_h is given by

$$\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta}) = I - \tilde{M}_h^{-1}(\boldsymbol{\theta})\tilde{A}_h(\boldsymbol{\theta}),$$

c.f. Lemma 3.15, where $\tilde{M}_h$ and $\tilde{A}_h$ are the symbols for M_h and A_h respectively. High and low frequencies for standard coarsening are given by

$$\boldsymbol{\theta} \in T^{\text{low}} := \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2, \quad \boldsymbol{\theta} \in T^{\text{high}} := \left[-\frac{\pi}{2}, \frac{3\pi}{2}\right)^2 \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2,$$

c.f. Definition 3.14. The smoothing factor is given by

$$\mu(\omega) = \mu\left(\mathcal{S}_h(\omega)\right) = \max_{\boldsymbol{\theta} \in T^{\text{high}}} \left\{ \left| \lambda_{\tilde{\mathcal{S}}_h(\omega,\boldsymbol{\theta})} \right| \right\},$$

c.f. (3.15) and Remark 3.16. In other words, the error-propagation operator $\mathcal{S}_h(\omega)$ is block-diagonalized by the matrix of Fourier modes, $\boldsymbol{\Phi}_h$. We index the columns of $\boldsymbol{\Phi}_h$ by the continuous index, $\boldsymbol{\theta}$, and the rows by their spatial location, $\boldsymbol{x}$, such that the entries of the matrix are specified by $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x})$.

Remark 4.3. The influence of boundary conditions is not taken into account when applying LFA, since the modes $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x})$ are defined on the infinite grid G_h. Experience with LFA show that it often serves as an exact prediction tool for problems with periodic boundary conditions, but degradation in performance may be seen with Dirichlet boundary conditions [58], c.f. Section 4.5.

We start with a smoothing analysis of the Vanka and the Triad smoother. This is followed by a two-grid analysis.

4.4.2 Vanka smoother

We start with the smoothing analysis of the Vanka smoother without coloring. Due to the overlapping cells, certain degrees of freedom are updated multiple times over the course of a single sweep of relaxation. Therefore, classical LFA techniques fail. LFA is based on the assumption that the iteration matrix M_h can be written as a Toeplitz operator, c.f. Section 3.3.1. It is not apparent that the same is true for overlapping smoothers.
An extension of the classical LFA analysis for the Vanka smoothers was first introduced by Sivaloganathan in [68]. Unfortunately, the paper includes several misprints. MacLachlan and Oosterlee generalized the LFA techniques to analyze overlapping smoothers for other PDEs and discretizations [51]. Their paper provides theoretical results in order to use the Fourier ansatz. A necessary assumption is that the error-propagation operator for coupled (overlapping) relaxation is block-diagonalized by the Fourier matrix, regardless of the distribution of degrees of freedom within the box. The key step in proving this is to show the following inductive step: if the errors before relaxation on degrees of freedom within a box $V_{i,j}$ satisfy a generalized LFA ansatz, then the errors after relaxation on $V_{i,j}$ satisfy the same ansatz advanced by one cell. MacLachlan and Oosterlee show that the error-propagation matrix for any coupled relaxation is an infinite-grid block-multilevel-Toeplitz matrix [51]. This means that we can attempt to analyze these techniques using classical multigrid smoothing and two-grid Fourier analysis tools to measure the effectiveness of the resulting multigrid cycles.
This thesis provides practical guidelines to obtain the smoothing symbol $\tilde{\mathcal{S}}_h(\boldsymbol{\theta})$ of the Vanka smoother to analyze this relaxation method. For more details and a generalization to other overlapping smoothers, we refer to [51]. One box that consists of the five unknowns visualized in Figure 4.5a can be described by the index set $V_{i,j} = \left\{\left(i, j-\frac{1}{2}\right), \left(i-\frac{1}{2}, j\right), (i,j), \left(i+\frac{1}{2}, j\right), \left(i, j+\frac{1}{2}\right)\right\}$. The update equation for one box is then given by

$$\begin{aligned} U_{i,j}^{\text{new}} = U_{i,j}^{\text{old}} + \omega A_{i,j}^{-1} R_{i,j}^{\text{old}} \quad &\Longleftrightarrow \quad U_{i,j}^{\text{new}} - U_{i,j}^{\text{old}} = \omega A_{i,j}^{-1} R_{i,j}^{\text{old}} \\ &\Longleftrightarrow \quad \frac{1}{\omega} A_{i,j} \left(E_{i,j}^{\text{old}} - E_{i,j}^{\text{new}}\right) = R_{i,j}^{\text{old}}, \end{aligned} \tag{4.20}$$

where $U_{i,j}^{\text{old}}$ and $U_{i,j}^{\text{new}}$ are the approximations before and after relaxation, respectively, $E_{i,j}^{\text{old}}$ and $E_{i,j}^{\text{new}}$ are the errors before and after relaxation, respectively, $R_{i,j}^{\text{old}}$ is the residual before relaxation at the nodes in $V_{i,j}$, $A_{i,j}$ denotes the subsystem connecting the unknowns in $V_{i,j}$, c.f. (4.17), and ω is a relaxation parameter. Before we relax on $V_{i,j}$, the number of times the variables have been updated differs, as visualized in Figure 4.9. The grey box includes the unknowns in $V_{i,j}$ and the number of lines denotes the number of updates, i.e.,

$$\text{Twice:} \qquad \left(i-\tfrac{1}{2}, j-1\right), \left(i+\tfrac{1}{2}, j-1\right), \left(i-\tfrac{3}{2}, j\right), \left(i, j-\tfrac{3}{2}\right),$$

$\left(i-1, j-\frac{1}{2}\right)$

Once: $\left(i-\frac{1}{2}, j\right), \left(i, j-\frac{1}{2}\right), \left(i+1, j-\frac{1}{2}\right), \left(i-1, j+\frac{1}{2}\right), (i, j-1),$

$(i-1, j)$

Not updated: $\left(i+\frac{1}{2}, j\right), \left(i+\frac{3}{2}, j\right), \left(i-\frac{1}{2}, j+1\right), \left(i+\frac{1}{2}, j+1\right), \left(i, j+\frac{1}{2}\right),$

$\left(i+1, j+\frac{1}{2}\right), \left(i, j+\frac{3}{2}\right), (i, j), (i+1, j), (i, j+1)$

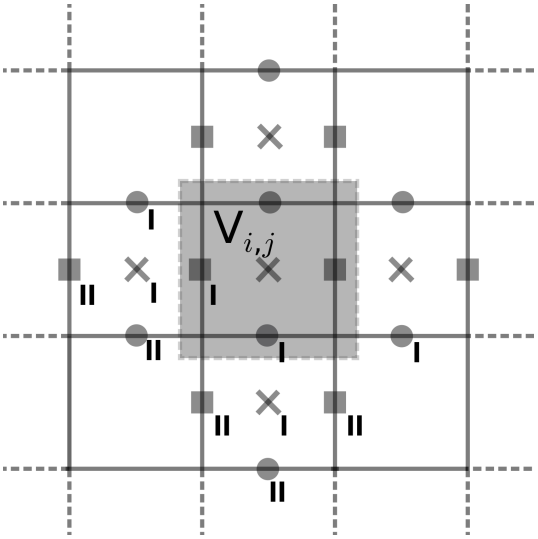


Figure 4.9: Number of updates prior to consider $V_{i,j}$.

Assume that before relaxation, the error components in u, v, p are defined by

$$\mathbf{e}_u = \sigma_u \cdot e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}}, \quad \mathbf{e}_v = \sigma_v \cdot e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}}, \quad \mathbf{e}_p = \sigma_p \cdot e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}},$$

with Fourier coefficients σ. During the smoothing process, the error in terms of the number of updates is given by

$$\mathbf{e}_u = \sigma_u'' e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}}, \quad \mathbf{e}_v = \sigma_v'' e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}}, \quad \mathbf{e}_p = \sigma_p' e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}}, \quad \mathbf{e}_u = \sigma_u' e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}}, \quad \mathbf{e}_v = \sigma_v' e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}/\mathbf{h}}.$$

We substitute these expansions into the residual equations associated with the five unknowns before the relaxation on $V_{i,j}$, where $r_{k,l}$ is the residual in the ap-

proximation to the unknowns for each dof (k,l),

$$\begin{aligned}
r_{i-\frac{1}{2},j} &= \left(\frac{1}{h^2}\left(4\sigma'_u - \sigma''_u e^{-\imath\theta_2} - \sigma''_u e^{-\imath\theta_1} - \sigma_u e^{\imath\theta_1} - \sigma_u e^{\imath\theta_2}\right)\right. \\
&\quad \left.+\frac{1}{h}\left(\sigma_p e^{\frac{1}{2}\imath\theta_1} - \sigma'_p e^{-\frac{1}{2}\imath\theta_1}\right)\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i-\frac{1}{2},j}/\mathbf{h}} \\
r_{i+\frac{1}{2},j} &= \left(\frac{1}{h^2}\left(4\sigma_u - \sigma''_u e^{-\imath\theta_2} - \sigma'_u e^{-\imath\theta_1} - \sigma_u e^{\imath\theta_1} - \sigma_u e^{\imath\theta_2}\right)\right. \\
&\quad \left.+\frac{1}{h}\left(\sigma_p e^{\frac{1}{2}\imath\theta_1} - \sigma_p e^{-\frac{1}{2}\imath\theta_1}\right)\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i+\frac{1}{2},j}/\mathbf{h}} \\
r_{i,j-\frac{1}{2}} &= \left(\frac{1}{h^2}\left(4\sigma'_v - \sigma''_v e^{-\imath\theta_2} - \sigma''_v e^{-\imath\theta_1} - \sigma'_v e^{\imath\theta_1} - \sigma_v e^{\imath\theta_2}\right)\right. \\
&\quad \left.+\frac{1}{h}\left(\sigma_p e^{\frac{1}{2}\imath\theta_2} - \sigma'_p e^{-\frac{1}{2}\imath\theta_2}\right)\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i,j-\frac{1}{2}}/\mathbf{h}} \\
r_{i,j+\frac{1}{2}} &= \left(\frac{1}{h^2}\left(4\sigma_v - \sigma'_v e^{-\imath\theta_2} - \sigma'_v e^{-\imath\theta_1} - \sigma_v e^{\imath\theta_1} - \sigma_v e^{\imath\theta_2}\right)\right. \\
&\quad \left.+\frac{1}{h}\left(\sigma_p e^{\frac{1}{2}\imath\theta_2} - \sigma_p e^{-\frac{1}{2}\imath\theta_2}\right)\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i,j+\frac{1}{2}}/\mathbf{h}} \\
r_{i,j} &= \frac{1}{h}\left(-\sigma'_u e^{-\frac{1}{2}\imath\theta_1} + \sigma_u e^{\frac{1}{2}\imath\theta_1} - \sigma'_v e^{-\frac{1}{2}\imath\theta_2} + \sigma_v e^{\frac{1}{2}\imath\theta_2}\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i,j}/\mathbf{h}}.
\end{aligned}$$

Substituting the appropriate Fourier expansions for the errors before and after relaxation at the nodes in $V_{i,j}$ gives the update equation (4.20),

$$\begin{pmatrix}
\frac{4}{h^2} & -\frac{1}{h^2} & 0 & 0 & \frac{1}{h} \\
-\frac{1}{h^2} & \frac{4}{h^2} & 0 & 0 & -\frac{1}{h} \\
0 & 0 & \frac{4}{h^2} & -\frac{1}{h^2} & \frac{1}{h} \\
0 & 0 & -\frac{1}{h^2} & \frac{4}{h^2} & -\frac{1}{h} \\
-\frac{1}{h} & \frac{1}{h} & -\frac{1}{h} & \frac{1}{h} & 0
\end{pmatrix}
\begin{pmatrix}
\frac{1}{\omega}\left(\sigma'_u - \sigma''_u\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i-\frac{1}{2},j}/\mathbf{h}} \\
\frac{1}{\omega}\left(\sigma_u - \sigma'_u\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i+\frac{1}{2},j}/\mathbf{h}} \\
\frac{1}{\omega}\left(\sigma'_v - \sigma''_v\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i,j-\frac{1}{2}}/\mathbf{h}} \\
\frac{1}{\omega}\left(\sigma_v - \sigma'_v\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i,j+\frac{1}{2}}/\mathbf{h}} \\
\frac{1}{\omega}\left(\sigma_p - \sigma'_p\right) e^{\imath\boldsymbol{\theta}\cdot\boldsymbol{x}_{i,j}/\mathbf{h}}
\end{pmatrix} =
\begin{pmatrix}
r_{i-\frac{1}{2},j} \\
r_{i+\frac{1}{2},j} \\
r_{i,j-\frac{1}{2}} \\
r_{i,j+\frac{1}{2}} \\
r_{i,j}
\end{pmatrix}.$$

Now, this system of five equations can be rearranged into a system of equations directly corresponding to the five updated Fourier coefficients,

$$L \begin{pmatrix} \sigma'_u \\ \sigma'_v \\ \sigma''_u \\ \sigma''_v \\ \sigma'_p \end{pmatrix} = Q \begin{pmatrix} \sigma_u \\ \sigma_v \\ \sigma_p \end{pmatrix}.$$

These equations may then be solved collectively, expressing

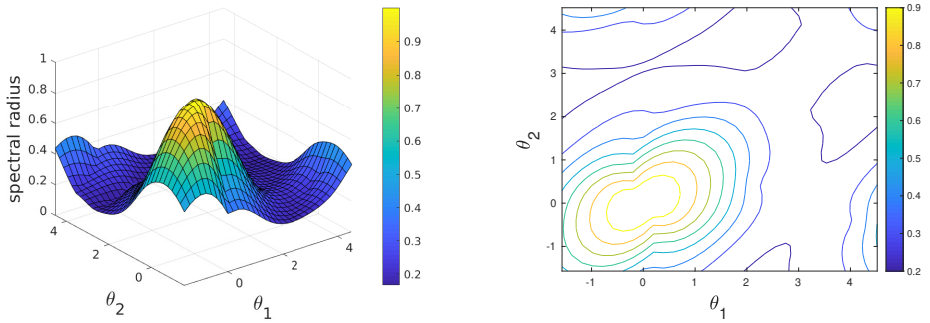
$$\begin{pmatrix} \sigma_u' & \sigma_v' & \sigma_u'' & \sigma_v'' & \sigma_p' \end{pmatrix}^T = L^{-1}Q \begin{pmatrix} \sigma_u \\ \sigma_v \\ \sigma_p \end{pmatrix} = \begin{bmatrix} \mathcal{S}_1 \\ \mathcal{S}_2 \end{bmatrix} \begin{pmatrix} \sigma_u \\ \sigma_v \\ \sigma_p \end{pmatrix},$$

where L is a five-by-five matrix and Q is a five-by-three matrix. The smoothing factor $\mathcal{S}_2$ maps the initial error coefficient of the Fourier mode into that after a sweep of the element-wise overlapping Vanka relaxation. Based on these amplification factors, we can then perform a classical smoothing analysis for the Vanka smoother. Figure 4.10 shows the smoothing factors as a function of the Fourier frequencies, $\boldsymbol{\theta}$, for the Vanka smoother. Computing the smoothing factors

$$\mu = \max_{\boldsymbol{\theta} \in T^{\text{high}}} \left\{ \left| \lambda_{\tilde{\mathcal{S}}_h(\omega,\boldsymbol{\theta})} \right| \right\},$$

where $\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta}) := \mathcal{S}_2$, we get for the Vanka smoother

$$\mu = 0.59.$$



(a) (b)

Figure 4.10: The spectral radius of the smoothing error-propagation symbol for the weighted Vanka relaxation with $\omega = 0.8$ and different frequencies $\boldsymbol{\theta}$.

4.4.3 Triad smoother

The LFA analysis for the Triad smoother is based on a simple expansion of the assumptions of LFA for scalar PDEs. We assume that the matrix A_h is now a block matrix, where each block is an infinite-grid Toeplitz matrix. Then, each block in A_h may be diagonalized with LFA.
We start with a smoothing analysis as described in Section 3.3 for scalar PDEs and

extended in Section 4.4. The smoothing symbol $\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta})$ for Triad-type smoothers is given by

$$\tilde{\mathcal{S}}_h(\omega, \boldsymbol{\theta}) = I - \tilde{M}_h^{-1}(\boldsymbol{\theta})\tilde{A}_h(\boldsymbol{\theta}),$$

with I the identity matrix, $\tilde{A}_h(\boldsymbol{\theta})$ as given in (4.19) and the symbol of the block iteration matrix $\tilde{M}_h$

$$\tilde{M}_h(\boldsymbol{\theta}) = \begin{pmatrix} \frac{1}{h^2}\left(\frac{4}{\omega} - e^{-\imath\theta_1} - e^{-\imath\theta_2}\right) & 0 & \frac{1}{h}\left(\frac{1}{\omega}e^{\frac{1}{2}\imath\theta_1} - e^{-\frac{1}{2}\imath\theta_1}\right) \\ 0 & \frac{1}{h^2}\left(\frac{4}{\omega} - e^{-\imath\theta_1} - e^{-\imath\theta_2}\right) & \frac{1}{h}\left(\frac{1}{\omega}e^{\frac{1}{2}\imath\theta_2} - e^{-\frac{1}{2}\imath\theta_2}\right) \\ \frac{1}{\omega h}e^{-\frac{1}{2}\imath\theta_1} & \frac{1}{\omega h}e^{-\frac{1}{2}\imath\theta_2} & 0 \end{pmatrix},$$

for a Gauss-Seidel updating process, as defined in (4.15), and

$$\tilde{M}_h(\boldsymbol{\theta}) = \frac{1}{\omega}\begin{pmatrix} \frac{4}{h^2} & 0 & \frac{1}{h}e^{\frac{1}{2}\imath\theta_1} \\ 0 & \frac{4}{h^2} & \frac{1}{h}e^{\frac{1}{2}\imath\theta_2} \\ \frac{1}{h}e^{-\frac{1}{2}\imath\theta_1} & \frac{1}{h}e^{-\frac{1}{2}\imath\theta_2} & 0 \end{pmatrix},$$

for a Jacobi updating process, as defined in (4.16). On the basis of parameter studies we figured out that the weighting factor $\omega = 0.8$ promises good smoothing behavior. Moreover, the smoothing analysis results show that the most promising Triad-type smoother is based on a weighted Gauss-Seidel updating process with $\omega = 0.8$ and smoothing factor

$$\mu = 0.87,$$

as seen in Table 4.3. Figure 4.11 shows the smoothing factors of this promising smoother as a function of the Fourier frequencies, $\boldsymbol{\theta}$.

	GS ($\omega = 1$)	GS ($\omega = 0.8$)	Jacobi ($\omega = 1$)	Jacobi ($\omega = 0.8$)
μ	1.00	0.87	1.00	0.92

Table 4.3: Smoothing factors for different Triad-type relaxation methods.

4.4.4 Two-grid analysis

The classical LFA smoothing factor, where the coarse-grid correction is assumed to be an ideal operator that annihilates the low-frequency error components and leaves the high-frequency components unchanged, serves as a good indication for the multigrid convergence behavior. Nevertheless, it sometimes fails to accurately predict the multigrid performance. Therefore, we additionally analyze the

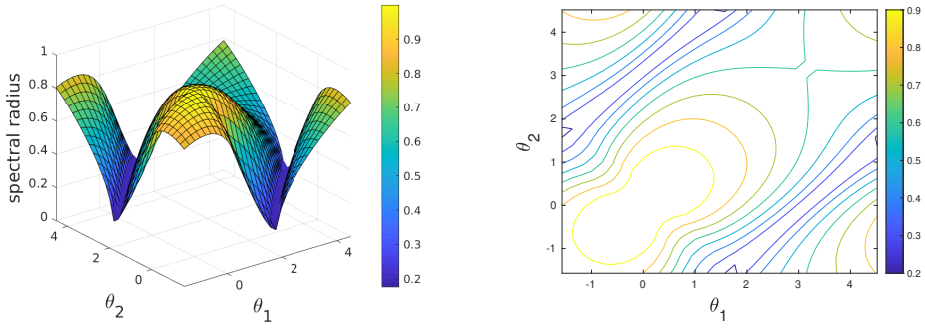


Figure 4.11: The spectral radius of the smoothing error-propagation symbol for the Triad relaxation based on a weighted Gauss-Seidel updating process with $\omega = 0.8$ and different frequencies $\boldsymbol{\theta}$.

two-grid method. We compare the Triad and Vanka smoother, including parallelization aspects such as a coloring scheme.
We start with the introduction of coarse-grid correction symbols. For systems of PDEs that are discretized on staggered grids, the analysis is challenging due to the different location of unknowns. In that case, LFA of transfer operators cannot be done as in the scalar case. MacLachlan and Oosterlee [51] discussed Fourier representations of grid-transfer operators for general staggered meshes in the context of systems of PDEs. Here, we have three types of grid points on the fine and coarse grid. The restriction operator can be decomposed based on the partitioning of dofs associated with the location of the unknowns on the grid. Osterlee and MacLachlan introduced, explained and proved the need to modify LFA due to this staggering. We emphasize this statement in Remark 4.4. More details can be found in [37, 51].

Remark 4.4. When calculating the symbol of a restriction operator that mixes different types of dofs, we must split it into the different types of dofs that it restricts from and to. If the restriction operator is defined on a staggered grid, we have $G_H = G_H^1 \cup G_H^2 \cup G_H^3$ (c.f. (4.18)) and the symbols depend on the location $\boldsymbol{x}$ of the unknowns.

In the next paragraph, we show how to determine the symbol of a restriction operator defined on a staggered mesh. A similar structure for interpolation is established subsequently. Let $\varphi_h(\boldsymbol{\theta^\xi}, \boldsymbol{x}) = e^{\imath \boldsymbol{\theta^\xi} \boldsymbol{x}/\mathbf{h}}$. We have the following equality

$$\varphi_h(\boldsymbol{\theta^\xi}, \boldsymbol{x}) = e^{\imath \boldsymbol{\xi} \pi \boldsymbol{x}/\mathbf{h}} \varphi_H(2\boldsymbol{\theta^{(0,0)}}, \boldsymbol{x}), \quad \text{for all } \boldsymbol{x} \in G_H.$$

Note that this equality is different to the classical theory and was developed by MacLachlan and Oosterlee in [51]. The following relations are based on this

update of the classical theory. Due to the definition of $G_H = G_H^1 \cup G_H^2 \cup G_H^3$, we have

$$\boldsymbol{x} = \begin{cases} \left(2jh, \quad 2(l+\frac{1}{2})h\right)^T, & \text{for } j,l \in \mathbb{Z}, \text{ if } \boldsymbol{x} \in G_H^1, \\ \left(2(j+\frac{1}{2})h, \quad 2lh\right)^T, & \text{for } j,l \in \mathbb{Z}, \text{ if } \boldsymbol{x} \in G_H^2, \\ \left(2(j+\frac{1}{2})h, \quad 2(l+\frac{1}{2})h\right)^T, & \text{for } j,l \in \mathbb{Z}, \text{ if } \boldsymbol{x} \in G_H^3. \end{cases}$$

Note that the following relations hold,

$$e^{\imath \boldsymbol{\xi} \pi \boldsymbol{x}/\mathbf{h}} = \begin{cases} e^{\imath\xi_1\pi 2j} e^{\imath\xi_2\pi 2l} e^{\imath\xi_2\pi} = e^{\imath\pi\xi_2} = (-1)^{\xi_2}, & \text{for } j,l \in \mathbb{Z}, \text{ if } \boldsymbol{x} \in G_H^1, \\ e^{\imath\xi_1\pi 2j} e^{\imath\xi_1\pi} e^{\imath\xi_2\pi 2l} = e^{\imath\pi\xi_1} = (-1)^{\xi_1}, & \text{for } j,l \in \mathbb{Z}, \text{ if } \boldsymbol{x} \in G_H^2, \\ e^{\imath\pi\xi_1} e^{\imath\pi\xi_2} = (-1)^{\xi_1}(-1)^{\xi_2}, & \text{for } j,l \in \mathbb{Z}, \text{ if } \boldsymbol{x} \in G_H^3. \end{cases}$$

Then, the Fourier representation of R_h^H is given by the (3×12) matrix

$$\hat{R}_h^H(\boldsymbol{\theta}) = \left(\tilde{R}_h^H(\boldsymbol{\theta}^{(0,0)}) \quad \tilde{R}_h^H(\boldsymbol{\theta}^{(1,1)}) \quad \tilde{R}_h^H(\boldsymbol{\theta}^{(1,0)}) \quad \tilde{R}_h^H(\boldsymbol{\theta}^{(0,1)})\right),$$

where $\tilde{R}_h^H(\boldsymbol{\theta}^{\boldsymbol{\xi}})$ is given in Definition 4.5 (c.f. Definition 3.21). Note that the matrices $\tilde{R}_h^H(\boldsymbol{\theta}^{\boldsymbol{\xi}})$ are diagonal matrices since the restriction operators of different type of unknowns are not coupled,

$$\tilde{R}_h^H(\boldsymbol{\theta}^{\boldsymbol{\xi}}) = \begin{pmatrix} \tilde{R}_h^H(\boldsymbol{\theta}^{\boldsymbol{\xi}})\big|_u & & \\ & \tilde{R}_h^H(\boldsymbol{\theta}^{\boldsymbol{\xi}})\big|_v & \\ & & \tilde{R}_h^H(\boldsymbol{\theta}^{\boldsymbol{\xi}})\big|_p \end{pmatrix}.$$

Definition 4.5. We call $\tilde{R}_h^H(\boldsymbol{\theta}^{\boldsymbol{\xi}}) := \sum_{\boldsymbol{\kappa} \in \mathbf{V}} \mathrm{r}_{\boldsymbol{\kappa}} e^{\imath \boldsymbol{\theta}^{\boldsymbol{\xi}} \cdot \boldsymbol{\kappa}} e^{\imath \boldsymbol{\xi} \pi \boldsymbol{x}/\mathbf{h}}$ the symbol of R_h^H.

Based on the definition of the restriction operators in Definition 4.1, we are able to compute the entries of $\hat{R}_h^H(\boldsymbol{\theta})$. We show the computation of the symbol for the restriction operator $R_h^H\big|_u$, i.e., the first row of the matrix $\hat{R}_h^H(\boldsymbol{\theta})$.

$$\begin{aligned}
\tilde{R}_h^H(\boldsymbol{\theta}^{(0,0)})\big|_u &= \frac{1}{8}e^{-\imath\theta_1}e^{-\frac{1}{2}\imath\theta_2} + \frac{1}{4}e^{-\frac{1}{2}\imath\theta_2} + \frac{1}{8}e^{\imath\theta_1}e^{-\frac{1}{2}\imath\theta_2} + \frac{1}{8}e^{-\imath\theta_1}e^{\frac{1}{2}\imath\theta_2} + \frac{1}{4}e^{\frac{1}{2}\imath\theta_2} + \frac{1}{8}e^{\imath\theta_1}e^{\frac{1}{2}\imath\theta_2}, \\
\tilde{R}_h^H(\boldsymbol{\theta}^{(1,1)})\big|_u &= -\frac{1}{8}e^{-\imath\theta_1}e^{-\frac{1}{2}\imath\theta_2} - \frac{1}{4}e^{-\frac{1}{2}\imath\theta_2} - \frac{1}{8}e^{\imath\theta_1}e^{-\frac{1}{2}\imath\theta_2} - \frac{1}{8}e^{-\imath\theta_1}e^{\frac{1}{2}\imath\theta_2} - \frac{1}{4}e^{\frac{1}{2}\imath\theta_2} - \frac{1}{8}e^{\imath\theta_1}e^{\frac{1}{2}\imath\theta_2}, \\
\tilde{R}_h^H(\boldsymbol{\theta}^{(1,0)})\big|_u &= \frac{1}{8}e^{-\imath\theta_1}e^{-\frac{1}{2}\imath\theta_2} + \frac{1}{4}e^{-\frac{1}{2}\imath\theta_2} + \frac{1}{8}e^{\imath\theta_1}e^{-\frac{1}{2}\imath\theta_2} + \frac{1}{8}e^{-\imath\theta_1}e^{\frac{1}{2}\imath\theta_2} + \frac{1}{4}e^{\frac{1}{2}\imath\theta_2} + \frac{1}{8}e^{\imath\theta_1}e^{\frac{1}{2}\imath\theta_2}, \\
\tilde{R}_h^H(\boldsymbol{\theta}^{(0,1)})\big|_u &= -\frac{1}{8}e^{-\imath\theta_1}e^{-\frac{1}{2}\imath\theta_2} - \frac{1}{4}e^{-\frac{1}{2}\imath\theta_2} - \frac{1}{8}e^{\imath\theta_1}e^{-\frac{1}{2}\imath\theta_2} - \frac{1}{8}e^{-\imath\theta_1}e^{\frac{1}{2}\imath\theta_2} - \frac{1}{4}e^{\frac{1}{2}\imath\theta_2} - \frac{1}{8}e^{\imath\theta_1}e^{\frac{1}{2}\imath\theta_2}.
\end{aligned}$$

The calculations for the restriction operators $R_h^H\big|_v$ and $R_h^H\big|_p$ are similar. In addition, an equivalent calculation (see the work of MacLachlan and Oosterlee [51]) gives the symbols of bilinear interpolation operators $\hat{P}_H^h(\boldsymbol{\theta})$. The Fourier

representation of the coarse-grid operator can be given similarly to the scalar case as defined in Definition 3.23. We analyze the two-grid method based on the smoothing operators as introduced in the previous section. Based on the updated restriction and prolongation operators, we perform the analysis of the two-grid method with Triad and Vanka smoothing procedures, i.e., we compute the two-grid convergence factor as defined in Definition 3.25 with $\nu_1 = \nu_2 = 2$. The asymptotic convergence factor is given by the spectral radius

$$\rho(E_{TG}) := \sup_{\boldsymbol{\theta} \in T^{\text{low}}} \rho\left(\hat{E}_{TG}(\boldsymbol{\theta})\right),$$

where

$$\begin{aligned}\hat{E}_{TG}(\boldsymbol{\theta}) &= \left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_2} \hat{B}_h^H(\boldsymbol{\theta}) \left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_1} \\ &= \left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_2} \left(\hat{I} - \hat{P}_H^h(\boldsymbol{\theta}) \left(\hat{A}_H(2\boldsymbol{\theta})\right)^{-1} \hat{R}_h^H(\boldsymbol{\theta}) \hat{A}_h(\boldsymbol{\theta})\right) \left(\hat{\mathcal{S}}_h(\boldsymbol{\theta})\right)^{\nu_1}.\end{aligned}$$

Two-grid results based on different multigrid components are shown in Tables 4.4 and 4.5.

P_H^h	$(R_h^H)^T$	$(R_h^H)^T$	bil.	bil.	mix	mix
A_H	Gal.	Red.	Gal.	Red.	Gal.	Red.
$\rho(E_{TG})$	0.49	0.13	**0.08**	0.20	0.11	0.15

Table 4.4: Two-grid convergence factors based on the Vanka smoother with $\omega = 0.8$.

P_H^h	$(R_h^H)^T$	$(R_h^H)^T$	bil.	bil.	mix	mix
A_H	Gal.	Red.	Gal.	Red.	Gal.	Red.
$\rho(E_{TG})$	0.50	0.34	**0.34**	0.43	0.34	0.43

Table 4.5: Two-grid convergence factors based on the Triad smoother with $\omega = 0.8$.

On the basis of parameter studies we figured out that the weighting factor $\omega = 0.8$ promises good smoothing behavior. Therefore, we use this factor. We noticed that a two-grid method based on a Triad-type smoother diverges without a weighting factor, i.e. $\omega = 1$. Apart from that, variations of ω result in rather small deviations of the convergence results. In addition, we use two pre- and postsmoothing

steps during the course of this section. Moreover, we distinguish three prolongation operators, namely bilinear interpolation, the transpose of restriction and piecewise linear pressure interpolation mixed with bilinear velocity interpolation (denoted by "bil.", "$(R_h^H)^T$" and "mix" in Tables 4.5 and 4.4). On the coarse grid, we use either the Galerkin operator or a rediscretization. In particular, we compare the results that we obtain by using different relaxation methods, Triad-type and Vanka-type smoothers. The convergence results of the Vanka smoother in Table 4.4 show its excellent smoothing behavior. The most promising Vanka-type smoother is based on a Gauss-Seidel updating process. In addition, the best combination of the two-grid components for the Triad smoother is a Galerkin coarse-grid operator with bilinear interpolation. The same is true for the Triad smoother. The predicted performance of the Triad smoother shows the potential of this relaxation method. The results in Tables 4.3 and 4.5 show good smoothing characteristics and good two-grid convergence. As visualized in Figures 4.12 and 4.13 and highlighted in Tables 4.4 and 4.5, the best combinations of MG components lead to the convergence factor

$$\rho(E_{TG}) = 0.08,$$

for the Vanka smoother, and

$$\rho(E_{TG}) = 0.34,$$

for the Triad smoother.

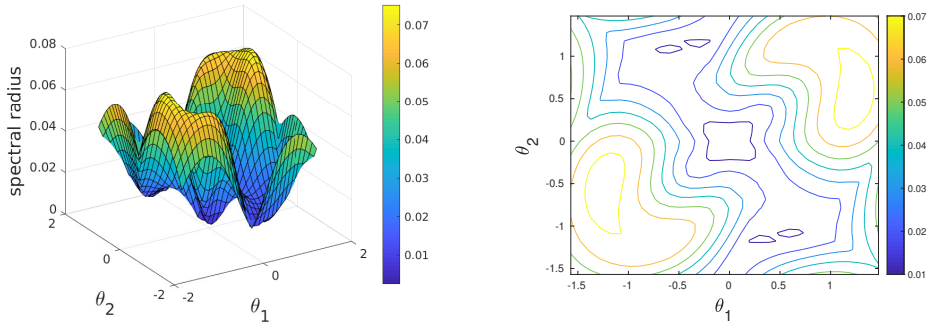


Figure 4.12: The spectral radius of the two-grid symbol for the Vanka relaxation based on a weighted Gauss-Seidel updating process with $\omega = 0.8$ and different frequencies $\boldsymbol{\theta}$.

In addition, we perform an analysis based on a colored smoothing procedure as visualized in Figure 4.7. A red-black coloring scheme for the Triad smoother is similar to the red-black scheme for the Laplace operator, see Lemma 3.27 and Example 3.28. The implementation of a five coloring scheme for the Vanka smoother is more challenging.

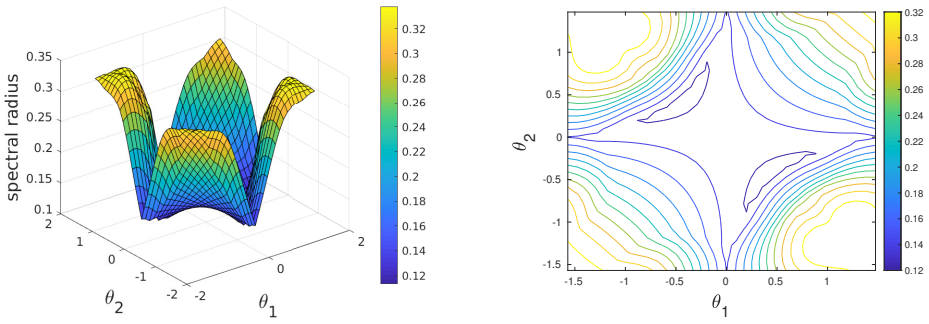


Figure 4.13: The spectral radius of the two-grid symbol for the Triad relaxation based on a weighted Gauss-Seidel updating process with $\omega = 0.8$ and different frequencies $\boldsymbol{\theta}$.

Therefore, we use a tool to perform the analysis of the two-grid operators with colored smoothers. The so-called *aLFA: automated local Fourier analysis* by Kintscher [47] is an open source Python package that is easy to use and adapt. The framework is described in [46]. The main purpose of this framework is to enable the reliable and easy-to-use analysis of complex methods on repetitive structures, i.e., multigrid methods with complex (overlapping) block smoothers. It is easy to use and easy to adapt. Therefore, it reduces the effort required to perform the analysis. This is especially true for the coloring scheme and the two-grid operators. For example, the adjustment of the classical theory to compute the transfer operators, introduced within this section, is not necessary. However, the framework does have some limitations. Each individual operator in the analysis is allowed to change each value of the unknowns at most once. Due to this limitation, a sequential overlapping smoother cannot be analyzed with the approach. This does not limit the application for colored smoothers.
We analyze the two-grid method with the five-color Vanka smoother and the two-color Triad smoother. Again, the methods are based on a Galerkin coarse-grid operator, bilinear interpolation and two pre- and postsmoothing steps, i.e. $\nu_1 = \nu_2 = 2$. The convergence factors are

$$\rho(E_{TG}) = 0.06,$$

for the five-color Vanka smoother, and

$$\rho(E_{TG}) = 0.28,$$

for the two-color Triad smoother. Even though the analysis shows that the Vanka smoother promises better results, we have to keep in mind that the Triad smoother is cheaper, i.e., less arithmetic operations are needed per smoothing step, and easier to parallelize, i.e., a two-coloring scheme is sufficient. However, as introduced

in Remark 4.3, LFA neglects boundary conditions. Therefore, we examine the numerical results of the two-grid method for different boundary conditions to determine their influence. We compare periodic and Dirichlet boundary conditions.

4.5 Numerical results

In this section, we show convergence results of a two-grid and of a V-cycle multigrid method for the Stokes equations. We start with periodic boundary conditions and observe convergence behavior as predicted by LFA in the previous section. That means we employ a two-grid method for a homogeneous problem with periodic boundary conditions based on a 33×33 staggered grid. We apply 20 two-grid cycles with two pre- and postsmoothing steps and start with a random initial guess. We measure the convergence via the 2-norm of the error, i.e. we compute $\|\mathbf{e}^k\|_2 / \|\mathbf{e}^{k-1}\|_2$ after $k = 20$ two-grid cycles. This gives a convergence factor of $\rho = 0.32$ for the weighted Gauss-Seidel-type Triad smoother and $\rho = 0.07$ for the weighted Gauss-Seidel-type Vanka smoother. These numerical results verify the LFA convergence results as given in Section 4.4.4. In addition, Figure 4.14 shows convergence results for the Vanka and the Triad smoother as well as the Braess-Sarazin smoother for a multigrid method.

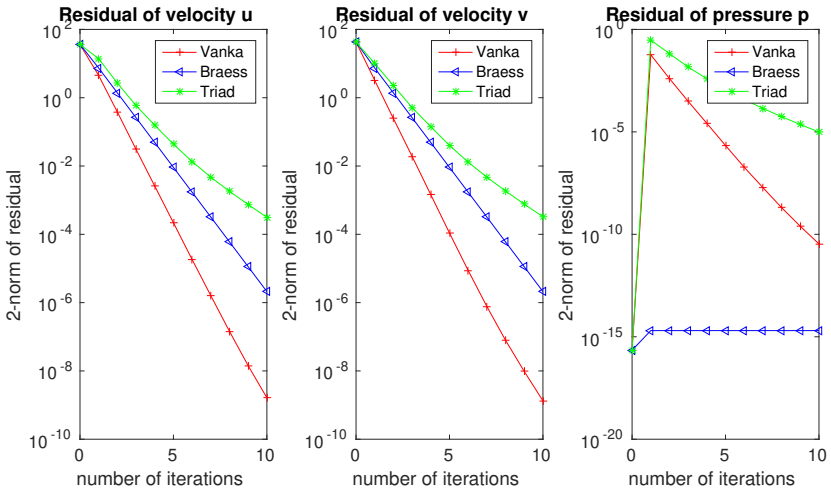


Figure 4.14: Convergence behavior of the multigrid method for the Stokes system with periodic boundary conditions.

We use the Braess-Sarazin smoother here to have an additional reference value. Figure 4.14 shows the two-norm of the residual for different numbers of multigrid V-cycles (iterations) for the Stokes equations. The figure is based on a 33×33 staggered fine grid and a 3×3 coarse grid on the domain $\Omega = (0, 1)^2$ discretized

by the finite difference method. We use two pre- and postsmoothing steps and the zero solution as initial guess. The right-hand side is set to

$$\mathbf{f}_1 = 8\pi^2 \cdot \sin(2\pi \mathrm{x}_1^1) \cdot \sin(2\pi \mathrm{x}_2^1) - 2\pi \cdot \sin(2\pi \mathrm{x}_1^1) \cdot \sin(2\pi \mathrm{x}_2^1),$$
$$\mathbf{f}_2 = 8\pi^2 \cdot \cos(2\pi \mathrm{x}_1^2) \cdot \cos(2\pi \mathrm{x}_2^2) + 2\pi \cdot \cos(2\pi \mathrm{x}_1^2) \cdot \cos(2\pi \mathrm{x}_2^2),$$

where the values x^j correspond to the grid points of the discretized domain $\Omega_h^j = \{\boldsymbol{x}^j := \boldsymbol{k}\mathbf{h} + \delta^j, \boldsymbol{k} \in \mathbb{Z}^2\}$, with $\delta^1 = (0, h/2)$ and $\delta^2 = (h/2, 0)$. Figure 4.14 presents the convergence factors corresponding to the different unknowns u, v, p individually. The convergence factors are obtained by computing the 2-norm of the residual, i.e. we compute $||\mathbf{r}^k||_2$ after each V-cycle. We notice good convergence factors for the three relaxation methods. However, the Vanka smoother converges even faster than the Braess-Sarazin smoother and the Triad smoother. In order to verify the LFA results for the colored smoothing procedures, we again apply 20 two-grid cycles with two pre- and postsmoothing steps and start with a random initial guess. This gives a convergence of $\rho = 0.28$ for the two-color Triad smoother and $\rho = 0.06$ for the five-color Vanka smoother. Based on the lower computational cost of the Triad smoother in combination with the parallelization potential, a natural idea is to increase the number of smoothing steps to improve the convergence behavior. Therefore, we apply 20 two-grid cycles with four pre- and postsmoothing steps for the Triad smoother which leads to the convergence factor $\rho = 0.15$. These additional smoothing steps improve the convergence of the two-grid method while we still have less computational costs compared to the Vanka smoother.
Next, we show convergence results for the Stokes system with Dirichlet boundary conditions. Again, practical two-grid convergence factors for a homogeneous problem discretized on a 33×33 grid are given. We apply 20 two-grid cycles with two pre- and postsmoothing steps and start with a random initial guess. That gives a convergence of $\rho = 0.70$ for the Triad smoother and $\rho = 0.08$ for the Vanka smoother. In addition, we discretize on a 33×33 grid with the domain $\Omega = (0, 1)^2$, use two pre- and postsmoothing steps and employ the zero solution as an initial guess as depicted in Figure 4.15.

The right-hand side is set to

$$\mathbf{f}_1 = 2\pi^2 \cdot \sin(\pi \mathrm{x}_1^1) \cdot \sin(\pi \mathrm{x}_2^1) + \pi \cdot \cos(\pi \mathrm{x}_1^1),$$
$$\mathbf{f}_2 = 2\pi^2 \cdot \cos(\pi \mathrm{x}_1^2) \cdot \cos(\pi \mathrm{x}_2^2) - \pi \cdot \sin(\pi \mathrm{x}_2^2).$$

We choose zero boundary conditions for the x-component u and $\pm\cos(\pi y)$ and $\pm\cos(\pi x)$ for the y-component v of the velocity unknown. Figure 4.15 shows good convergence for the Vanka and Braess-Sarazin smoother (similar to that in the

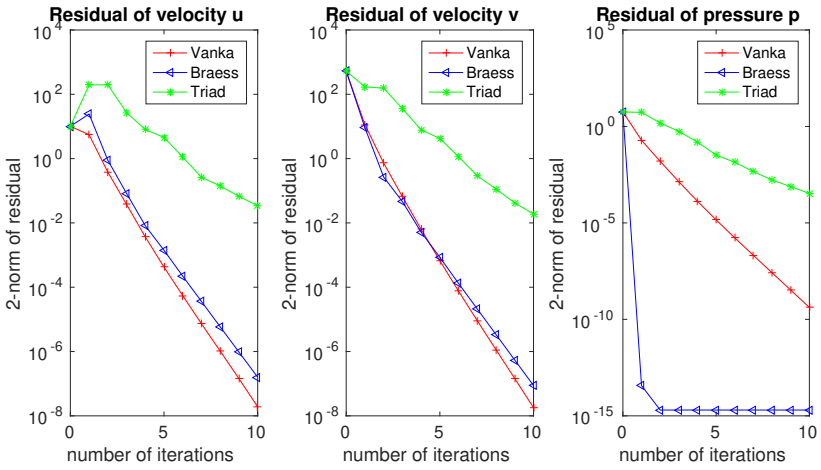


Figure 4.15: Convergence behavior of the multigrid method for the Stokes system with Dirichlet boundary conditions.

periodic case). In contrast, the Triad smoother does not show satisfactory convergence for Dirichlet boundary conditions. An increase in the number of smoothing steps leads to better convergence results. However, we get a convergence factor of $\rho = 0.57$ if we apply six pre- and postsmoothing steps which causes as much computational costs as using the Vanka smoother. The poor convergence behavior of the Triad smoother results from the boundary treatment, as it does not treat the unknowns at the boundary (or near the boundary) in an appropriate way. The unknowns at the boundary are predefined and therefore do not need to be relaxed. The idea of the Triad smoother is to relax three unknowns at the same time, which is not possible with Dirichlet boundary conditions regardless of the choice of the three unknowns. An illustration with an examplary choice of Triad boxes is given in Figure 4.16.

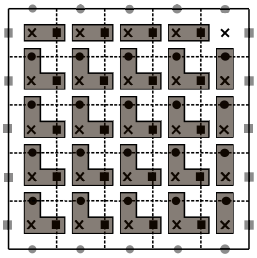


Figure 4.16: Examplary choice of Triad boxes with Dirichlet boundary conditions.

We notice the boxes include three unknowns whenever possible. Close to the boundary we have boxes with two unknowns only. At one corner we have a "box" that consists of one unknown only. Needless to say, different boundary treatment is possible. We tried different boundary treatments, but we could not find one that

fixes the issue. Figure 4.17 illustrates the incorrect treatment by a visualization of the approximate solution after 20 multigrid cycles.

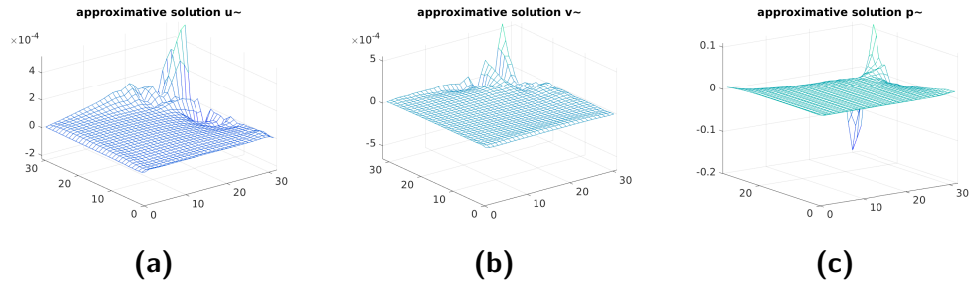


Figure 4.17: Approximate solution of the Stokes equations with Dirichlet boundary conditions after 20 multigrid cycles based on the Triad smoother.

4.6 Algorithm development

The results of the local Fourier analysis, c.f. Section 4.4, and the numerical results for periodic boundary, c.f. Section 4.5, in combination with parellelization properties and computational work show the potential of the Triad relaxation method. However, numerical results for the Stokes system with Dirichlet boundary conditions raise the issue of the Triad method. In what follows, we show how to fix this issue. For this purpose more computational work is needed. The idea is to repeat the relaxation process four times while changing the unknowns contained in one box after each iteration. In addition, we vary the order in which the boxes are updated. This idea is illustated in Figure 4.18.

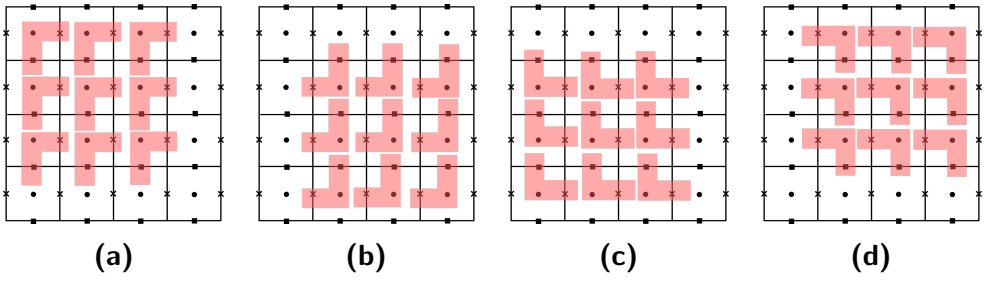


Figure 4.18: New algorithm for the Triad relaxation method. The order of the procedure is performed as visualized in the figures from left to right.

The order of the iteration is visualized from left to right, i.e., Figure 4.18a illustrates the first iteration and Figure 4.18d the last one. The order of relaxing each box in Figures 4.18a and 4.18b is column-wise starting at the top left. The boxes in Figures 4.18c and 4.18d are relaxed in lexicographical order, i.e., row-wise starting

with the bottom left box. This new algorithm improves the convergence results. The two-grid convergence factor of the updated version of the Triad smoother for a homogeneous problem with Dirichlet boundary conditions is $\rho = 0.11$. Again, we applied 20 two-grid cycles with two pre- and postsmoothing steps starting with a random initial guess. Results of a multigrid method are visualized in Figure 4.19.

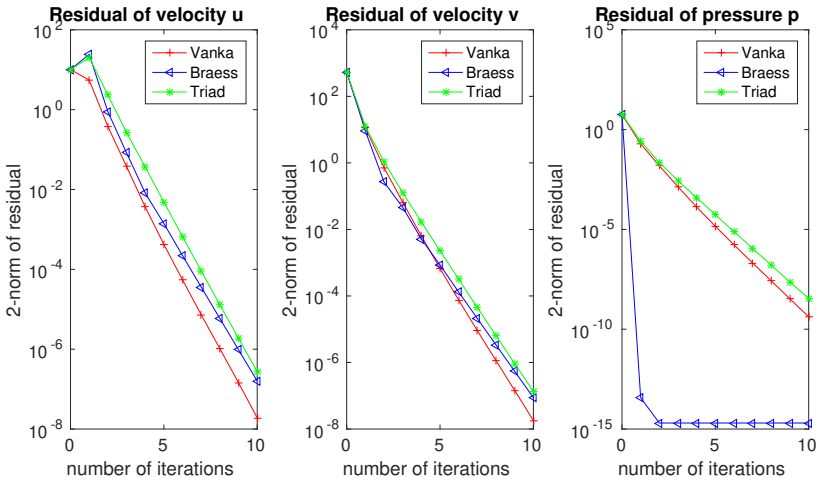


Figure 4.19: Convergence behavior of the multigrid method for the Stokes system with Dirichlet boundary conditions including the new Triad algorithm.

The modified procedure causes four times more computational work. In comparison with the results given in Section 4.3 this leads to a higher number of arithmetic operations for the Triad smoother compared to the Vanka smoother. We have a total number of $139 \cdot (N-1)^2$ for the Vanka smoother versus $176 \cdot (N-1)^2$ operations for the updated Triad smoother. Moreover, the new Triad procedure convergences almost as good as the Vanka smoother. We have $\rho = 0.11$ for the Triad smoother compared to $\rho = 0.08$ for the Vanka smoother. The parallelization properties of the Triad smoother have changed due to the modification. Beforehand, the Triad smoother had the potential for easier and more efficient implementation in parallel compared to the Vanka smoother. Now, one of the four iteration processes has the same parallelization properties as the original Triad smoother. The sequential processing of the four steps limits the parallelization properties.
In conclusion, for the Stokes equations with periodic boundary conditions the two-grid and multigrid convergence of the well-known Vanka smoother is better than the convergence of the Triad smoother. However, the Triad smoother has better parallelization properties and is less expensive than the Vanka smoother. For the Stokes equations with Dirichlet boundary conditions the good convergence of the Vanka smoother remains unchanged, while the convergence of the Triad smoother is not satisfying. However, the modified Triad smoother leads

to convergence results that are almost as good as the convergence results of the Vanka smoother. For a more complete comparison, the algorithms would have to be implemented in parallel, but this is out of the scope of this thesis.

5 Automatic construction of algebraic multigrid smoothers

The preceding chapter shows the efficiency of geometric multigrid methods for saddle point systems. However, the geometric multigrid method is limited due to geometric properties of the underlying grid. To this end, this chapter focuses on algebraic multigrid methods to overcome this limitation and to address general matrix equations. The objective is the development of algorithms that automatically construct advanced smoothing techniques when needed. This is especially important in the algebraic multigrid setting, when no geometric information is given. The classical algebraic multigrid (AMG) method [18, 52] was originally developed as a black box solver to address general matrix equations. This powerful idea is based on the fact that the method focuses on matrix coefficients only. The classical AMG method of Ruge and Stüben [52] works well for a variety of problem classes. However, it is limited, since it is based on properties of M-matrices [71]. To address this, a new class of algorithms was developed based on the multigrid theory, which is denoted by *AMGe* [20, 25, 39, 45]. The classical AMG methods as well as the new class of algorithms assume a simple pointwise smoother and put effort into the construction of a complementary coarse-grid correction to eliminate the so-called algebraically smooth error left over by the relaxation method. Here, we want to address broader classes of problems, for example the second-order definite Maxwell equations as given in Equation (5.1). The difficulty of this equation is characterized by a large near null-space where a pointwise smoother leads to a non-optimal method. To overcome this problem, non-pointwise smoothers are needed, such as the overlapping Schwarz smoother by Arnold, Falk and Winther [5] or the distributive relaxation by Hiptmair [40]. These methods are based on geometric multigrid settings. There are also algebraic multigrid algorithms that are able to overcome the difficulty of Maxwell's equations [11, 41, 42, 48, 62]. These methods, that we denote as *algebraic extended Hiptmair smoother* and *auxiliary space Maxwell solver* (AMS) are based on projections into auxiliary spaces or on auxiliary nodal matrices and they depend on geometric information. This lack of fully algebraic algorithms motivates us to the automatic construction of algebraic multigrid smoothers and complementary coarse-grid correction procedures. The theory developed in [30] provides guidance in constructing these algorithms.

5.1 Model problem - Maxwell's equations

The two-dimensional second-order definite Maxwell problem commonly referred to as Maxwell's equations, serves as our main model problem,

$$\nabla \times \nabla \times \mathbf{z} + \beta \mathbf{z} = \mathbf{f}, \quad \text{in } \Omega, \tag{5.1}$$

where $\beta > 0$ is the spatially varying electrical conductivity, $\mathbf{z}$ is the unknown electric field to be computed and $\mathbf{f}$ is the known right-hand side. The domain, Ω, is an open, bounded and connected Lipschitz domain in $\mathbb{R}^2$, and we impose Dirichlet boundary conditions. We use $\Omega = [0,1]^2$ and $\beta = 0.1$ throughout this thesis. Note that Equation (5.1) involves two two-dimensional curl operators, the two-dimensional curl of a vector-valued function, $w = (w_1, w_2)^T : \Omega \to \mathbb{R}^2$, and the two-dimensional curl of a scalar function $v : \Omega \to \mathbb{R}$, defined as

$$\underline{\text{curl}}\ v := \begin{pmatrix} \partial_{x_2} v \\ -\partial_{x_1} v \end{pmatrix}, \qquad \text{curl}\ w := \partial_{x_1} w_2 - \partial_{x_2} w_1,$$

respectively. The curl operators give rise to the standard Sobolev space,

$$H(\text{curl}, \Omega) := \{v \in L^2\left(\Omega, \mathbb{R}^2\right) \,|\, \text{curl}\, v \in L^2\left(\Omega\right)\},$$

where L^2 denotes the space of square Lesbesgue integrable functions. We therefore discretize our model problem using linear Nédélec's $H(\text{curl}, \Omega)$-conforming finite elements [56] as introduced in Section 2.2.2. The resulting linear system is denoted by

$$A\mathbf{u} = \mathbf{f}, \tag{5.2}$$

with $A = N + Z$, where N is the discrete approximation of the weak form of the curl-curl term in (5.1), and Z is the discrete approximation of the weak form of the β term in (5.1). The discrete unknown is denoted by the vector $\mathbf{u}$ and the discrete right-hand side by the vector $\mathbf{f}$. As introduced in Section 2.2.2, we distinguish discretization into squares or triangles. An exemplary choice of corresponding structured grids is illustrated in Figure 5.1. The linear Nédélec edge dofs are denoted by dots on edges in the figures. The set of dofs is indicated by $V = \{1, \ldots, n\}$. The system matrix A can be represented using the stencil notation on the two grids in Figure 5.1,

$$\frac{1}{h^2} \begin{bmatrix} & -1 & \\ -1 & & 1 \\ & 2 & \\ 1 & & -1 \\ & -1 & \end{bmatrix} + \beta \begin{bmatrix} & \frac{1}{6} & \\ & & \\ & \frac{2}{3} & \\ & & \\ & \frac{1}{6} & \end{bmatrix},$$

for the horizontal edge unknowns and

$$\frac{1}{h^2}\begin{bmatrix} & -1 & & 1 & \\ -1 & & 2 & & -1 \\ & 1 & & -1 & \end{bmatrix} + \beta \begin{bmatrix} \frac{1}{6} & & \frac{2}{3} & & \frac{1}{6} \\ & & & & \end{bmatrix},$$

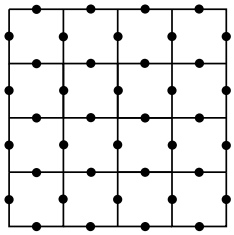
for the vertical unknowns on the quadrilateral grid. In additon, we have the stencils,

$$\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 2 & -1 \\ & -1 & \end{bmatrix} + \beta \begin{bmatrix} & \frac{1}{3} & \\ & & \end{bmatrix},$$

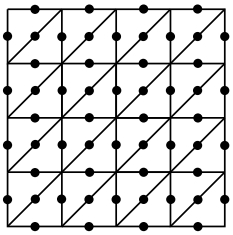
$$\frac{1}{h^2}\begin{bmatrix} & & 1 \\ -1 & 2 & -1 \\ 1 & & \end{bmatrix} + \beta \begin{bmatrix} & & -\frac{1}{6} \\ -\frac{1}{6} & \frac{2}{3} & \\ & & \end{bmatrix},$$

$$\frac{1}{h^2}\begin{bmatrix} & -1 & 1 \\ & 2 & \\ 1 & -1 & \end{bmatrix} + \beta \begin{bmatrix} & & -\frac{1}{6} \\ -\frac{1}{6} & \frac{2}{3} & \\ & & \end{bmatrix},$$

for the unknowns on the triangular grid.



(a) quadrilateral grid



(b) triangular grid

Figure 5.1: Location of unknowns

5.1.1 The difficulty of Maxwell's equations

The difficulty of using AMG methods to solve Maxwell's equations results from the kernel of the curl operator. Using the identity

$$\nabla \times (\nabla \psi) = 0, \tag{5.3}$$

we see that gradients of scalar functions ψ lie within the kernel of the curl operator. Thus, the kernel includes the gradients of all differentiable scalar functions ψ. Since the gradient of a smooth function is smooth and the gradient of an oscillatory function is oscillatory, it is obvious that the kernel contains both smooth and oscillatory functions. An example of a high frequency component in the near

null-space is given in Figure 5.2. For further details, consult [42]. The discrete null-space analogue of (5.3) is

$$NG = \Theta,$$

where the matrix G is a discrete gradient operator and Θ denotes the zero matrix.

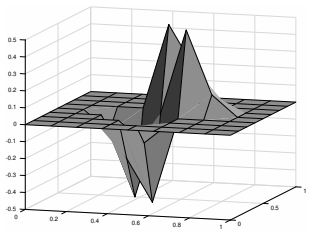


Figure 5.2: High frequency component in the near null-space for the two-dimensional definite Maxwell equation on a quadrilateral grid.

The matrix G is trivial to construct. In particular, each row contains at most two non-zeros (with value ± 1) and corresponds to an edge between two nodes of the associated nodal mesh. To explain this in more detail, we construct a minimal example.

Example 5.1. Let us assume Figure 5.3 denotes one part of a quadrilateral grid with edge dofs $1, \ldots, 7$ as labeled in the figure. Then, the relevant sub-blocks of the curl matrix N and gradient matrix G needed to evaluate the product NG for edge dof 4 are given by

$$N = \begin{pmatrix} 1 & -1 & -1 & 2 & -1 & -1 & 1 \end{pmatrix}, \quad G = \begin{pmatrix} -1 & 1 & & & & & \\ & -1 & 1 & & & & \\ -1 & & & 1 & & & \\ & -1 & & & 1 & & \\ & & -1 & & & 1 & \\ & & & -1 & 1 & & \\ & & & & -1 & 1 & \end{pmatrix}.$$

The columns (and rows) of matrix N correspond to edge dofs $1, \ldots, 7$. The columns of matrix G correspond to nodes in the associated grid and the rows correspond to the edge dofs. It is easy to see that the product NG equals zero.

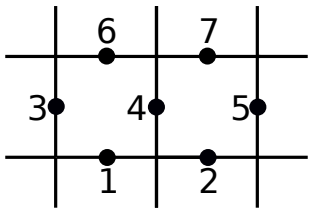


Figure 5.3: Part of a quadrilateral grid with edge dofs $1, \ldots, 7$.

5.1.2 Geometric multigrid methods for Maxwell's equations

Geometric multigrid methods with the overlapping Schwarz smoother by Arnold, Falk and Winther (AFW) [5] or with distributive relaxation proposed by Hiptmair [40] are efficient solvers for Maxwell's equations discretized on quadrilateral grids. We use these relaxation methods to motivate the construction of two different AMG smoothers.
First, we review the idea of the geometric relaxation procedures. The geometric overlapping Schwarz smoother for Maxwell's equations is a box relaxation method. The edge points are clustered into small overlapping boxes $V_{i,j}$ (see Figure 5.4). We solve the systems $A_{i,j}\mathbf{u}_{i,j} = \mathbf{f}_{i,j}$, where (i, j) indicates the index of the current box, $\mathbf{f}_{i,j}$ is some right-hand side, $\mathbf{u}_{i,j}$ is a subvector that contains all unknown edges/degrees of freedom (dofs) of box $V_{i,j}$, and $A_{i,j}$ is a principal submatrix of the Maxwell matrix A that belongs to box $V_{i,j}$. The number of boxes corresponds to the number of nodal points in the grid. The boxes contain all geometric nearest neighbor edges of the nodal points as depicted in Figure 5.4.

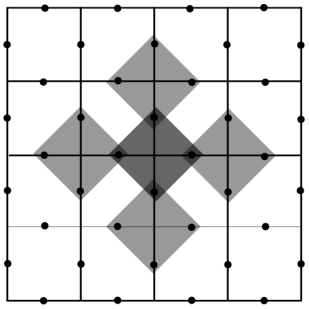


Figure 5.4: Geometric idea of grid separation on quadrilateral grids.

Geometric distributive relaxation for Maxwell's equations uses a gradient matrix G that is constructed such that each row corresponds to an edge between two nodes in the associated nodal mesh, as described in Section 5.1.1.
Equivalently, the matrix G can be constructed such that it uses the same blocks $V_{i,j}$ as the overlapping Schwarz smoother. As described above, each block $V_{i,j}$

defines a submatrix $A_{i,j}$ of the Maxwell matrix A. We can compute the eigenvectors corresponding to the smallest eigenvalue, the so-called smallest eigenvector, of each submatrix $A_{i,j}$. Then, we define the columns of the gradient matrix G with these eigenvectors. That means, each column of the matrix G is the smallest eigenvector corresponding to one of the submatrices $A_{i,j}$. Therefore, the number of columns of the matrix G equals the number of boxes $V_{i,j}$. Neglecting sets at the boundaries, the matrix G ends up being exactly the same as the discrete gradient matrix defined above. In order to illustrate this, we give a minimal example.

Example 5.2. We consider one box $V_{i,j} := \{4, 6, 7, 9\}$ in a given quadrilateral grid, see Figure 5.5. On the one hand, the gradient matrix as described in Example 5.1 is given by

$$G_1 = \begin{bmatrix}
-1 & 1 & & & & & & & & & & \\
 & -1 & 1 & & & & & & & & & \\
-1 & & & 1 & & & & & & & & \\
 & -1 & & & \mathbf{1} & & & & & & & \\
 & & -1 & & & 1 & & & & & & \\
 & & & -1 & \mathbf{1} & & & & & & & \\
 & & & & \mathbf{-1} & 1 & & & & & & \\
 & & & -1 & & & 1 & & & & & \\
 & & & & \mathbf{-1} & & & 1 & & & & \\
 & & & & & -1 & & & 1 & & & \\
 & & & & & & -1 & 1 & & & & \\
 & & & & & & & -1 & 1 & & &
\end{bmatrix},$$

where the number of columns corresponds to the number of nodal points in the associated grid. We highlight the column that solely includes values that correspond to interior edge dofs. On the other hand, the smallest eigenvector of the submatrix $A_{i,j}$ is

$$\begin{pmatrix} 1 & 1 & -1 & -1 \end{pmatrix}^T.$$

We use this eigenvector to build the matrix G_2, s.t.

$$G_2^T = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & -1 & 0 & -1 & 0 & 0 & 0 \end{pmatrix},$$

where the number of columns corresponds to the number of boxes $V_{i,j}$. It is easy to see that $G_2 = G_1$ in the interior of the grid.

Now, we have two different ways to define the gradient matrix G. After the construction of G, the distributive relaxation method applies G as a projector for the Maxwell matrix A defining a new matrix $G^T AG$. Then, a pointwise Gauss-Seidel smoother is applied to this new matrix.

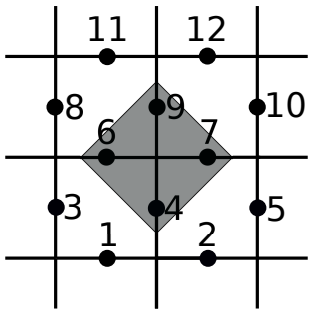


Figure 5.5: Part of a quadrilateral grid with edge dofs $1, \ldots, 12$.

5.2 Automatic construction of AMG smoothers

In AMG, we do not have access to a geometric grid or a finite element mesh. Instead, the graph of the matrix serves as a kind of grid where the graph vertices are the *(grid) points*. The distance between two points is the length of the shortest path in the graph, and the diameter of a set of points is the maximum distance over all pairs of points. The two-grid theory (see Theorem 3.33) implies that small eigenmodes must be handled either by relaxation or coarse-grid correction. For some problems, an optimal method requires using a non-pointwise smoother to damp some of the near null-space components on the fine grid, instead of taking all of the near null-space components to the coarse grid. Since eigenmodes that cannot be handled efficiently by coarse-grid correction have to be eliminated by the smoother, the idea is to devise an automatic algorithm to first construct sets with a diameter smaller than (or equal to) the coarsening factor d that contain near null-space components, then use those sets to build smoothers similar to those in Section 5.1.2. In this thesis, the coarsening factor is chosen to be $d = 2$. Since the geometry of the underlying discretization is not of interest in AMG, for the sake of simplicity, we stick to the quadrilateral discretization of Maxwell's equations to motivate the construction of our algorithms. However, the methods can be applied to any linear system. We start with an automatic construction of sufficient sets.

5.2.1 Automatic construction of sets

A simple strategy to find the corresponding sets is given in Algorithm 5.1. Due to the underlying finite element mesh, we can associate AMG grid points with edges on the mesh. As a result, points associated with edges of the same finite element are all distance-one neighbors of each other (see Figure 5.6).

Algorithm 5.1: Find near null-space sets

$\mathcal{X} = \emptyset$
$\mathcal{K} =$ all sets with diameter $\leq d$
for $J \in \mathcal{K}$ **do**
  for all J that include near null-space components **do**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{J\}$
  end for
end for

A natural first idea for constructing diameter-two sets is to visit each degree of freedom and collect its distance-one neighbors to form $V_{i,j}$. Although this approach does generate diameter-two sets, it does not generate all such sets. And in the Maxwell case, it misses the most important sets to capture.

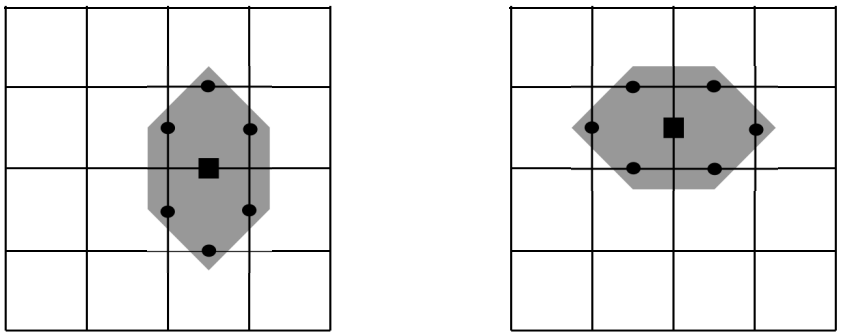


Figure 5.6: Example of distance-one neighbors on quadrilateral grids. The box indicates a given initial grid point, (i, j), and dots indicate distance-one neighbors of (i, j).

Since the distance-one neighbor sets do not account for all diameter-two subdomains, they are not sufficient as box smoothing sets $V_{i,j}$. Accordingly, we construct a method that generates all sets with diameter two that include a near null-space component. The algorithm can be found in Algorithm 5.2 and is explained in the following.
For all degrees of freedom, first the algorithm constructs a set that includes all distance $d = 2$ neighbors (illustrated by crosses in Figure 5.7). Thereafter, a set $V_{k,l}$ is constructed that includes the initial dof k (marked by a rectangle in Figure 5.7), a distance-two neighbor l and all common distance-one neighbors of k and l. Based on these sets, we construct matrices $A_{k,l}$. All matrices are checked if they are nearly singular and we keep only the sets that include a near null-space, i.e. where the corresponding matrix is nearly singular.

Algorithm 5.2: Construct near null-space sets with diameter two

```
Set 𝒳 := ∅, V_point = V
for k = 1 : |V| do
  𝒯 ={all distance-two neighbor dofs of k}
  for l ∈ 𝒯 do
    V_{k,l} = {k, l} ∪ {common distance-one neighbors of k and l}
    A_{k,l} = A(V_{k,l}, V_{k,l})
    if A_{k,l} is nearly singular and V_{k,l} ∉ 𝒳 then
      𝒳 ← 𝒳 ∪ {V_{k,l}}
      V_point ← V_point − V_{k,l}
    end if
  end for
end for
```

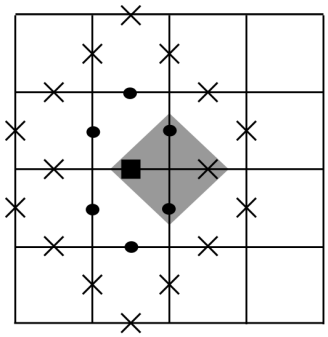


Figure 5.7: Example of an AMG subdomain, $V_{k,l}$, constructed with Alg. 5.2. The box indicates a given initial grid point, (i, j) that corresponds to dof k, crosses indicate distance-two neighbors of (i, j), and dots indicate distance-one neighbors of (i, j).

There are several ways to determine if a submatrix $A_{k,l}$ is nearly singular. For example, if a few steps of local pointwise relaxation on the subsystem result in poor convergence, then $A_{k,l}$ is nearly singular. In this thesis, we compute the smallest eigenvalue λ_0 of $A_{k,l}$ and compare it to a threshold. If the eigenvalue is smaller than the threshold, we keep the corresponding set. The threshold is defined as the maximal entry of $A_{k,l}$ multiplied by a scale factor τ. Based on numerical experiments, we choose $\tau = 0.01$ for Maxwell's equations.
One of the issues with the above algorithm is that it visits sets multiple times. One way to reduce the visits by half is to first construct a matrix where each non-zero entry in row k corresponds to a diameter-two neighbor, then use the lower triangle of this matrix. This does not eliminate all multiple visits, since each set may include more than one distance-two pair. Developing techniques to further optimize the algorithm is future work.
In the special case of quadrilateral elements, the algorithm above generates the

same sets as AFW and Hiptmair in the geometric case. However, this is a general algorithm that can be applied to any linear system and has the potential to generate appropriate smoothers in other settings such as for unstructured grids.

5.2.2 Automatic construction of smoothers

The next step is to build an automatic relaxation method based on the automatically constructed sets development in Section 5.2.1. We construct two algorithms, one is based on the geometric box smoother and the other one is a distributive smoother as introduced in Section 5.1.2.

Overlapping Schwarz smoother for AMG

The idea of the overlapping Schwarz smoother (OSS) is to seperate the degrees of freedom into overlapping subsets $V_{k,l}$ and solve the corresponding systems $A_{k,l}\mathbf{u}_{k,l} = \mathbf{f}_{k,l}$ as described in Section 5.1.2. After smoothing on all of these systems, Gauss-Seidel relaxation is applied to the system defined by the remaining points V_{point}. Section 5.2.1 described the construction of sufficient overlapping sets $V_{k,l}$. The corresponding overlapping Schwarz algorithm is as follows:

Algorithm 5.3: Overlapping Schwarz algorithm

for $m = 1, 2, ...$ **do**
  for $V_{k,l} \in \mathcal{X}$ **do**
    $\mathbf{r}_m \leftarrow \mathbf{f} - A\mathbf{u}_m$
    $\mathbf{u}_m \leftarrow \mathbf{u}_m + I_{V_{k,l}} A_{k,l}^{-1} I_{V_{k,l}}^T \mathbf{r}_m$
  end for
  $\mathbf{r}_m \leftarrow \mathbf{f} - A\mathbf{u}_m$
  $\mathbf{u}_{m+1} = \mathbf{u}_m + I_{V_{\text{point}}} A_{\text{point}}^{\sim 1} I_{V_{\text{point}}}^T \mathbf{r}_m$
end for

Here, $A, A_{k,l}, \mathbf{f}$ and $\mathbf{u}$ are defined as introduced in Section 5.1. The injection matrix with regard to a subdomain V_{point} is indicated by $I_{V_{\text{point}}}$ and V_{point} refers to the set of remaining points, which are not included in one of the near nullspace sets $V_{k,l}$. The term $A_{\text{point}}^{\sim 1}$ denotes the approximate solution of the system $A_{\text{point}}\mathbf{u}_{\text{point}} = I_{V_{\text{point}}}^T \mathbf{r}_m$. Here, we use a Gauss-Seidel relaxation method. In particular, a forward iteration loop for the presmoother and a backward iteration loop for the postsmoother is applied to obtain a symmetric algorithm.

Distributive relaxation smoother for AMG

The distributive relaxation (DR) uses the subsets $V_{k,l}$ to construct a matrix G. We compute the smallest eigenpair $(\lambda_0, \mathbf{v}_0)$ of each subset. If the smallest eigenvalue λ_0 is smaller than a threshold τ, the corresponding eigenvector $\mathbf{v}_0$ defines a column of G. We choose $\tau = 0.01$. If there is more than one small eigenpair, i.e., the smallest eigenvalue is not unique, we randomly select one of the small eigenpairs. For Maxwell's equations and a structured grid the matrix G is equivalent to the discrete gradient matrix used in the geometric Hiptmair smoother described in Section 5.1.2. The Gauss-Seidel relaxation method is used to relax all dofs and the matrix $G^T AG$ is used to relax the local near null-space components. The distributive relaxation algorithm is as follows:

Algorithm 5.4: Distributive Relaxation algorithm

for $V_{k,l} \in \mathcal{X}$ **do**
  Compute the smallest eigenvector $\mathbf{v}_{k,l}$ of $A_{k,l}$
  Add $\mathbf{v}_{k,l}$ to columns of G
end for
for $m = 1, 2, ...$ **do**
  $\mathbf{r}_m \leftarrow \mathbf{f} - A\mathbf{u}_m$
  $\mathbf{u}_m \leftarrow \mathbf{u}_m + A^{\sim 1}\mathbf{r}_m$
  $\mathbf{r}_m \leftarrow \mathbf{f} - A\mathbf{u}_m$
  $\mathbf{u}_{m+1} = \mathbf{u}_m + G(G^T AG)^{\sim 1}G^T\mathbf{r}_m$
end for

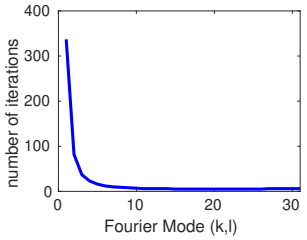
Here, $A, A_{k,l}, G, \mathbf{f}$ and $\mathbf{u}$ are defined in Sections 5.1. The terms $(G^T AG)^{\sim 1}$ and $A^{\sim 1}$ denote the approximate solution by the application of the symmetric Gauss-Seidel relaxation method.

5.3 Evaluating the smoothers

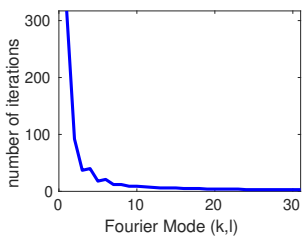
In Section 5.2.2, we introduced two general algorithms for constructing smoothers in AMG. Now, we would like to test them on various problems, but it is challenging in general to evaluate the quality of the resulting smoothers, especially without a corresponding coarse-grid correction. We start with what we call the *Fourier mode study* and highlight positive aspects as well as downsides of this evaluation method. Moreover, we apply two additional ways of evaluating the established smoothers which are based on the so-called *ideal interpolation operator* and the *optimal interpolation operator*.

5.3.1 Fourier mode study

A well-known method to gain some understanding of how smoothers perform is to evaluate how well they damp individual Fourier error components as introduced in Section 3.2.1. To this end, the smoother is applied to the homogeneous linear system $A\mathbf{u} = \mathbf{0}$ with an initial guess given by a Fourier mode with frequency (k, l), i.e. $\tilde{\mathbf{u}} = \sin(k\pi x_1)\sin(l\pi x_2)$, with $l = k = 1, \ldots, n-2$. Good smoothers quickly eliminate highly oscillatory waves which correspond to large values of k and l. Figure 5.8 visualizes the results of the Fourier mode study of the distributive relaxation (DR) and the overlapping Schwarz smoother (OSS). It shows the excellent smoothing property of both relaxation methods for a 33×33 quadrilateral grid.



(a) distributive relaxation (DR)



(b) overlapping Schwarz smoother (OSS)

Figure 5.8: Number of iterations for different Fourier modes $k = l = 1, \ldots, n-2$ with 2112 degrees of freedom (33×33 quadrilateral grid).

However, the Fourier mode study does not work as an evaluation method for unstructured grids since it is based on geometric information. Specifically, the study does not show that the smoother quickly eliminates highly oscillatory waves even if we have a good smoother. In summary, for Maxwell's equations, this method works in structured-grid settings, but does not generalize.

5.3.2 Ideal interpolation operator

Another approach for evaluating the smoothers is to use a general, but impractical, interpolation operator and then compute the A-norm of the resulting two-grid error propagator directly. In the classical AMG setting, a natural choice for this operator is the so-called *ideal interpolation* as introduced in Lemma 3.34. Under the usual assumption that the fine dofs are partitioned and ordered first by F-points and then by C-points, we have

$$A = \begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix}, \quad R^T = \begin{pmatrix} 0 \\ I \end{pmatrix}, \quad S = \begin{pmatrix} I \\ 0 \end{pmatrix},$$

and the ideal interpolation operator P_0 is given by

$$P_0 = (I - S(S^T AS)^{-1} S^T A) R^T = \begin{pmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{pmatrix}.$$

Note that P_0 is the minimizer of the weak approximation property,

$$\|E_{TG}\|_A^2 \leq 1 - \frac{1}{K}, \quad \text{where} \quad K = \sup_{\mathbf{v}} \frac{\|(I-Q)\mathbf{v}\|_{\tilde{M}}^2}{\|\mathbf{v}\|_A^2},$$

c.f. [30] and Theorem 3.33. We compute the two-grid estimate with ideal interpolation, i.e.

$$\begin{aligned} \|E_{TG}\|_A &= \|(I - M^{-T}A)(I - P_0(P_0^T A P_0)^{-1} P_0^T A)(I - M^{-1}A)\|_A \\ &= \|(I - M^{-T}A)(S(S^T AS)^{-1} S^T A)(I - M^{-1}A)\|_A. \end{aligned}$$

The idea is that the two-grid estimate indicates whether or not we use a good smoother M for the problem under consideration. This means small values should correspond to a good smoother and large values of the estimate indicate a poor smoothing behavior. We will see in the following that this is not always the case. In order to partition the fine-grid points into F-points and C-points, we first have to define a coarse grid. In the geometric case, one cell of the coarse grid appears as shown in Figure 5.9a. In the AMG setting, we stick to the case where the coarse-grid points are a subset of the fine points. Here, we choose the algebraic coarse grid close to the geometric one, see Figure 5.9b.

(a) geometric setting **(b)** algebraic setting

Figure 5.9: Coarse-grid cell including fine-grid cells. Boxes indicate coarse-grid points and dots indicate the remaining fine-grid points.

In Table 5.1, we show convergence factors for Maxwell's equations using OSS and DR relaxation methods. We consider three grid sizes: 9×9 grid with 144 dofs, 17×17 grid with 544 dofs and 33×33 grid with 2112 dofs. Table 5.1a displays two-grid convergence factors using the ideal interpolation operator P_0, whereas Table 5.1b presents two-grid convergence factors that are based on a well-known geometric interpolation operator, which is called *Hiptmair interpolation* [40].

#dofs	DR	OSS
144	0.74	0.51
544	0.91	0.83
2112	0.97	0.94

(a) P_0

#dofs	DR	OSS
144	0.11	0.04
544	0.15	0.05
2112	0.15	0.05

(b) geometric interpolation

Table 5.1: Two-grid convergence factors, $\|E_{TG}\|_A^2$, with OSS and DR using P_0 and geometric interpolation.

That means, we use the algebraic coarse grid with Hiptmair interpolation, which is given by the following stencils,

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \mathbf{\frac{1}{2}} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \mathbf{\frac{1}{2}} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{4} & \mathbf{\frac{1}{2}} & \frac{1}{4} \\ & & \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ & & \\ \frac{1}{4} & \mathbf{\frac{1}{2}} & \frac{1}{4} \end{bmatrix}, \tag{5.4}$$

where the bold numbers correspond to the coarse-grid dofs. On the one hand, P_0 produces poor convergence for both smoothers. On the other hand, using Hiptmair geometric interpolation [40] instead of P_0 achieves good h-independent results of 0.15 and 0.05. The comparison of both tables shows that a good interpolation operator exists, but using P_0 does not produce good results. That means the ideal interpolation operator P_0 does not serve as a good interpolation for Maxwell's equations, i.e., it is not the "best" operator.
The reason for this failure of the ideal interpolation operator is not immediately evident. We like to emphasize that the ideal interpolation operator is based on the splitting into C-points and F-points and does not include the influence of the smoother. We refer to the following sections where we identify how the contribution of the smoother may effect the interpolation property of an operator.
The convergence results in Table 5.1 are based on a two-grid method that uses the same coarse grid but different interpolation operators. We give an additional example which highlights that the underlying coarse grid may also influence the interpolation property of the operator P_0. To this end, we consider the elliptic diffusion equation with jumping coefficients, c.f. Definition 2.4, i.e. we have Poisson-like stencils with different scaling factors,

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & -1 & 0 \\ -\epsilon & 2+2\epsilon & -\epsilon \\ 0 & -1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & -\epsilon & 0 \\ -1 & 2+2\epsilon & -1 \\ 0 & -\epsilon & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & -\epsilon & 0 \\ -\epsilon & 4\epsilon & -\epsilon \\ 0 & -\epsilon & 0 \end{bmatrix},$$

where $\epsilon = 1 \cdot 10^{-6}$. The "location" of jumps in the grid is visualized in Figure 5.10. For the sake of simplicity, we call the diffusion equation *Jump problem* hereinafter.

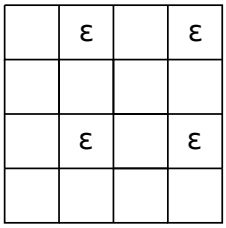


Figure 5.10: Location of jumps for a elliptic diffusion equation with jumping coefficients.

Table 5.2 shows two-grid convergence results for the Jump problem and two choices of coarse grids as visualized in Figure 5.11. We distinguish F-points, denoted by "F" and C-points indicated by "C". One choice for a coarse grid is given in Figure 5.11a. Here, the coarse points are located at the jumps ϵ.

#dofs	DR	OSS
144	0.64	0.19
544	0.88	0.58
2112	0.97	0.86

(a) $\|E_{TG}\|_A^2$ for choice 1

#dofs	DR	OSS
144	0.04	0.03
544	0.05	0.03
2112	0.05	0.03

(b) $\|E_{TG}\|_A^2$ for choice 2

Table 5.2: Two-grid convergence factors, $\|E_{TG}\|_A^2$, with OSS and DR using P_0 and two different coarse grids for the Jump problem.

F	C	F	C
F	F	F	F
F	C	F	C
F	F	F	F

(a) coarse grid choice 1

F	F	F	F
C	F	C	F
F	F	F	F
C	F	C	F

(b) coarse grid choice 2

Figure 5.11: Two different coarse grids for the Jump problem.

The convergence results in Table 5.2 demonstrate that different choices of coarse grids may influence the interpolation property of the ideal interpolation operator. Summarizing, the ideal interpolation operator does not serve as the "best" interpolation operator for all kinds of problems. That stresses the need of an interpolation operator that is complementary to the smoother and does not depend on the choice of the coarse grid.

5.3.3 Optimal interpolation operator

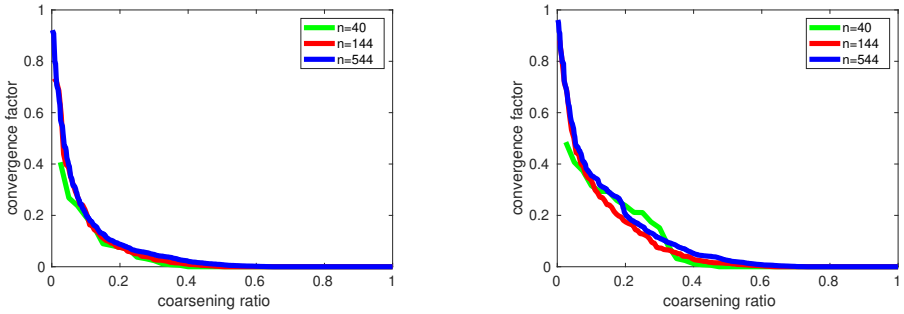
Another interpolation operator we can use is the so-called *optimal interpolation* operator as introduced in Lemma 3.36. This is the minimizer of the sharp version of the approximation property in Theorem 3.35. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and $\mathbf{v}_1, \ldots, \mathbf{v}_n$ denote the eigenvalues and orthonormal eigenvectors of the generalized eigenvalue problem $A\mathbf{v}_i = \lambda_i \tilde{M}\mathbf{v}_i$, then the optimal interpolation operator is given by

$$P_{\#} = (\mathbf{v}_1 \mathbf{v}_2 \ldots \mathbf{v}_{n_c}),$$

where n_c is the number of coarse points. With $P_{\#}$, the convergence factor has the following relationship to the eigenvalues λ_i and the coarse-grid size,

$$\|E_{TG}(P_{\#})\|_A^2 = 1 - \frac{1}{K_{\#}}, \quad K_{\#} = \frac{1}{\lambda_{n_{c+1}}}.$$

For details we refer to Section 3.4. The optimal interpolation operator $P_{\#}$ does not depend on the choice of the coarse-grid points and is based on the smoothing operator M. Therefore, $P_{\#}$ serves as an appropriate measure for the smoothing property of the underlying relaxation method as we will see during the course of this section.
The theoretical two-grid convergence factor $\|E_{TG}(P_{\#})\|_A^2$ can be computed in the following way. The first step is to compute and sort the eigenvalues of the generalized eigenvalue problem. The second step is to choose the number of coarse-grid points, i.e., the coarse-grid size n_c. Then, the two-grid convergence factor can be obtained with the n_{c+1} eigenvalue $\lambda_{n_{c+1}}$. Different choices of the number of coarse-grid points lead to different convergence factors. The number of coarse-grid points n_c in relation to the number of fine-grid points n defines the so-called coarsening ratio. From this, we generate the following figures, which show the convergence behavior using a smoother and a given coarsening ratio n_c/n. The standard factor-2 coarsening ($d = 2$) implies a coarsening ratio of 0.25. As indicated in Figure 5.12, for Maxwell's equations on a quadrilateral grid and factor-2 coarsening we obtain good convergence factors of 0.06 and 0.21 for OSS and DR, respectively. In comparison, results for a standard pointwise Gauss-Seidel relaxation method are visualized in Figure 5.13. We see that the Gauss-Seidel smoother does not serve as a good relaxation method. For Maxwell's equations on a star-shaped mesh, depicted in Figure 5.14 provided by the finite element library MFEM [2], we obtain the results visualized in Figure 5.15. These figures present good convergence factors of 0.08 and 0.18 with regard to full coarsening for OSS and DR, respectively. Some additional interesting results can be obtained for the Jump problem, c.f. Definition 2.4 as introduced in Section 5.3.2. Comparing Figure 5.16a and Figure 5.16b, we notice that we obtain the same results using the pointwise Gauss-Seidel smoother and the distributive relaxation method.



(a) overlapping Schwarz smoother (OSS) **(b)** distributive relaxation (DR)

Figure 5.12: Two-grid convergence factors using $P_{\#}$ with OSS and DR on a quadrilateral grid for Maxwell's equations.

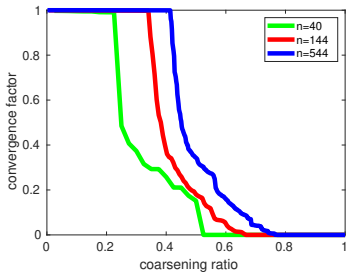


Figure 5.13: Two-grid convergence factors using $P_{\#}$ with Gauss-Seidel (GS) relaxation on a quadrilateral grid for Maxwell's equations.

This is due to the automatic construction of the underlying box sets. As introduced in Section 5.2, our automatic algorithm creates all sets that need additional smoothing. For the Jump problem, no sets are created and the G matrix ends up being the identity matrix. Therefore, the distributive smoother is equivalent to a point-wise Gauss-Seidel relaxation on the original system. That highlights the strength of our automatic approach. It only creates sophisticated smoothers when it is necessary.

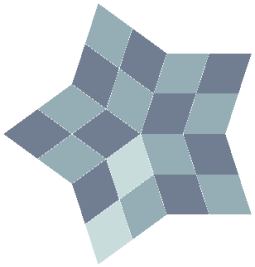
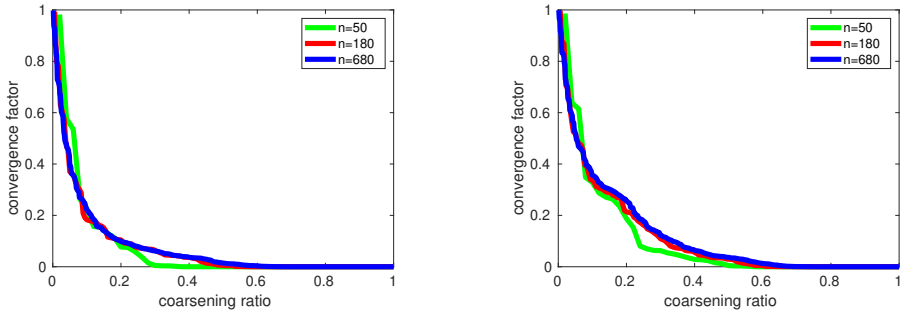
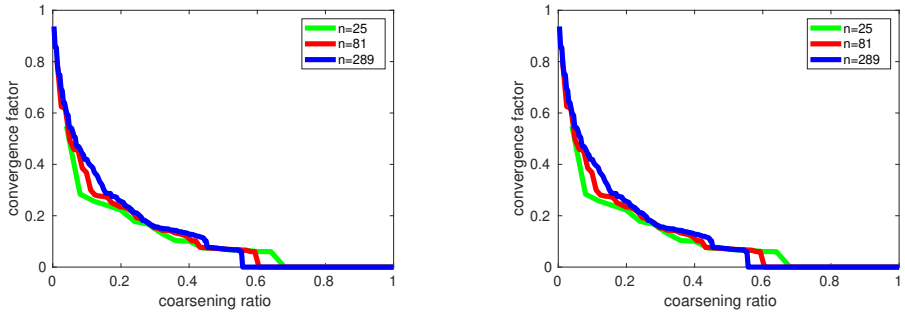


Figure 5.14: Star-shaped mesh.



(a) overlapping Schwarz smoother (OSS) **(b)** distributive relaxation (DR)

Figure 5.15: Two-grid convergence factors using $P_\#$ with OSS and DR on a star-shaped mesh for Maxwell's equations.



(a) Gauss-Seidel (GS) relaxation **(b)** distributive relaxation (DR)

Figure 5.16: Two-grid convergence factors using $P_\#$ with GS and DR for the Jump problem.

5.4 Constructing an interpolation operator

The remaining objective is to find a practical interpolation operator that complements the AMG smoothers in Section 5.2.2. The ideal interpolation operator P_0 in Lemma 3.34 has proven useful for developing practical algorithms. Unfortunately, as we saw in Section 5.3.2, it is not a good interpolation operator for Maxwell's equations even in its original impractical (non-sparse) form. Furthermore, it is unclear how one might use the optimal interpolation operator $P_\#$ to motivate a practical method, since it requires the computation of n_c eigenvectors of the generalized eigenproblem, and $P_\#$ is typically a dense matrix. One idea is to start with the general minimizer of (3.27),

$$P_* = (I - S(S^T AS)^{-1} S^T A) R^T (RR^T)^{-1}, \tag{5.5}$$

and find operators S and R such that the two-grid convergence factor with P_* is closer to the optimal convergence factor using $P_\#$. We also aim for P_* to have more potential in practice than $P_\#$. As a reminder, we think of range(S) as the space on which the smoother must be effective, whereas R defines the coarse variables. Keeping this in mind, we employ the matrix G created by Algorithm 5.4 and define

$$R^T = H \begin{pmatrix} 0 \\ I \end{pmatrix} = H_R, \quad S = G \begin{pmatrix} I \\ 0 \end{pmatrix} = G_S, \tag{5.6}$$

where H is a new matrix, such that $H_R^T G_S = 0$. Here, G is augmented with additional columns such that $G \in \mathbb{R}^{n \times n}$, where n is the number of degrees of freedom. More precisely, we combine the pointwise Gauss-Seidel smoother applied to the original system matrix A ($G^T AG$ with $G = I$) with the pointwise Gauss-Seidel smoother applied to the matrix $G^T AG$. There are different possibilities to augment G with additional columns. Here, we use a Gram-Schmidt process to generate an orthonormal matrix by adding unit vectors and then apply Gram-Schmidt steps [38].
To demonstrate the potential of this idea, we manually construct H_R and G_S for Maxwell's equations. We consider a structured grid case and define the operators based on a regular 2×2 aggregation of fine cells into coarse cells as depicted in Figure 5.17. Consider the case where the coarse cell is in the center of the mesh with neighboring coarse cells on all sides. The boundary case is similar and is not shown here. We choose coarse points by picking one fine dof on each coarse edge as introduced in Section 5.3.2, see Figure 5.9b. Boxes denote the coarse points and dots represent the remaining fine points.

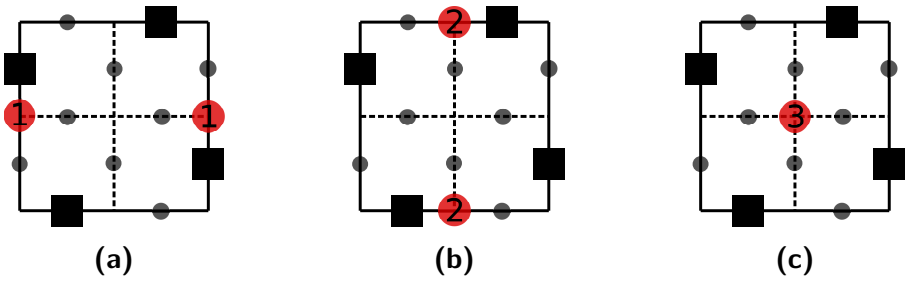


Figure 5.17: Canonical coarse-grid cell where boxes denote coarse points and dots denote the remaining fine points. Numbers correspond to centers of related stencils.

To show that $H_R^T G_S = 0$ and range(H_R) $\bigoplus$ range(G_S) $= \mathbb{R}^n$, we represent each column of G_S and H_R by a stencil on the grid. Note that the center of a stencil representation does not correspond to a dof. The stencils are all centered at nodal locations in the fine mesh and superscripts are used to indicate the node number

(1,2, or 3). The stencils are:

$$G_S^1 = G_S^2 = \begin{bmatrix} & 1 & \\ -1 & & 1 \\ & -1 & \end{bmatrix}, \quad H_R^1 = \begin{bmatrix} & 1 & \\ 0 & & 0 \\ & 1 & \end{bmatrix}, \quad H_R^2 = \begin{bmatrix} & 0 & \\ 1 & & 1 \\ & 0 & \end{bmatrix},$$

$$G_S^3 : \left\{ \begin{bmatrix} & 1 & \\ 0 & & 0 \\ & 0 & \end{bmatrix}, \begin{bmatrix} & 0 & \\ 1 & & 0 \\ & 0 & \end{bmatrix}, \begin{bmatrix} & 0 & \\ 0 & & 1 \\ & 0 & \end{bmatrix}, \begin{bmatrix} & 0 & \\ 0 & & 0 \\ & 1 & \end{bmatrix} \right\}.$$

The stencils G_S^1 and G_S^2 are columns of the original discrete gradient matrix (up to a scaling factor), that is, they are near null-space components. It is clear that $(H_R^1)^T G_S^1 = 0$; for example, as are all other column inner products, hence $H_R^T G_S = 0$.
The matrix H_R is related to the coarse space. This means that for each coarse-grid variable there is a corresponding stencil H_R^i and range(H_R^T)= $\mathbb{R}^{n_c}$. Moreover, G_S is related to the remaining grid points, where G_S^1 and G_S^2 are centered on the boundary of a coarse cell and the interior points are associated with the unit vectors G_S^3. Hence, range$(H_R) \bigoplus$ range$(G_S) = \mathbb{R}^n$.
Using this choice of R and S in (5.5) leads to a modified ideal interpolation operator $\tilde{P}_*$ that produces convergence factors comparable to the optimal case with $P_\#$. For Maxwell's equations, $\tilde{P}_*$ also shows promise as a means of constructing practical interpolation operators. In particular, if we approximate $(S^T AS) = G_S^T A G_S$ in (5.5) by its diagonal, we get an interpolation operator $\hat{P}_*$ that is similar to the natural geometric interpolation [40]. The geometric interpolation operator used by Hiptmair to complement distributive relaxation, is given in (5.4) for an algebraic setting. In contrast, the interpolation operator $\hat{P}_*$ appears in stencil notation for example as,

$$\begin{bmatrix} & 0.0313 & & & & 0.0313 & \\ -0.0313 & & 0.0313 & & -0.0313 & & 0.0313 \\ & 0.2180 & & 0.5 & & 0.2180 & \\ & & & \bullet & & & \\ & 0.2492 & & 0.5 & & 0.2492 & \end{bmatrix},$$

where the red dot, $\bullet$, denotes the center nodal dof as illustrated in Figure 5.17a. Even though, the interpolation operator $\hat{P}_*$ is slightly different from the Hiptmair interpolation operator, we obtain the same two-grid convergence results as given in Table 5.1. Instead, if we modify the Z term of A such that the modified Z is diagonal with coefficient β and approximate the $S^T AS$ term with the diagonal

of $G_S^T A_{\text{modified}} G_S$, then the approximate ideal interpolation operator has exactly the same nonzero pattern as Hiptmair and converges to it as β goes to zero. For $\beta = 0.1$, the stencils appear as,

$$\begin{bmatrix} 0.2495 & & 0.2495 \\ 0.5 & \bullet & 0.5 \\ 0.2495 & & 0.2495 \end{bmatrix}, \quad \begin{bmatrix} 0.2495 & 0.5 & 0.2495 \\ & \bullet & \\ 0.2495 & 0.5 & 0.2495 \end{bmatrix}.$$

In general, if $(S^T AS)$ is well-conditioned, we have the potential to apply a similar approximation strategy to build practical interpolation operators for a broader class of problems and PDEs.

6 Conclusions & Outlook

A key point of an efficient multigrid method is the right choice of the smoothing procedure.

In this thesis we developed multigrid smoothers for saddle point systems, such as the Stokes equations and Maxwell's equations. The main achievements of this thesis are the construction and evaluation of two new algorithms with regard to geometric and algebraic multigrid.

Within the scope of geometric multigrid methods, we developed a new smoothing method, an adaptation of the so-called *Triad smoother*. We were able to demonstrate the strength of the Triad smoother due to its characteristics. In comparison to the well-known and powerful but not easy to parallelize Vanka smoother, the Triad smoother provides good parallelization properties with respect to coloring schemes and communication effort. Although, the adapted Triad smoother is more expensive than the Vanka smoother, we see potential for parallel implementations.
However, it would be beneficial to further analyze, implement and modify the Triad smoother. The investigation of problems with both, Dirichlet and periodic boundary conditions could give interesting insights. In addition, the combination of the Triad smoother, in interior regions of the underlying grid, and the Vanka smoother, close to boundaries, has the potential to create an algorithm with good parallelization properties and good convergence results.

Within the scope of algebraic multigrid methods, we developed a new algorithm that is able to automatically construct smoothing operators. This is especially important in regards to Maxwell's equations where a non-pointwise smoother is not sufficient to eliminate the local high osciallatory modes. Moreover, we evaluated the smoothing algorithm by means of the optimal interpolation operator. Thereby, we demonstrated the challenge to evaluate algebraic multigrid smoothers without an underlying coarse-grid correction. Especially, the surprising failure of the ideal interpolation operator gives new insights in theoretical assumptions. Furthermore, we developed an initial idea to establish a practical interpolation operator that is complementary to the smoothing method.

In future research, it is necessary to prove the failure of the ideal interpolation operator to obtain an understanding of the underlying theory. From there, it would be interesting to generate a fully automatic multigrid method including the construction of a coarse grid and a coarse-grid correction operator.

List of Figures

List of Tables

List of Algorithms & Scripts

List of Notations

Throughout this thesis, scalars are denoted by lower-case letters, vectors are denoted by bold lower-case letter and matrices and constants are denoted by upper-case letters. In addition, the following abbreviations and notations are used across all chapters:

Symbols

$\partial u/\partial x$	the partial derivative of u with respect to variable x
∇	the gradient operator, a multi-variable generalization of the derivative
$\nabla\cdot$	the divergence operator, a differential operator that produces a scalar field
$\nabla\times$	the curl operator
$\Delta = \nabla \cdot \nabla$	the Laplace operator, a differential operator
$\langle a, b\rangle := \int_\Omega ab$	L^2-inner product on continuous functions
$\langle \mathbf{a}, \mathbf{b}\rangle := \sum_i a_i b_i$	Euclidean inner product on vectors
$\|\|\mathbf{a}\|\|_2 := \sqrt{\langle \mathbf{a}, \mathbf{a}\rangle}$	Euclidean norm
$\|\|\mathbf{a}\|\|_A := \sqrt{\langle A\mathbf{a}, \mathbf{a}\rangle}$	A-norm or energy norm

Spaces

$\mathbb{R}^d$	the d-dimensional real space
$\mathbb{Z}$	the set of integers
$\mathbb{Z}_+ = \mathbb{N} = \{1, 2, \dots\}$	the set of positive integers
$\mathbb{N}_0 = \{0, 1, 2, \dots\}$	the set of non-negative integers
$\Omega \subset \mathbb{R}^d$	the bounded and connected domain of a PDE
Ω_h	the discrete domain
$\partial\Omega$	the boundary of the domain Ω
$C^k(\Omega)$	the set of k times continuously differentiable functions on Ω
$C_0^k(\Omega)$	the set of functions $\phi \in C^k(\Omega)$ having compact support in Ω
$C_0^k(\Omega)'$	the dual space of $C_0^k(\Omega)$
$L^p(\Omega), 1 \le p < \infty$	the set of functions ϕ on Ω for which $\|\phi\|^p$ is Lebesgue integrable.
$W^{s,p}(\Omega)$	the fundamental Sobolev spaces
$H^s(\Omega), s \in \mathbb{Z}_+$	the Hilbert spaces
$H(\mathrm{curl}; \Omega)$	$:= \{u \in (L^2(\Omega))^3 \mid \nabla \times u \in (L^2(\Omega))^3\}$

Upper case letters

$\mathcal{A}$	a linear partial differential operator
A	the system matrix or stencil, consists of the coefficients of the variables in a set of linear equations
A_{FF}	the part of the system matrix that belongs to F-points
$\tilde{A}_h(\boldsymbol{\theta})$	Fourier symbol of an operator A respectively A_h
B_h^H	coarse-grid correction operator
C_1, C_2	constants
C-points	set of points that belong to the coarse grid
D	diagonal matrix
E_{TG}	two-grid operator
$\mathcal{F}_h$	space of harmonics
F	a partial differential operator
F-points	set of points that do not belong to the coarse grid
G_h	infinite discrete grid
H	grid width of a coarse grid
I	identity matrix
$K, K_\#, K_*$	specific constants in AMG theory
L	lower triangular matrix
M	smoother iteration matrix
$\mathcal{O}$	Big O notation, refers to the order of a function
P_H^h, P	prolongation operator
$P_*, P_\#$	"best" interpolation operators
R_h^H, R	restriction operator
$\mathcal{S}, \mathcal{S}_h, \mathcal{S}_h(\omega), \mathcal{S}_h(\omega, \boldsymbol{\theta})$	error propagation smoothing operator
S	smoother space (used in AMG theory)
T^{low}	$:= \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$
T^{high}	$:= [-\frac{\pi}{2}, \frac{3\pi}{2})^2 \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$
U	upper triangular matrix
W_l	computational work per MG cycle on level l

Lower case letters

e_i	edge of a finite element for Nédélec edge elements
$\mathbf{e} = (e_1, \ldots, e_n)^T$	vector that corresponds to an error, i.e. $\mathbf{e} = \mathbf{u} - \tilde{\mathbf{u}}$
$\mathrm{f}_i = f(\mathbf{x})$	i-th function in a system of PDE that depends on $\mathbf{x}$
$\mathbf{f} = (f_1, \ldots, f_n)^T$	vector that corresponds to the right-hand side of a discrete system
h	grid width of a fine grid
$\imath$	imaginary unit, i.e. $\imath^2 = -1$
n	number of grid points
$\mathbf{r} = (r_1, \ldots, r_n)^T$	vector that corresponds to the residual, i.e. $\mathbf{r} = \mathbf{f} - A\mathbf{u}$
t_i	tangential unit vector of edge e_i
$u(\mathbf{x})$	function that depends on $\mathbf{x}$
$\mathbf{u} = (u_1, \ldots, u_n)^T$	vector that corresponds to the discrete unknown of a system of equations
$u_i = u(ih)$	function evaluated at gridpoint ih (one dimensional case and notation in context of matrix representations of operators)
$u_{i,j} = u(ih, jh)$	function evaluated at gridpoint (ih, jh) (two dimensional notation in context of stencil representations of operators)
$\tilde{\mathbf{u}}$	approximation to the solution $\mathbf{u}$
$\mathbf{v} = (v_1, \ldots, v_n)^T$	eigenvector of a matrix
$\mathbf{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_d)$	vector of arguments
$x_i = ih$	location of gridpoint with index i
$\mathbf{z} = (z_1, \ldots, z_q)$	vector of functions with components z_l, $l = 1, \ldots, q$, in one dimension, we define $\mathbf{z} = (z_1) = u(\mathbf{x})$
α_i	a degree of freedom (dof)
β	parameter within the formulation of Maxwell's equation
$\delta_{i,j} = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases}$	Dirac measure
$\boldsymbol{\theta} = (\theta_1, \theta_2) \in [-\frac{\pi}{2}, \frac{3\pi}{2})^2$	vector of parameters that characterizes the frequency of the Fourier mode
λ_A	an eigenvalue of A
$\mu_h(\mathcal{S}_h(\omega))$	smoothing factor of $\mathcal{S}_h(\omega)$
ν	number of smoothing steps
$\rho(\mathcal{S})$	convergence factor of the method that is described by the operator $\mathcal{S}$
ϕ	shape function or basis function
$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) := e^{\imath\boldsymbol{\theta}\boldsymbol{x}/\mathbf{h}}$	Fourier mode
$\omega \in \mathbb{R}$	weighting factor

Acronyms

AMG	algebraic multigrid (method)
dof	degree of freedom
FD	finite difference (method)
FE	finite element (method)
FV	finite volume (method)
FW	full weighting (operator)
GS	Gauss-Seidel (method)
GS-LEX	lexicographic Gauss-Seidel (method)
GS-RB	red-black Gauss-Seidel (method)
LFA	local Fourier analysis
MG	geometric multigrid (method)
ODE	ordinary differential equation
PDE	partial differential equation
SOR	successive over-relaxation (method)

Bibliography

[1] *Counting operations in gaussian elimination.* Accessed: 2019-05-12.

[2] *MFEM: Modular finite element methods library.*

[3] *Multigrid solutions to elliptic flow problems*, in Numerical Methods for Partial Differential Equations, S. V. PARTER, ed., Academic Press, 1979, pp. 53 – 147.

[4] I. Anjam and J. Valdman, *Fast MATLAB assembly of FEM matrices in 2D and 3D: Edge elements*, Appl. Math. Comput., 267 (2014).

[5] D. Arnold, R. Falk, and R. Winther, *Multigrid in H(div) and H(curl)*, Numer. Math., 85 (2000), pp. 197–217.

[6] A. Baker, R. D. Falgout, T. Kolev, and U. Yang, *Multigrid smoothers for ultraparallel computing*, SIAM J. Sci. Comput., 33 (2011), pp. 2864–2887.

[7] N. Bakhvalov, *On the convergence of a relaxation method with natural constraints on the elliptic operator*, USSR Comput. Math. Math. Phys., 6 (1966), pp. 101 – 135.

[8] B. Barney, *Introduction to parallel computing*, 2012. Accessed: 2019-06-01.

[9] M. Benzi, G. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[10] A. Bienz and L. N. Olson, *Raptor: parallel algebraic multigrid v0.1*, 2017. Release 0.1.

[11] P. Bochev, C. Garasi, J. Hu, A. C, and S. Tuminaro, *An improved algebraic multigrid method for solving maxwell's equations*, SIAM J. Sci. Comput., 25 (2002).

[12] D. Braess, *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*, Cambridge University Press, Cambridge, UK, 3 ed., 2007.

[13] D. Braess and W. Dahmen, *A cascadic multigrid algorithm for the stokes equations*, Numer. Math., 82 (1999), pp. 179–191.

[14] D. Braess and R. Sarazin, *An efficient smoother for the stokes problem*, Appl. Numer. Math., 23 (1997), pp. 3–19.

[15] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comput., 31 (1977).

[16] A. Brandt, *Rigorous quantitative analysis of multigrid, i. constant coefficients two-level cycle with l_2-norm*, SIAM J. Numer. Anal., 31 (1994), pp. 1695–1730.

[17] A. Brandt and N. Dinar, *Multigrid solutions to elliptic flow problems*, Numer. Methods Partial Differential Equations, (1979), pp. 53–147.

[18] A. Brandt, S. McCormick, and J. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications (Loughborough, 1983), Cambridge University Press, Cambridge, UK, 1985, pp. 257–284.

[19] J. Brannick, F. Cao, K. Kahl, R. Falgout, and X. Hu, *Optimal interpolation and compatible relaxation in classical algebraic multigrid*, SIAM J. Sci. Comput., 40 (2017).

[20] M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, and J. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2001), p. 1570–1592.

[21] H. Brezis, *Functional Analysis, Sobolev Spaces and Partial Differential Equations*, Springer, New York, 2010.

[22] F. Brezzi and M. Fortin, *Mixed and hybrid finite element method*, Springer Ser. Comput. Math., 15 (1991), p. 350.

[23] W. Briggs, V. Henson, and S. McCormick, *A Multigrid Tutorial, 2nd ed.*, SIAM publications, Philadelphia, 2000.

[24] O. Castillo Reyes, J. De la Puente, V. Puzyrev, and J. Cela, *Assessment of edge-based finite element technique for geophysical electromagnetic problems: efficiency, accuracy and reliability*, 04 2015.

[25] T. Chartier, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. Vassilevski, *Spectral AMGe (ρAMGe)*, SIAM J. Sci. Comput., 25 (2003), pp. 1–26.

[26] L. Chen, *Lecture notes in finite difference methods*. Accessed: 2019-06-01.

[27] E. Chow, R. Falgout, J. Hu, R. Tuminaro, and U. M. Yang, *A survey of parallelization techniques for multigrid solvers*, tech. rep., 01 2006.

[28] D. Drzisga, L. John, U. Rüde, B. Wohlmuth, and W. Zulehner, *On the analysis of block smoothers for saddle point problems*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 932–960.

[29] R. Falgout, *An introduction to algebraic multigrid*, Comput. Sci. Eng., 8 (2006), pp. 24–33.

[30] R. Falgout and P. Vassilevski, *On generalizing the AMG framework*, SIAM J. Numer. Anal., 42 (2003), pp. 1669–1693.

[31] R. Falgout, P. Vassilevski, and L. Zikatanov, *On two-grid convergence estimates*, Numer. Linear Algebra Appl., 12 (2005), pp. 471 – 494.

[32] R. Falgout and U. M. Yang, *hypre: A library of high performance preconditioners*, in Computational Science - ICCS 2002 Sloot P.M.A., Hoekstra A.G., Tan C.J.K., Dongarra J.J., vol. 2331, Springer, Berlin, 2002.

[33] R. Fedorenko, *A relaxation method for solving elliptic difference equations*, USSR Comput. Math. Math. Phys., 1 (1962), pp. 1092 – 1096.

[34] W. Hackbusch, *Convergence of multi-grid iterations applied to difference equations*, Math. Comput., 34 (1980), pp. 425–440.

[35] ——, *Multi-Grid Methods and Applications*, vol. 4, Springer, Berlin, 1985.

[36] Y. He and S. MacLachlan, *Local fourier analysis of block-structured multigrid relaxation schemes for the stokes equations*, Numer. Linear Algebra Appl., 25 (2018).

[37] Y. HE and S. MacLachlan, *Two-level fourier analysis of multigrid for higher-order finite-element methods*, 2018.

[38] J. Hefferon, *Linear Algebra*, Mathematics, Saint Michael's College, Colchester, Vermont, 2017.

[39] V. Henson and P. Vassilevski, *Element-free AMGe: General algorithms for computing interpolation weights in AMG*, SIAM J. Sci. Comput., 23 (2001), pp. 629–650.

[40] R. Hiptmair, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1998), pp. 204–225.

[41] R. Hiptmair and J. Xu, *Nodal auxiliary space preconditioning in H(curl) and H(div) spaces*, SIAM J. Numer. Anal., 45 (2007), pp. 2483–2509.

[42] J. Hu, R. Tuminaro, P. Bochev, C. Garasi, and A. Robinson, *Toward an h-independent algebraic multigrid method for Maxwell's equations*, SIAM J. Sci. Comput., 27 (2006), pp. 1669–1688.

[43] V. John and L. Tobiska, *Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible navier–stokes equations*, Int. J. Numer. Methods Fluids, 33 (2000), pp. 453–473.

[44] J. Jones and S. McCormick, *Parallel multigrid methods*, in Parallel Numerical Algorithms, Keyes, Sameh, and Venkatakrishnan, Kluwer Academic, 1997, p. 203–224.

[45] J. Jones and P. Vassilevski, *AMGe based on element agglomeration*, SIAM J. Sci. Comput., 23 (2001), pp. 109–133.

[46] K. Kahl and N. Kintscher, *Automated local fourier analysis (alfa)*, 2018.

[47] N. Kintscher, *python3-package alfa : automated local fourier analysis*.

[48] T. Kolev and P. Vassilevski, *Parallel auxiliary space AMG for H(curl) problems*, J. Comput. Math., 27 (2009), pp. 604–623.

[49] M. Larin and A. Reusken, *A comparative study of efficient iterative solvers for generalized stokes equations*, Numer. Linear Algebra Appl., 15 (2008), pp. 13–34.

[50] C. Lengauer, S. Apel, M. Bolten, A. Größlinger, F. Hannig, H. Köstler, J. Rüde, U. Teich, A. Grebhahn, S. Kronawitter, S. Kuckuk, H. Rittich, and C. Schmitt, *Exastencils: Advanced stencil-code engineering – first project report*, tech. rep., 2014.

[51] S. MacLachlan and C. Oosterlee, *Local fourier analysis for multigrid with overlapping smoothers applied to systems of pdes*, Numer. Linear Algebra Appl., 18 (2011), pp. 751 – 774.

[52] S. McCormick, *Algebraic multigrid (AMG)*, Frontiers Appl. Math., 3 (1987), pp. 73–130.

[53] B. Metsch, *Algebraic Multigrid (AMG) for Saddle Point Systems*, PhD thesis, 01 2013.

[54] M. Mohr and R. Wienands, *Cell-centred multigrid revisited*, Comput. Visual Sci., 7 (2004), pp. 129–140.

[55] P. Monk, *Finite Element Methods for Maxwell's Equations*, Oxford Scholarship Online, Oxford, UK, 2003.

[56] J. Nedelec, *Mixed finite elements in* $\mathbb{R}^3$, Numer. Math., 35 (1980), pp. 315–341.

[57] ——, *A new family of mixed finite elements in* $\mathbb{R}^3$, Numer. Math., 50 (1986), pp. 57–81.

[58] A. Niestegge and K. Witsch, *Analysis of a multigrid stokes solver*, Appl. Math. Comput., 35 (1990), pp. 291–303.

[59] C. Oosterlee and F. Gaspar, *Multigrid methods for the stokes system*, Comput. Sci. Eng., 8 (2006), pp. 34–43.

[60] C. Oosterlee and F. Gaspar, *Multigrid relaxation methods for systems of saddle point type*, Appl. Numer. Math., 58 (2008).

[61] S. Patankar and D. Spalding, *A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows*, Int. J. Heat Mass Transf., 15 (1972), pp. 1787–1806.

[62] S. Reitzinger and J. Schoeberl, *An algebraic multigrid method for finite element discretizations with edge elements*, Numer. Linear Algebra Appl., 9 (2002), pp. 223 – 238.

[63] M. Renardy and R. Rogers, *An Introduction to Partial Differential Equations*, vol. 13, Springer, New York, 2 ed., 2004.

[64] H. Rittich, *Extending and Automating Fourier Analysis for Multigrid Methods*, PhD thesis, University of Wuppertal, June 2017.

[65] J. RUGE, *Efficient solution of finite difference and finite element equations by algebraic multigrid (amg)*, tech. rep., 1984.

[66] A. Schneebeli, *An H(curl;ω)-conforming FEM: Nedelec elements of first type*, tech. rep., 2003.

[67] S. Serra-Capizzano and C. Tablino-Possio, *Multigrid methods for multilevel circulant matrices*, SIAM J. Sci. Comput., 26 (2004), pp. 55–85.

[68] S. Sivaloganathan, *The use of local mode analysis in the design and comparison of multigrid methods*, Comput. Phys. Commun., 65 (1991), pp. 246 – 252.

[69] K. Stüben, *Algebraic multigrid (amg): experiences and comparisons*, Appl. Math. Comput., 13 (1983), pp. 419–451.

[70] L. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM publications, 1997.

[71] U. Trottenberg, C. Oosterlee, and A. Schuller, *Multigrid*, Academic Press, 2001.

[72] A. Tveito and R. Winther, *Introduction to Partial Differential Equations*, vol. 29, Springer, Berlin, 2005.

[73] S. Vanka, *Block-implicit multigrid solution of navier-stokes equations in primitive variables*, J. Comput. Phys., 65 (1986), pp. 138–158.

[74] M. Wang and L. Chen, *Multigrid methods for the stokes equations using distributive gauss-seidel relaxations based on the least squares commutator*, J. Sci. Comput., 56 (2013), pp. 409–431.

[75] R. Wienands and W. Joppich, *Practical Fourier Analysis for Multigrid Methods*, Numerical Insights, CRC Press, 2004.

[76] G. Wittum, *Multi-grid methods for stokes and navier-stokes equations*, Numer. Math., 54 (1989), pp. 543–563.

[77] J. Xu and L. Zikatanov, *Algebraic multigrid methods*, Acta Numer., 26 (2017), pp. 591–721.

[78] X. Zhu and L. Zhang, *Smoothing analysis of distributive red-black jacobi relaxation for solving 2d stokes flow by multigrid method*, Math. Probl. Eng., 2015 (2015), pp. 1–7.

[79] W. Zulehner, *A class of smoothers for saddle point problems*, Computing, 65 (2000), pp. 227–246.