Efficient Multigrid Methods

based on

Improved Coarse Grid Correction Techniques

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. J.T. Fokkema,
voorzitter van het College van Promoties,
in het openbaar te verdedigen op dinsdag 1 september 2009 om 12.30 uur

door

Hisham BIN ZUBAIR,
Master of Science (M.Sc.) Applied Mathematics, University of Karachi, Pakistan
Master of Computer Science (M.C.S.), University of Karachi, Pakistan

geboren te Karachi, Pakistan.

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. C. W. Oosterlee

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. C. W. Oosterlee	Technische Universiteit Delft, promotor
Prof.dr.ir. C. Vuik	Technische Universiteit Delft, NL
Prof.dr.ir. P. Wesseling	Technische Universiteit Delft, NL
Prof.dr.ir. B. Koren	Universiteit Leiden, NL
Prof.dr.ir. S. Vandewalle	Katholieke Universiteit Leuven, BE
Prof. Dr. W. Vanroose	Universiteit Antwerpen, BE
Prof. Dr. S. P. MacLachlan	Tufts University, Medford, Boston, US

This thesis has been completed in partial fulfillment of the requirements of Delft University of Technology (Delft, The Netherlands) for the award of the Ph.D. degree. The research described in this thesis was supported in parts by three institutions. (1) Delft University of Technology, (2) HEC Pakistan, (3) Textile Institute of Pakistan. I thank them sincerely for their support.

Published and distributed by: Hisham bin Zubair
E-mail: h.binzubair@gmail.com

ISBN # 978-90-9024398-6

Keywords: Multigrid, $d-$multigrid, sparse grids, multidimensional partial differential equations, multigrid preconditioined Krylov methods, grid-stretching, indefinite Helmholtz, adaptive mesh refinement.

Printed in The Netherlands

Summary

Multigrid efficiency often suffers from inadequate coarse grid correction in different prototypic situations. We select a few problems, where coarse grid correction issues arise because of anisotropic coefficients, non-equidistant or non-uniform grid stretching, or inherent indefiniteness in the partial differential equation. Most of the work in this thesis can be classified as an attempt to increase multigrid efficiency by analyzing and developing novel grid coarsening techniques that ensure sufficient coarse grid correction for the multigrid algorithm.

Anisotropy in discrete systems can stem from various continuous and discrete features of the problem and has to have its negative effects countered before a successful multigrid solution can be brought about. We select multidimensional stationary diffusion equation as the first important problem to be treated in this context. The work for dimensions higher than three, is aimed at developing grid coarsening strategies for discretization on rectangular hyper-grids that differ greatly in their dimensions, and thus induce the so-called *grid-aligned* anisotropies in the system. Coarse grids formed through standard coarsening fail to provide sufficient coarse grid correction, and alternative block relaxation techniques are expensive in high dimensions. We also investigate and test coarsening strategies with the aim that their use would allow point based relaxation to stay effective in this non-equidistant multigrid scenario. Through local Fourier analysis we also analyze ω-RB Jacobi, and implement a computer program through which we compute the optimal relaxation parameters. There are three important inferences in this regard. (1) Partial (and grid dependent) coarsening strategies allow the successful use of point relaxation methods for this problem. (2) Quadrupling along a few dimensions is a very attractive partial coarsening choice. (3) Optimal relaxation parameters have a significant enhancement effect on multigrid convergence in high dimensions.

The efficient solution of time-dependent multidimensional equations (discretized with implicit time-integration schemes) is also a challenge. We first use the *sparse grid* technique to reduce the exponential complexity of the discrete problem, and then use the d-dimensional multigrid techniques to solve the sparse grid subproblems. In this situation, i.e., with a multitude of different non-equidistant grids, evaluating and using optimal relaxation parameters on the fly is not an option anymore. As a multigrid solver in high dimensions depends on optimal attributes to quite a large extent, we

employ the method as a preconditioner, instead of a solver. This results in a very robust and efficient multigrid preconditioned Bi-CGSTAB solver.

Another coarsening strategy that we develop in this thesis is aimed at two-dimensional grids that are non-uniformly stretched. We investigated different experimental coarsening strategies. A strategy based on improving individual mesh aspect ratios of grid cells proves successful both theoretically as well as experimentally. It is based on adaptive coarsening so that on each successive coarsening step the grid cells become more square. We can also successfully use point relaxation methods with the proposed technique and get nice multigrid convergence in this case. This bit of work also has consequence for locally refined grids.

Efficient multigrid techniques for the indefinite Helmholtz equation form a separate research theme included in the thesis. We employ the complex shifted operator preconditioning technique for model problems that stem from quantum mechanics applications. These model problems have strongly varying wavenumbers which perturbs the solution. The mesh size requirement in the region of this perturbation is quite demanding. This requirement can be eased by saturating the grid in that area. We find that standard coarsening in this situation works well. This is in contrast to the existing strategies where coarsening is only done in the region of refinement, until the grid is regularized. We discretize the model problems both on equidistant and also on locally refined grids, and the efficiency of the multigrid preconditioner is tested in both these situations. Experiments point to the fact that existing techniques for the indefinite Helmholtz are still not satisfactory and must be enhanced.

Some conclusions and outlook mark the end of the thesis.

Samenvatting

In verschillende prototypische situaties verliest multirooster aan efficiëntie door een niet-aangepaste grof-roostercorrectie. We selecteren enkele problemen waarbij de grofroostercorrectie nader onderzocht dient te worden. Voorbeelden zijn anisotropie in de coefficienten, een niet-equidistant of niet-uniform gerekt rooster of het inherent indefiniet zijn van de partiële differentiaal vergelijking. Het grootste deel van dit proefschrift kan worden gezien als een poging om de efficiëntie van multirooster te vergroten door het analyseren en het ontwikkelen van nieuwe vergrovingsstrategiën die voldoende grofroostercorrectie garanderen.

Verschillende continue en discrete eigenschappen van het probleem kunnen aanleiding geven tot anisotropie in het discrete stelsels en de ontwikkeling van een efficient multirooster algorithme vereist het opvangen van de negatieve effecten van deze anisotropie. We selecteren multidimensionele stationaire diffusievergelijkingen als eerste belangrijke problemen die behandeld dienen te worden in deze context. Het werk in meer dan drie dimensies heeft als doel heeft als doel roostervergrovingsstrategiën te ontwikkelen voor discretisatie op rechthoekige hyper-roosters die sterk varieren in dimensie en bijgevolg anisotropie gealigneerd met het rooster in het stelsel introduceren. Grove roosters gevormd door standaard coarsening geven onvoldoende grofroostercorrectie, en alternatieve blokrelaxatieschemas zijn te duur in hoge dimensies. We onderzoeken en testen vergrovingsstrategiën die toelaten dat puntrelaxatie schemas effectief blijven in dit niet-equidistant multigrid scenario. Aan de hand van locale Fourier analyse analyseren we tevens ω-RB Jacobi, en implementeren we een computer programma dat toelaat de optimale relaxatieparameters te berekenen. Uit dit werk kunnen de volgende drie belangrijke conclusies getrokken worden. (1) Partiële (en roosterafhankelijke) vergrovingsstrategiën laten toe puntrelaxatie-methoden effectief toe te passen voor dit probleem. (2) Het verviervoudigen van de roosterafstand in enkele dimensies is een zeer aantrekkelijke partiële vergrovingsstrategie. (3) Optimale relaxtieparameters leiden tot een significante verbetering van de multirooster-convergentie in hoge dimensies.

Het efficiënt oplossen van tijdsafhankelijke multi-dimensionele vergelijkingen (gediscretiseerd met behulp van impliciete tijdsintegratieschema's) is tevens een uitdaging. We maken eerst gebruik van de *sparse grid* techniek om de exponentiële complexiteit van het probleem te reduceren, en maken vervolgens gebruik van d-dimensionele

multigrid technieken om de sparse grid deelproblemen op te lossen. In deze situatie, namelijk gegeven een veelvoud van verschillende niet-equidistante roosters, is het instantaan berekenen en toepassen van optimale relaxatieparameters geen optie meer. Omdat een multigrid solver in meerdere dimensies in hoge mate afhangt van optimale attributen, maken we gebruik van de methode als een preconditioner in plaats van als een solver. Dit resulteert in een erg robust en efficiënt multirooster gepreconditioneerd Bi-CGSTAB algoritme.

Een ander vergrovingsalgorithme dat we ontwikkelden in deze thesis is bedoeld voor twee-dimensionele niet-uniform gerekte roosters. Een strategie gebaseerd op het verbeteren van de aspect ratio van de individuele rooster cellen blijkt succesvol zowel in theorie als in de praktijk. Het is gebaseerd op adaptieve vergroving zodanig dat na elke successieve vergrovingsstap de cellen meer vierkant worden. In de voorgestelde methode kunnen we tevens gebruik maken van puntrelaxatie-methoden en verkrijgen we in deze situatie mooie multirooster convergentie.

Efficiënte multiroostertechnieken voor de indefiniete Helmholtz vergelijking vormt een afzonderlijke thema in dit proefschrift. We maken gebruik van preconditioneeringstechnieken gebaseerd op een operator met complexe shift voor modelproblemen die voortvloeien uit toepassingen in de quantum mechanica. Deze modelproblemen hebben sterk veranderlijke golfgetallen die de oplossing verstoren. De verstoringen leggen sterke beperkingen op aan de roosterdichtheid in het gebied van de verstoringen. Aan deze beperkingen kan worden tegemoet gekomen door het rooster in dit gebied locaal te verfijnen. We vinden dat standaard vergroving in deze situatie goed werkt. Dit is in contrast met bestaande strategiën waarin vergroving wordt toegepast in de gebieden van verfijning totdat het rooster uniform is. We discretizeren het probleem op zowel uniform als locaal verfijnde roosters en onderzoeken de efficiëntie van multigrid in beide situaties. Experimenten bevestigen dat bestaande technieken voor de indefiniete Helmholtz vergelijking verder dienen te worden verbeterd.

Enkele conclusies en aanbevelingen voor verder onderzoek sluiten dit proefschrift af.

Preface

This thesis is based on the scientific papers that I wrote during my stay at the Delft University of Technology. The matter is unfolded and rearranged to highlight coherence between the papers, and to illuminate their common denominator, i.e., efficient coarse grid correction in computational scenarios involving the multigrid method.

I have tried to keep my thesis largely self-contained. However, it is important to realize that multigrid stands as a growing subject, and therefore, a complete coverage of all developments since its inception by A. Brandt (1977), would involve what is beyond the scope and ambition of this work. *A multigrid Tutorial* by Briggs, Henson, and McCormick is an indispensable treatise for the aspiring beginner. For qualitative details and theory, see *Multi-Grid Methods and Applications* by Hackbusch. A very large coverage of applied problems with quantitative estimates and working details appear in *Multigrid* by Trottenberg, Oosterlee and Schüller et al.

Multigrid methods are most effective for solving the kind of partial differential equations, for which some *discrete ellipticity* can be achieved. Unmoved, by this ironic limitation, scientists and mathematicians continue to explore and exploit multilevel methods and techniques for problems that violate this limitation to some extent. There seems to be wide agreement, that this is the attitude with which any science progresses, and multigrid is no exception. During my stay at Delft, it has been my passion and privilege to work on such problems, either for which no efficient solution was around, or else alternatives were complex and difficult to implement.

From a bird's eye view, I have tried to address problems which result from elliptic differential operators, in a discretization scenario that renders a poor discrete ellipticity. These problems are the diffusion equation on different kinds of anisotropic grids, and particular indefinite Helmholtz equations on locally refined grids. These problems (and their subsequent solution presented in this thesis) have offered many new and interesting insights. They include various ways of grid coarsening, various ways of building point-based relaxation methods, and insights in multigrid acceleration through Krylov techniques. They also point to new avenues of research, such as multigrid analysis through different new techniques, as well as efficient implementation of different multilevel algorithms on parallel architectures.

Special thanks to Kees Oosterlee I wish to express my profound thanks to Kees for providing me the opportunity to work at Delft in the NW group. Both from a logistic and research point of view, Kees has been very supportive, helpful and patient throughout my stay in Delft. I learnt a lot of professionalism from him, of which scientific computing is just a subset. I thank Kees Oosterlee for all his personal and professional help.

ACKNOWLEDGEMENTS First and foremost, I'd like to thank Allah SWT, for picking me out of the teeming millions of my kind in Pakistan, and making my foreign stay and research possible through His obvious and obscure ways. I thank Him for guiding my life so far, and bestowing on me so much of His bounties which I so little deserved.

Next, I thank Mehwish for being a loving wife and friend, and bearing with patience all the toil and tribulation that came with my research. Its a joy and comfort sharing my life with her. In various ways I depended on her to keep me trouble free and focused on my research and for singly taking care of our children, Moatasim, Tayyib and our newborn baby Walied. Thanks are also due to Moatasim and Tayyib in understanding why their dad had to be at work on some weekends, and bearing it with a lot of innocent resolve.

My very capable parents, Ammi and Abbu, did a lot for their kids. My primary education in Karachi's best school, well-being, and their undivided attention in all my problems, cannot be thanked in words. I am also indebted to my grandparents, Dada mian (late) and Dadda, for educating me in various walks of life. It surpasses my articulation skills to thank these four people in a befitting manner. I also wish to thank all close members of my and Mehwish's family who in one way or the other contributed in making my stay in The Netherlands, possible, through their help on the Pakistani side.

In the Netherlands, I enjoyed the company of some Pakistani friends. They include, Haroon, Qaiser, Nadeem Shirazi, Mehfooz, Zubair, Tariq, Laiq, Muhammad Nadeem, Ahmad, Shah Muhammad, Malik Naeem, Soban Umar (with his group of Leiden friends), Ibrahim, Naila, Akram, Tahira, Safina and their families. We had not known each other in Pakistan, but have lived here for the last four years as an extended family. I thank them all.

In the department, I learnt a lot from Prof. Wesseling, Piet Sonneveld, Martin van Gijzen and Kees Vuik. Prof. Wesseling was instrumental in illuminating to me the intricacies of Red-Black analysis with Dirichlet conditions. I enjoyed working with my native and foreign colleagues, which include Yogi, Dwi, Coen, Ariel, Fred, Etelvina, Guus, Jennifer, John, Sander, Sander (van der Pijl), Tijmen, Reijer, Scott, Fang, Pauline, Huang, Bin Chen, Grezelak, Jos, Fons, Duncan, Bowen, Abdul Hannan, Diana, Mechteld, and others. The translation of the summary, i.e., *samenvatting* (in Dutch), is a personal favour by my friend and colleague Domenico. The Dutch translation of the propositions was kindly provided by Kees Oosterlee. I'll take this opportunity to thank all these friends for their help and the cosy working environment they created as a group, working in which was a lot of enjoyment and pleasure.

Thanks are also due to Tang Hong, my Chinese apartment-mate and pal for his wonderful company in the first year of my stay at Delft.

A computing infra-structure is complementary to the discipline of scientific computing. During my stay at Delft, I hardly ever saw any major breakdown of this facility at DIAM, and therefore, I wish to appreciate the excellent service of the system administrators Kees Lemmens and Eef Hartman. CICAT and NUFFIC provided a lot of logistic help in various office matters. I'd like to thank the staff in both of these organizations for their help, in particular Mrs. Loes Minkman.

Finally, I thank the members of the opposing committee for devoting some of their precious time to scrutinize my thesis, and some of them, for their off-shore travel to Delft for the event of public defense of this dissertation. Thank you all very much for your time and interest.

Hisham bin Zubair
July 23, 2009

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter serves as an introduction to the work presented in the thesis. It touches upon the history of multigrid and describes the open problems on rectangular domains that impair multigrid convergence. The increased accessibility of modern discretization methods, such as *Discontinuous Galerkin*, and higher order *Finite Element Methods*, for geometrically crooked domains, has given way to a legacy of underestimating the importance of techniques meant primarily for rectangular domains. The practical side of the story is quite different. Rectangular domains still play a very important role in many varied real world applications, and they allow their geometry to be exploited for computational efficiency. This thesis deals with some open problems of bad multigrid convergence on rectangular domains, and this introductory chapter, gives a picturesque view of the problems treated in this thesis.

Section 1.1 covers the origin of the multigrid idea. It gives a brief sketch of the mathematical evolution of multigrid, acknowledges the efforts of pioneering (and later) mathematicians who worked in this field, and points at some of the works that they produced. Through Section 1.2, we point out the importance of the role that rectangular domains continue to play in numerical models of interesting problems. This section lists some example applications. Section 1.3 briefly describes each of the open problems treated in the thesis.

1.1 Origin: Brief Notes from History

The basic approach of isolating the troublesome aspects of complicated problems into small and easier ones, solving the latter, and then using these solutions in various ways to solve the original complicated problem, has been employed by man since early times. The multigrid method of solving partial differential equations is a mathematical manifestation of the same decomposition approach applied to the numerical solution of partial differential equations; henceforth PDEs. Isolated ideas including relaxation, nested iteration, and total reduction have been known as early as 1930s, 1960s and 1970's respectively. 1970 seems to be the decade when multigrid was for-

malized into an algorithm and its efficiency realized by the pioneer of the field, Achi Brandt [1]. Many mathematicians took up the exciting trail and contributed to the idea. Active research in this area has continued unabated ever since. Listing just a few prominent monographs and papers would include Stüben and Trottenberg [2], Brandt, McCormick, and Ruge [3], Hackbusch [4], Dendy [5], Wesseling [6], Yavneh [7], Trottenberg, Oosterlee and Schüller [8], and Wienands [9]. A thorough history of the genesis of multigrid can be found in the monograph [8] (Section 1.5.5) and the references therein.

The majority of the problems for which multigrid has been developed are elliptic, and therefore for two dimensional elliptic problems on standard equidistant grids, multigrid is well established [8]. The term *multigrid* usually refers to *geometric multigrid*, while *AMG (Algebraic Multigrid)* refers to an algebraic variant [3]. At the expense of setup overhead (which AMG entails), it is customary to employ AMG to handle problems discretized on non-uniform grids.

1.2 Applications Modeled on Rectangular Domains

Application areas modeled on rectangular domains are wide and varied. We present a few of them in this section, which highlights the importance of efficient multigrid methods on rectangular domains.

The pricing of options dependent on multiple (d) underlying assets [10], is often modeled by the d-dimensional Black Scholes equation, which appears later in Chapter 5 with mathematical details. This equation, for specific final and boundary conditions, can be transformed to the standard heat equation in d-dimensions on a rectangular domain, and has to be solved efficiently. Higher dimensionality is often tackled by approximation methods that render a multitude of subproblems on grids that have a lower cell density along different dimensions. This is a fine recipe for bringing in anisotropy which (unaccounted) can badly hamper the multigrid solution of these subproblems.

The set of Maxwell's electromagnetic equations in conjunction with Ohm's law is used as the basic model for describing the propagation of electromagnetic waves in Earth's crust. The application in focus is sub-terrain fossil energy exploration [11]. In that work, Maxwell's equations for conducting media are discretized on stretched Cartesian grids, mapping a very large rectangular domain. This domain models a large and interesting stretch of the sea-bed. The resulting discrete system is solved by multigrid and the performance of the solver in the case of stretched grids, is reported to be unsatisfactory, which indicates coarse grid correction issues.

The Schrödinger equation for scattering applications in quantum mechanics [12] can be transformed in certain cases to a Helmholtz type equation on a large square domain. The wavenumber in this case has spatial dependence and grows steeply at Dirichlet boundaries. This imposes a maxima on the mesh size that must be adhered to, near these boundaries, for reasons of approximation accuracy [13]. Using this maxima throughout the domain entails taxing the CPU needlessly. This can be circumvented by saturating grid cells near Dirichlet boundaries, and using the appropriate (relatively larger) mesh size elsewhere in the domain. It allows reduction of the

fine grid complexity, but also implies inclusion of discrete anisotropies; which must be handled for a successful multigrid solution.

1.3 Motivation: Open Problems

Anisotropic coefficients in the PDE as well as discretization on non-uniform and non-equidistant grids result in linear systems with discrete anisotropies, which elude multigrid solution methods due to coarse grid correction issues. Point-smoothing and standard coarsening based multigrid methods display bad convergence in such a situation, and either of these components have to be altered to prevent deterioration of multigrid convergence factors. The known alternatives include replacing point-smoothing by implicit block smoothing, or standard full coarsening by semi-coarsening [6, 8]. In higher dimensions, implicit block smoothing becomes computationally expensive and therefore novel grid coarsening strategies are highly desirable in order to retain point-smoothing.

High-dimensional PDEs are often handled by the sparse-grid method which results in d-dimensional anisotropic subproblems. Solutions of these anisotropic d-dimensional problems are required in order to solve the discrete PDE. These subproblems are difficult to handle due to the high dimensionality of the PDE, as it imposes constraints on geometric visualization. Moreover, extension of the existing computational techniques (for 2 and 3 dimensions) to abstract d-dimensions is non-trivial. Efficient coarse grid correction for high dimensional elliptic PDEs on non-equidistant grids is one issue that the work in this thesis aims to treat.

The other problem treated here, arises in the context of solving the indefinite Helmholtz equation. In this context multigrid is employed for approximate inversion of the Krylov preconditioner. The preconditioner is composed of the original Helmholtz operator with a complex shift [14]. Accuracy requirements as well as considerations of minimizing boundary-reflections, often requires discretization of the Helmholtz equation on logarithmically stretched grids. In this context, the use of the so-called perfectly matched layers PML [15, 16] results in anisotropy in the multilevel perspective. Anisotropy in this situation is non-uniform and as a result error-smoothing switches direction during multigrid relaxation sweeps. We propose a new grid coarsening strategy to handle logarithmically stretched grids. This technique is called *L-shaped* coarsening, and works by improving the mesh aspect ratio of grid cells. This provides one way of diluting anisotropy related adverse effects on the multigrid method.

Coarse grid correction related issues of multigrid methods, are the main motivation of this thesis. Thus in this thesis, we collect and address scattered problems; we propose new partial coarsening techniques to handle anisotropy in a d-dimensional grid; provide d-dimensional extension of some widely used geometric multigrid components, and show the use of the d-multigrid method in a sparse grid setting. We propose the so-called L-shaped coarsening technique for anisotropy resulting from logarithmic grid stretching for a model diffusion equation, and demonstrate the use of a particular multigrid method on locally refined grids as a Krylov preconditioner for the indefinite Helmholtz equations.

Chapter 2

Multigrid Survey: Components and Analysis

This chapter is intended to provide a brief multigrid introduction through a short survey of the literature. The aim is to acquaint non-numerical scientific readers with the basics of the subject, as well as to provide the experienced reader with a recap of the basic multigrid building blocks. The emphasis is on the multigrid concepts and components that we have used in the rest of the chapters.

Section 2.1 details the basic numerical concept of solving partial differential equations approximately. It also describes in detail the short-comings of the stationary iterative methods in the context of asymptotic convergence. Section 2.2 introduces the 2-grid concept and extends it naturally into multigrid. The error iteration operators are given. This is followed by Section 2.3 in which we describe the popular smoothing methods, transfer and coarse grid operators, and provide suitable pointers for the omitted intricacies. Also included in this section is multigrid $O(N)$ optimality which is one of the most fundamental and important properties of the multigrid algorithm. Different multigrid cycle-types which are commonly used in computations, are discussed next. Finally, Section 2.4.3 describes and explains the numerical issues that form the main drive of the thesis.

2.1 Some Preliminary Concepts

In order to study and appreciate the multigrid idea, we need to build at least two preliminary concepts. These are (1) the concept of a computational grid and discretization, (2) the short-comings of stationary iterative methods, such as the Jacobi and the Gauss-Seidel methods.

2.1.1 The Computational Grid and the Discrete Problem

For a brief survey of multigrid we require a discrete elliptic boundary value problem. In Equation 2.1, L^{Ω} represents a general elliptic operator, in continuous differential form, applicable in the interior of the domain Ω. L^{Γ} represents the operator on the domain boundary $\Gamma = \partial\Omega$. $u(\boldsymbol{x})$ (the principal unknown) is assumed to possess continuous second order derivatives. $f^{\Omega}(\boldsymbol{x})$ is the source function and $f^{\Gamma}(\boldsymbol{x})$ is the right hand side of the boundary equation prescribed on the domain boundary. $\boldsymbol{x}$ is a general d-tuple representing a Cartesian point in $\mathbb{R}^d$. $\Omega = \prod_{i=1}^{d}(a_i, b_i)$ is a rectangular hypercube in $\mathbb{R}^d$.

$$\begin{aligned} L^{\Omega}u(\boldsymbol{x}) &= f^{\Omega}(\boldsymbol{x}), \qquad && \boldsymbol{x} \in \Omega \\ L^{\Gamma}u(\boldsymbol{x}) &= f^{\Gamma}(\boldsymbol{x}), && \boldsymbol{x} \in \partial\Omega \end{aligned} \tag{2.1}$$

The first step in discretizing the problem is to define a discrete domain. This is done by sampling the continuous domain Ω at selected discrete points within it; this we call the *computational grid* (Ω_h). The selection of discrete points, at which (or around which) $u(\boldsymbol{x})$ (the principal unknown) has to be approximated, is invariably done through particular rules, which suit the desired accuracy pattern. E.g., one such rule can be to define, first, the number of divisions along each dimension of the domain (i.e., $N_i, \quad i = 1, 2, \cdots, d$), followed by connecting these markers by straight line-segments. This divides the entire grid into rectangular d-dimensional cells. Two different layout schemes for discrete unknowns can be realized by the position of these unknowns. In the so-called vertex-centered layout the discrete unknowns are placed on the points of intersection of these line segments, and represent nodal approximation of the principal unknown. In the cell-centered scheme the discrete unknowns are placed in the center of the cells formed by the line-segments, and stand for an approximate cell average value of $u(\boldsymbol{x})$. The dimension index $i = 1, 2, \cdots, d$ and the dimension mesh-size $h_i = (b_i - a_i)/N_i$. This defines the vertex centered grid as:

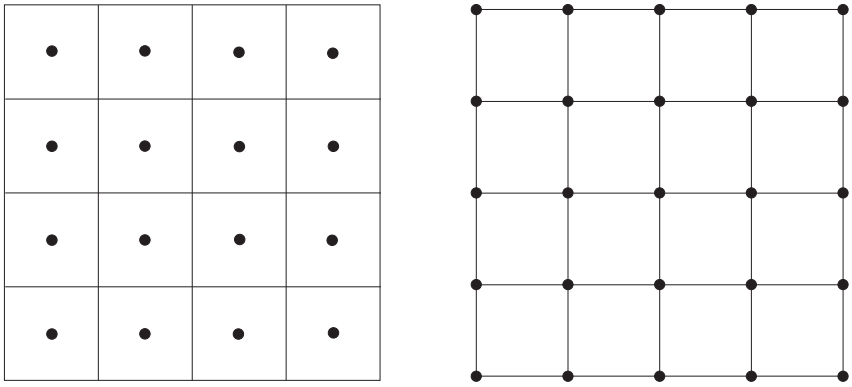
$$\begin{aligned} \Omega_h := \{\boldsymbol{x} = (x_1, x_2, \cdots, x_d); \mid x_i = (x_i)_{j_i} = a_i + j_i h_i\}, \\ j_i = 1, 2, \cdots, (N_i - 1) \\ \Gamma_h := \{\boldsymbol{x} = (x_1, x_2, \cdots, x_d); \mid x_i \in \{a_i, b_i\}\}, \end{aligned} \tag{2.2}$$

and the cell-centered grid as:

$$\begin{aligned} \Omega_h := \{\boldsymbol{x} = (x_1, x_2, \cdots, x_d); \mid x_i = (x_i)_{j_i} = a_i + \frac{h_i}{2} + j_i h_i\}, \\ j_i = 0, 1, \cdots, (N_i - 1) \end{aligned} \tag{2.3}$$

A two-dimensional visualization appears as in Figure 2.1

Grid functions are discrete functions that exist only on the grid points. In this sense, $u_h(\boldsymbol{x})$ which approximates the continuous function u at the grid point $\boldsymbol{x}$, $e_h(\boldsymbol{x}) = u(\boldsymbol{x}) - u_h(\boldsymbol{x})$ (the discrete error), $r_h(\boldsymbol{x})$ (the discrete residual) etc., are all grid functions. Discrete analogs to the continuous differential -and boundary operators, represented by L_h^{Ω} and L_h^{Γ} respectively, are called *discrete operators*, and can be obtained by any suitable discretization technique, e.g. finite differences, finite volume or the finite element method. In this thesis we use the finite difference and the finite volume methods



(a) 2d cell centered grid (b) 2d vertex centered grid

Figure 2.1: 2*d* examples of vertex -and cell-centered grids

for discretization. These techniques relate neighbouring grid unknowns in difference formulae conventionally called *stencils*.

Consequently the discrete boundary value problem is given by:

$$\begin{array}{ll} L_h^{\Omega} u_h(\boldsymbol{x}) = f_h^{\Omega}(\boldsymbol{x}) & \text{(interior)} \\ L_h^{\Gamma} u_h(\boldsymbol{x}) = f_h^{\Gamma}(\boldsymbol{x}) & \text{(boundary)} \end{array} \tag{2.4}$$

In Chapter 4, we show how 1-dimensional finite difference stencils can be used in Kronecker tensor products to form the actual high-dimensional discretization operator matrix. For now it suffices to assert that Equation 2.4 leads to a system of discrete linear equations. This system can usually be written as the matrix equation:

$$A_h u_h = b_h \tag{2.5}$$

The assumption made here, is that truncated version of the interior stencil (or a modified stencil in case of derivative boundary conditions) is used for discrete unknowns neighbouring the domain boundaries. The other possibility is to construct a coupled system of two matrix equations, one representing interior connections and the other, boundary connections. The use of this scheme is more prevalent with higher order boundary treatment. It is worthwhile to note, that a vertex centered scheme is more natural for Dirichlet boundaries. With Neumann and Robin boundary conditions, it is comparatively easier to use a cell-centered layout, where unknowns do not exist on the domain boundaries [6].

In the case of vertex centered discretization, the discrete boundary operator will differ for specific boundary conditions. E.g. for Dirichlet boundary conditions L_h^{Γ} would be equal to the identity operator. For Neumann and Robin boundary conditions, it has to be specified suitably. From the aspect of incorporating the boundary conditions, we use two slightly different schemes for the resulting matrix equation; these are the so-called *eliminated boundary* (used in Chapter 4) and the *non-eliminated boundary* (used in Chapter 5) schemes. In the first scheme, the boundary conditions are explicitly eliminated from the matrix equation by incorporating them in the right hand

side of the system, and therefore boundary unknowns do not appear in the matrix. In the other scheme, boundary conditions are not eliminated and therefore boundary unknowns form part of the matrix equation. However, for either case, $\boldsymbol{A}_h$ is square and assumed to be non-singular and $\boldsymbol{b}_h$ is the right hand side of the system, consisting of the source function sampled at the grid nodes, and the boundary corrections.

Equation 2.5 can be solved in a variety of ways, some of which are classified as *direct* (e.g. Gaussian elimination, Cholesky and LU decompositions etc.), and some as *iterative*. Iterative methods are further classified as *stationary* (e.g. Jacobi, Gauss-Seidel, SOR etc.) and *non-stationary* (e.g. Krylov subspace methods) iterative methods. The two standard and well-known multilevel iterative approaches for solving discrete PDEs are *multigrid* and *AMG*, having their own pros and cons. This thesis is about enhancing a sub-process of multigrid, which is known as *Coarse Grid Correction*, in various troublesome situations.

2.1.2 On the Behaviour of Stationary Iterative Methods

It has been known since the 1930's that stationary iterative methods such as weighted Jacobi, Gauss-Seidel, successive over-relaxation etc., perform well during the first few iterations and then the decay in the residual norm stalls. The reason for this phenomena is the local smoothing trait of these methods, which reduces the amplitude of the highly oscillating components of the error very quickly, but does not interfere much with the less oscillating components. As a result, though global reduction in the error norm is not much, the error gets geometrically smooth.

To observe this phenomenon we analyze the effect of the weighted Jacobi iteration on a 1-dimensional discrete Poisson equation with homogeneous Dirichlet boundary conditions. The domain is restricted to unity $(0, 1)$ and is discretized with N divisions, so that the mesh size is $h = 1/N$. First, we see from Equation B.11 (Appendix B) that the eigenfunctions of the discrete $1d$ $O(h^2)$ FDM Laplacian L_h^{2o} are:

$$\begin{aligned}\varphi_h(\theta, x) &= \sin\left(\frac{\theta}{h}\, x\right);\ \ \theta = lh\,\pi,\ \&\ l \in \{1, 2, \cdots, N-1\} \\ &= \sin\left(\frac{l\, j\, \pi}{N}\right);\ \ x = jh,\ \ j = 0, 1, \cdots, N \end{aligned} \tag{2.6}$$

With each l, for $j = 0, 1, \cdots, N$ these eigenfunctions generate an eigenvector of the discretization matrix A_h.

The Jacobi method is a special case of ω-Jacobi with $\omega = 1$. From Equation A.4 (Appendix A), we have the error iteration operator of the ω-Jacobi method given as:

$$e_h^{i+1} = S_h^{\gamma} e_h^i = |\widetilde{S}_h^{\gamma}| e_h^i \tag{2.7}$$

where γ is the number of times, the ω-Jacobi method is iterated. It is trivial to verify that the eigenvectors of S_h are the same as that of A_h, and the eigenvalues $\widetilde{S}_h$ are given by:

$$\widetilde{S}_h(\omega) = 1 - 2\omega \sin^2\left(\frac{l\,\pi}{2N}\right) \tag{2.8}$$

We illustrate the effect of applying the operator S_h to five different choices of the initial error e_h^0. In this example we use $N = 16$ divisions. The first four choices are given by Equation 2.6, with $l = 1$, $l = 2$, $l = 5$, and $l = 8$ respectively, so that for $j = 0, 1, \cdots, 16$, we have:

$$e_{h,1}^{(0)} = \sin\left(\frac{j\pi}{16}\right); e_{h,2}^{(0)} = \sin\left(\frac{2j\pi}{16}\right); e_{h,3}^{(0)} = \sin\left(\frac{5j\pi}{16}\right); e_{h,4}^{(0)} = \sin\left(\frac{8j\pi}{16}\right);$$

and the fifth one, by a particular linear combination, so that:

$$e_{h,5}^{(0)} = \frac{4}{6}\sin\left(\frac{2j\pi}{16}\right) + \frac{1}{6}\sin\left(\frac{5j\pi}{16}\right) + \frac{1}{6}\sin\left(\frac{10j\pi}{16}\right)$$

Consider Figure 2.2. Each row shows three diagrams. The first representing the initial error, the second -after one ω-Jacobi sweep, and the third -after three ω-Jacobi sweeps. We observe that the Jacobi method works rather well for eigenvectors with $l \geqslant N/2$ and very poorly for eigenvectors with $l \ll N/2$. We thus define low-frequency eigenvectors as those for which $0 \leqslant l < N/2$; and high-frequency eigenvectors, for which $N/2 \leqslant l < N$. From the last row of diagrams (m),(n),and (o), we observe that if the initial error is composed of low and high frequency components (the natural case), then after a few iterations, the low frequency components remain while the high frequency ones get damped.

From the view-point of asymptotic convergence, we observe from Equation 2.7, that the *minimal* error decay depends on the *maximal* eigenvalue (the spectral radius) of the error iteration matrix. If the spectral radius is greater than 1.0, the error will magnify instead of decaying; if it is less than -but close to 1.0, the error will decay very slowly and if it is close to 0.0, the error will decay very fast.

We select a prototype low frequency mode $l = 1$ and a prototype high frequency mode $l = N/2$. Equation 2.8 reveals that the eigenvalues for these modes are:

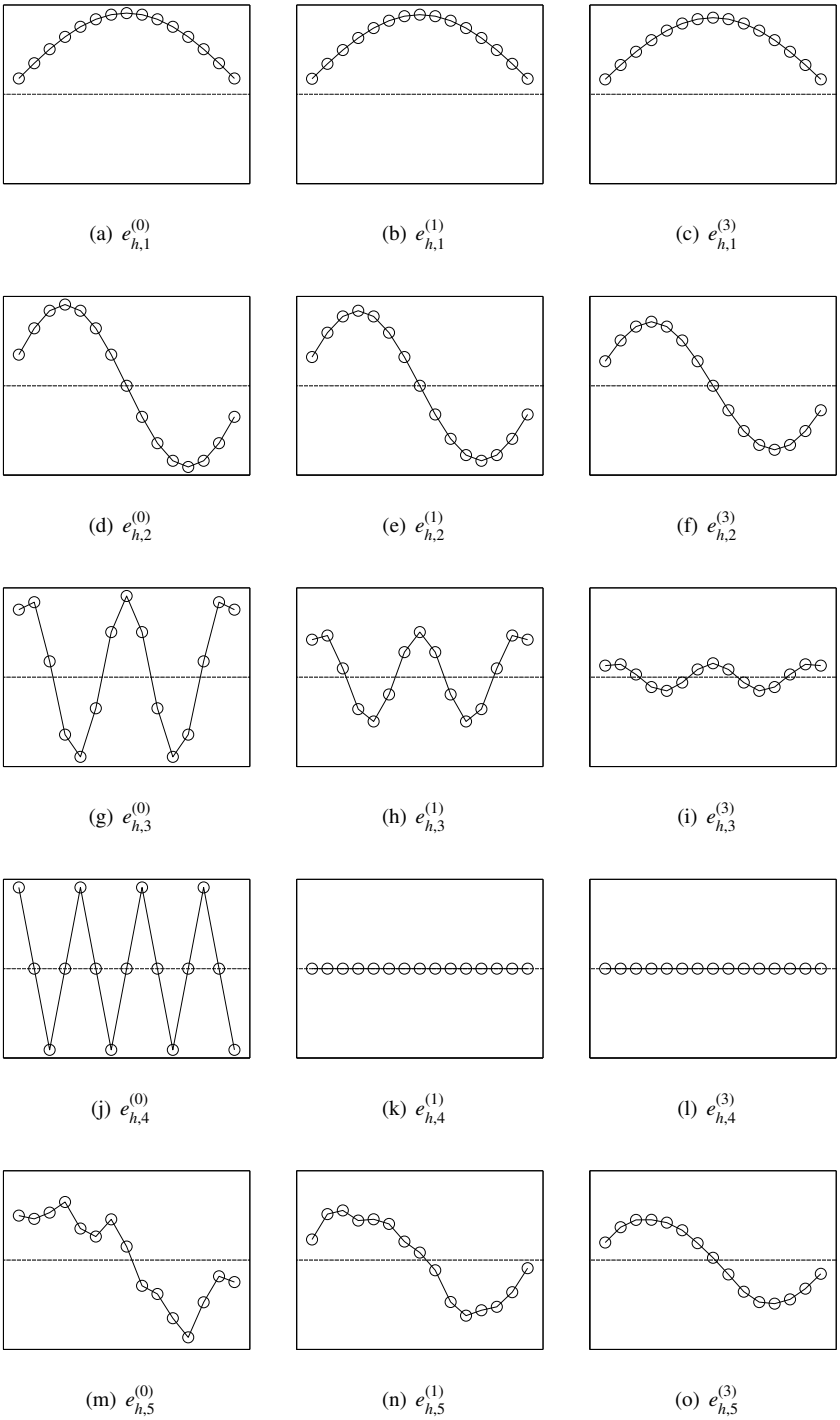
$$\left|\widetilde{S}_h^{(1)}(\omega)\right| = \left|1 - \omega\frac{\pi^2 h^2}{2}\right| \quad \& \quad \left|\widetilde{S}_h^{(N/2)}(\omega)\right| = |1 - \omega| \text{ respectively}$$

These eigenvalues lead to two clear observations:

- For sufficiently small mesh sizes, the use of ω cannot reduce eigenvalues corresponding to low frequency eigenvectors, such as $|\widetilde{S}_h^{(1)}(\omega)|$.
- A large ω, optimized for maximum effect on low frequency modes, e.g. $\omega = 2/(\pi^2 h^2)$, excites the high frequency modes.

These observations suggest that ω should be optimized for maximum reduction of the high frequency modes. With this optimized ω, the largest value of $|\widetilde{S}_h^{(l)}(\omega)|$ (for $N/2 \leqslant l \leqslant (N-1)$) provides an estimate of the smallest reduction in the amplitude of the high frequency modes, and is called the *smoothing factor* of the iterative method.

Definition 2.1.1. (The smoothing factor) *The concept of a smoothing factor is easily generalizable to the case where A_h results from a d dimensional PDE discretized with N equidistant divisions along each dimension. In this case, the wave number is* $\boldsymbol{l}$ =



(a) $e_{h,1}^{(0)}$ (b) $e_{h,1}^{(1)}$ (c) $e_{h,1}^{(3)}$

(d) $e_{h,2}^{(0)}$ (e) $e_{h,2}^{(1)}$ (f) $e_{h,2}^{(3)}$

(g) $e_{h,3}^{(0)}$ (h) $e_{h,3}^{(1)}$ (i) $e_{h,3}^{(3)}$

(j) $e_{h,4}^{(0)}$ (k) $e_{h,4}^{(1)}$ (l) $e_{h,4}^{(3)}$

(m) $e_{h,5}^{(0)}$ (n) $e_{h,5}^{(1)}$ (o) $e_{h,5}^{(3)}$

Figure 2.2: Effect of doing 1 and 3 Jacobi iterations on different choices of the initial error, based on low and high frequency eigenvectors.

$(l_1, l_2, \cdots, l_d)$, *and high frequencies are characterized by* $N/2 \leqslant \max(l_i) \leqslant (N-1)$. *From the perspective of this rigorous analysis, the smoothing factor of a relaxation method is thus defined as:*

$$\mu^{(I)}(h, \omega) = \max \left| \widetilde{S}_h^{(I)}(\omega) \right|, \quad N/2 \leqslant \max(l_i) \leqslant (N-1); \tag{2.9}$$

i is the dimension index, so that $i = 1, 2, \cdots, d$.

Many stationary iterative schemes bear the relaxation behaviour, in that, they damp the oscillatory modes of the error and leave the smooth modes intact. As the smooth modes persist the error decay stalls. Multigrid is basically aimed at supplementing this deficiency of the basic iterative schemes, so that all error modes can be effectively reduced.

2.2 Multigrid

2.2.1 Inception of the 2-grid Idea

Before presenting the actual 2-grid algorithm, it is worth while to check, how the smooth error modes look when projected onto a smaller sized grid. Figure 2.3 shows the 4^{th} Fourier mode for a grid with 8 points, superimposed over the 4^{th} mode for a grid with 16 points. The horizontal axis for the grid with 8 points is scaled to match the grid with 16 points. Evidently, we see that the smooth 4^{th} mode of the *fine grid*, translates into the oscillatory 4^{th} mode of the coarse grid. It is then logical to expect that (for this error mode) the stationary iterative method will work more effectively on the coarse grid than the fine grid. This observation hints at a nested procedure that can be set up as a supplement to the basic iterative method, in order to decrease the amplitude of the low frequency modes effectively.

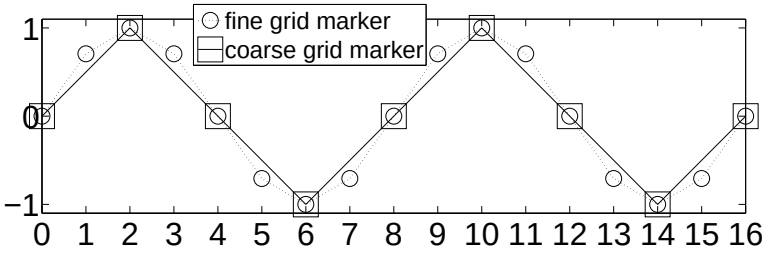


Figure 2.3: The 4^{th} eigenmode on the coarse grid superimposed on the fine grid. This mode is visibly oscillatory on the coarse grid, while it is smooth on the fine grid.

2.2.2 The 2-grid Scheme from an Algorithmic View

We start from a 2-grid perspective. Imagine two grids in the framework, i.e., a fine grid (Ω_h) and a coarse grid (Ω_H), such that the coarse grid nodes form a subset of the fine grid nodes, ($\Omega_H \subset \Omega_h$). In a largely abstract sense, the idea is to run *a*

stationary iterative method (e.g. ω-Jacobi) a *few times* on the discrete problem thereby rendering the error smooth, then *restrict* the residual to the coarser grid . Presume that an invertible *coarse grid operator* exists and is at hand. *Solve* the coarse grid defect equation *exactly* (easier due to fewer unknowns). The solution of the coarse grid equation is actually the coarse representation of the fine grid error. *Interpolate* the missing error components at the fine-grid-only nodes ($\Omega_h \setminus \Omega_h \cap \Omega_H$), and add this correction to the last solution iterate on the fine grid. Finally, run the stationary iterative method *a few times* again. This completes a 2-grid cycle, and the process may be continued until the residual norm is satisfactorily diminished.

Remark 2.2.1. (Postsmoothing) *Due to the phenomenon of aliasing, the high frequency modes on the fine grid (geometrically) coincide, and are indistinguishable from the low frequency modes on the coarse grid. Therefore, relaxation on the coarse grid leaves these modes unaltered. However, the coarse grid correction process involves interpolation. If the prolongation operator is perfect (which means that its range contains the fine grid error exactly), the high frequencies do not need any further damping. In practice this is usually not the case, and interpolation may result in exciting some of the high frequency modes. It is therefore important to damp them before the start of the next cycle. Postsmoothing is performed to bring about this re-damping of the high frequency modes.*

Algorithm 1 A 2-grid cycle

1	$u_h^{i+1/3} = S_h^{\nu_1}\, u_h^i + s_h$	ν_1 presmoothing sweeps
2	$r_h = b_h - A_h u_h^{i+1/3}$	residual computation
3	$r_H = I_h^H\, r_h$	restriction of the residual to Ω_H
4	$e_H = A_H^{-1} r_H$	exact determination of the error on Ω_H
5	$e_h = I_H^h\, e_H$	prolongation of the error to Ω_h
6	$u_h^{i+2/3} = u_h^{i+1/3} + e_h$	correction of the last solution iterate
7	$u_h^{i+1} = S_h^{\nu_2}\, u_h^{i+2/3} + s_h$	ν_2 postsmoothing sweeps

The actual problems to be treated can be greatly diverse, and therefore, this abstract description of the 2-grid idea leaves options open for multigrid components. This includes choices of stationary methods for smoothing purpose, the number of times to smooth before (pre-smooth) and after (post-smooth) coarse grid correction. The available choices of restriction operators to confine the residual to the coarser grid, and considerations that bias the use of one choice against another. Approximate (and yet suitably accurate) representation of the fine grid operator on the coarse grid, and factors that influence available choices. The prolongation operator to use to interpolate the fine grid error from the coarse grid error, etc.

The grid functions in this case are the discrete solution u_h, the residual r_h, the error e_h etc. Smoothing, transfer of grid functions between fine and coarse grid, and problem representation on the coarse grid, depend on discrete operators (S_h, I_h^H, I_H^h, A_H respectively) and are collectively called *multigrid components*. Multigrid components are not always trivial to choose, and it takes research, analysis and experience to successfully close in upon components, which when harnessed together, form an efficient

and robust algorithm. As we shall see, the work in this thesis can be viewed as an attempt to find good components in different situations.

2.2.3 The Multigrid Operator

In this section, we make a notational change for greater clarity. We assume a hierarchy of l grids such that Ω_0 stands for the finest grid, and Ω_l stands for the coarsest grids. Consequently, all grid functions (and operators) will carry these subscripts to indicate their grid levels. In this setting, k stands for a grid level, and $(k+1)$ is considered to be the next coarse level. Under this formulation, Algorithm 1 yields the 2-grid error iteration operator as (see Appendix A):

$$M_k^{k+1} = S_k^{\nu_2}\left(I_k - I_{k+1}^k A_{k+1}^{-1} I_k^{k+1} A_k\right) S_k^{\nu_1} \tag{2.10}$$

$$= S_k^{\nu_2} K_k^{k+1} S_k^{\nu_1} \tag{2.11}$$

The coarse grid correction operator $K_k^{k+1} = I_k - I_{k+1}^k A_{k+1}^{-1} I_k^{k+1} A_k$ is useless as an isolated iterative process, as it does not reduce high frequencies [2].

Standard (full) grid coarsening reduces the number of unknowns in the fine grid by a factor of 2^d. This implies that on coarse grids, formed by 2^d reduction of sufficiently fine grids, A_{k+1}^{-1} is still not a viable option. *A very important observation here is to note that exact inversion of the coarse grid operator is not necessary (either in theory or in practice).* The usual practice is to approximate the discrete coarse grid operator by recursively bringing the grid down to a coarser level. This recursion may be continued until the grid is coarse enough so that an exact inversion is optimal. The sequence of coarse errors is then interpolated back unrolling the recursion, so that e_h is finally obtained.

We see (from Appendix A) that this recursive error-correction scheme yields the so-called *multigrid error-iteration operator* as:

$$M_k = S_k^{\nu_2}\left[I_k - I_{k+1}^k \left\{I_{k+1} - (M_{k+1})^\gamma\right\} A_{k+1}^{-1} I_k^{k+1} A_k\right] S_k^{\nu_1} \tag{2.12}$$

with $k = 0, 1, 2, \cdots, l$ and $M_{l+1} = 0$.

It suffices to mention that these operators in the presented algebraic representation, are mainly used for theoretical convergence estimations. This purpose is served quite well by the 2-grid operator, but nevertheless the 3-grid operator and the k-grid operators have been used for deeper insights in [17] and [18], respectively.

2.3 Multigrid Components in Common Use

In this part, we will survey some multigrid components in common use. They include relaxation (error-smoothing) schemes, grid transfer techniques, coarse grid operators, and grid cycling methods. Multigrid mainly works as a balance between the relaxation method and the coarse grid correction process. *Anisotropy* and *coupling* in the discrete operator must be defined before multigrid components and therefore come first. Popular smoothing schemes follow. Then a brief survey of transfer operators, followed by commonly employed coarse grid correction operators, is given. Finally, we discuss

different recursions which produce the V, W, and the F-cycles. Mesh-independent convergence is a very well known virtue of multigrid methods, and is surveyed last in this section. It is important to note that the diffusion operator is usually denoted by Δ, and stands for $\sum_i {}^{\partial^2}/_{\partial x_i{}^2}$.

Definition 2.3.1. (Coupling and Anisotropy) *Two entities play a crucial role in defining coupling and anisotropy. They are the continuous operator and the discretization grid. We will consider concrete operators for this discussion. Consider the anisotropic diffusion operator in* $2d$*, given by:*

$$L = -\epsilon \frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2}; \qquad \epsilon \ll 1 \tag{2.13}$$

An $O(h^2)$ *central finite difference discretization of this operator on a regular equidistant grid (with mesh size* h*), gives the following stencil (computational molecule).*

$$L_h = \begin{bmatrix} & -1/h^2 & \\ -\epsilon/h^2 & (2\epsilon+2)/h^2 & -\epsilon/h^2 \\ & -1/h^2 & \end{bmatrix} \tag{2.14}$$

Evidently, the stencil coefficients coupling vertical neighbours are much larger in magnitude than the corresponding horizontal coefficients. Thus the discrete operator is strongly coupled in the y-direction compared to the x. It is well-known that local relaxation methods only smooth in the direction of strong coupling, such as the y-direction in this case. This directional drift in the discrete operator is known as anisotropy. Anisotropy in this case resulted due to the perturbation in the differential operator. Anisotropy can also result from discretization of a perfectly isotropic operator on non-equidistant and logarithmically stretched grids, and has consequences for multigrid convergence.

2.3.1 Some Commonly Used Relaxation Methods

Two main families of relaxation techniques can be broadly classified as the *point-relaxation method* and the *block-relaxation method*. Block-relaxation methods (e.g. *line* and *plane* smoothing) are mainly used for anisotropic PDEs. They are also somewhat expensive in terms of CPU-time as each relaxation step consists of solving a small linear system *exactly* (or to an acceptable accuracy). Fortunately, in most situations they can be substituted by point-relaxation methods, with some adjustment in the coarsening scheme (which takes care of the anisotropy). Here, we will focus on a survey of point-relaxation methods [8, 19, 20]. We use Equation 2.5 as our prototype and we drop the subscript h for brevity. The operator matrix A is square and has the order N. Elements of A are denoted by a and indexed by i and j, the row and the column indices respectively. D, L, and U in this section refers to the diagonal, the strictly lower triangular, -and the strictly upper triangular parts of A, respectively.

Description 2.3.1. (The ω-Jacobi Method) *We studied the error smoothing behaviour of the* ω*-Jacobi method in Section 2.1.2. In this method each unknown (grid-point) is updated using the values of its neighbours at the previous iteration. Newly updated*

values are not used until the next iteration. Update of the i^{th} unknown in the m^{th} ω-Jacobi iterate can be expressed as:

$$u_i^{(m)} = (1-\omega)u_i^{(m-1)} + \frac{\omega}{a_{i,i}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{N} a_{i,j}u_j^{(m-1)}\right)$$

Alternately, in matrix terms:

$$u^{(m)} = (I - \omega D^{-1}A)u^{(m-1)} + \omega D^{-1}b$$

where D is the diagonal matrix containing the main diagonal of the operator matrix A. ω-Jacobi is fully parallelizable (all unknowns can be simultaneously updated), and therefore the degree of parallelism of ω-Jacobi is said to be N. $0.5 \leqslant \omega \leqslant 0.8$ yields acceptable error-smoothing for very wide problem classes. It is important to note that the optimal relaxation parameter ω is d-dependent.

Description 2.3.2. (The Multi-parameter Jacobi Method) *This is a variant of ω-Jacobi. In multi-parameter Jacobi, each smoothing step consists of p, ω-Jacobi iterations, with a different relaxation parameter ω_j for each iteration j. A p-parameter Jacobi iteration (where $p \in \mathbb{Z}^+$), can be specified as:*

$$\begin{aligned} u^{(0)} &= u^m \\ u^{(j)} &= u^{(j-1)} + \omega_j D^{-1}\left(b - Au^{(j-1)}\right) \quad j = 1, 2, \cdots, p \\ \bar{u}^m &= u^{(p)} \end{aligned}$$

The optimal relaxation parameters ω_j have to be evaluated for the concrete operator that is being dealt in a particular situation. E.g. ω_j for the $O(h^2)$ FD discretization (of the d-dimensional Poisson equation [8]) are given by,

$$\omega_j = \left[\frac{2d+1}{2d} + \frac{2d-1}{2d}\cos\left(\frac{2j-1}{2p}\pi\right)\right]^{-1}$$

and for $O(h^4)$ FD long stencil discretization, by:

$$\omega_j = \left[\frac{16d+7}{15d} + \frac{16d-7}{15d}\cos\left(\frac{2j-1}{2p}\pi\right)\right]^{-1}$$

Description 2.3.3. (GS, The Gauss-Seidel Method) *In contrast with ω-Jacobi, the Gauss-Seidel method makes use of the latest available values of the neighbouring unknowns. The iteration can be written as:*

$$u_i^{(m)} = \frac{1}{a_{i,i}}\left(b_i - \sum_{j<i} a_{i.j}u_j^{(m)} - \sum_{j>i} a_{i,j}u_j^{(m-1)}\right)$$

Alternately in matrix terms:

$$u^{(m)} = (D+L)^{-1}\left(b - Uu^{(m-1)}\right)$$

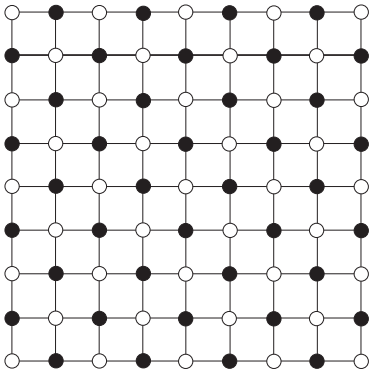


Figure 2.4: Red-Black layout of the grid points

We would like to point out that grid point ordering influences the result of a GS sweep. The updates are obviously sequential and therefore the degree of parallelism is $\sqrt{N}$ *which is not good. The smoothing properties however are fairly good for a big problem class, and can be further enhanced by the use of a relaxation parameter* ω*, which yields the SOR method.*

Description 2.3.4. (SOR, The Successive over-relaxation method) *When GS is used in conjunction with a relaxation parameter* ω*, it is called SOR. The error-smoothing properties of SOR are more pronounced than simple GS only for higher dimensional problems. The SOR iteration is:*

$$u_i^{(m)} = (1-\omega)u_i^{(m-1)} + \frac{\omega}{a_{i,i}}\left(b_i - \sum_{j<i} a_{i,j}u_j^{(m)} - \sum_{j>i} a_{i,j}u_j^{(m-1)}\right)$$

In alternate matrix form, this can be written as:

$$u^{(m)} = (D+\omega L)^{-1}\left[(1-\omega)D - \omega U\right]u^{(m-1)} + \omega(D+\omega L)^{-1}b.$$

The degree of parallelization is the same as that of GS, i.e., $\sqrt{N}$*. The smoothing greatly depends on the choice of the relaxation parameter* $0 < \omega < 2$*, and is usually expected to be slightly better than that of GS. Trivially,* $\omega = 1$ *yields the GS method.*

Description 2.3.5. (The ω-Red-Black Jacobi method) *The* ω*-RB Jacobi method is also known as the odd-even method due to the odd-even ordering of the grid points as shown in Figure 2.4 (which depicts the Red (odd) points by small empty circles, and the Black (even) points by solid black circles). Red points of the* m^{th} *iterate are updated by using the black points of the* $(m-1)^{th}$ *iterate. For the* $O(h^2)$ *FDM discretization (the usual* $(2d+1)$*-point stencil), an* ω*-RB Jacobi sweep is equivalent to an* ω*-Red-Black Gauss-Seidel sweep, because each color is independent of other members of the same color. However, for other discretizations, the two are not equivalent. In* ω*-RB Jacobi, each color is updated in the Jacobi fashion, while in* ω*-RB Gauss Seidel, the update of each color is done in the GS manner, using latest same color updates.*

The smoothing properties of ω-RB Jacobi are superior compared to ω-Jacobi, multi-parameter ω-Jacobi, Gauss-Seidel, and SOR. It's degree of parallelization is $N/2$ which is superior than simple sequential GS (often termed as lexicographic GS). For regular grids on which odd and even numbering is possible, ω-RB Jacobi is therefore a preferred smoother; however, operator properties also have to be considered. In Chapter 3, we provide Fourier smoothing analysis for it in a d-dimensional setting, which is used in Chapter 4 for developing the d-multigrid method. In Appendix D, we show how ω-RB-Jacobi might actually be implemented for a general d-dimensional grid.

Remark 2.3.1. (ILU Smoother) *The smoothing procedures discussed in this section are mainly based on local relaxation techniques. "Local" implies that the high-frequencies in the error are local or clustered. In such a situation local relaxation techniques do a fairly good smoothing job. However, in the case of anisotropic operators, the high frequencies occupy a larger frequency space; for this reason local relaxation techniques have to be adjusted (modified into block relaxation) to remain effective. Variants of ILU relaxation methods have the property that the rendered smoothing is largely global. Being a global smoother, ILU(0) is known to work better than local alternatives in situations where local smoothers perform poorly, such as in the standing wave context of Chapter 7.*

2.3.2 Some Robust Transfer Schemes

Here we will describe some transfer schemes for the restriction and prolongation of grid functions, in the context of $2d$ grids and standard coarsening. Higher dimensional grids and partial coarsening with $h \rightarrow 2h$ (doubling) and $h \rightarrow 4h$ (quadrupling) transfers are dealt with in Chapter 4, Section 4.3.4. We'd like to point out that for all elliptic second order PDEs, if bilinear interpolation is employed for prolongation (I_{2h}^{h}), a corresponding restriction operator (I_{h}^{2h}) can also be constructed as its adjoint. The following *adjoint rule* for transfer operators holds in the case of standard coarsening:

$$I_{2h}^{h} = 2^{d}\,(I_{h}^{2h})^{T} \quad \text{T represents the transpose} \tag{2.15}$$

Description 2.3.6. (Injection: (Restriction for VC grids)) *In vertex centered grid sequences, coarse grid nodes form a subset of fine grid nodes. In such a setting the simplest restriction is injection. The procedure consists of ignoring any information on the fine-grid-only nodes. Coupled with bilinear prolongation it works in many situations.*

Description 2.3.7. (2^{nd} order prolongation and restriction -VC grids) *Consider the $2d$ vertex-centered grid shown in Figure 2.5(a). Positions marked by • are shared by the coarse and the fine grids, so coarse grid values there are assumed to be fine grid values as well. Fine grid values at ⋆ are interpolated linearly from the coarse grid values • situated at their east and west. Similarly the fine values at □ are interpolated from the coarse values at • at their north and south. The fine values at △ are interpolated by four surrounding coarse values at • by the Four Point average. This results in*

overall bilinear interpolation. The stencil for this prolongation operator is given by:

$$I_{2h}^{h} \triangleq \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{2h}^{h}$$

The so-called full weighting (FW) restriction operator (described below) and bilinear prolongation operator obey the adjoint rule. This restriction operator employs full weighted averaging for restricting fine grid information to the coarse grid nodes. For standard coarsening in 2d, the FW restriction operator for vertex centered grids is given by the stencil:

$$I_{h}^{2h} \triangleq \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{h}^{2h}$$

This is a robust, and efficient restriction operator. In Chapter 4, Section 4.3.4, we show how 1d FW operators can be used in Kronecker tensor products to yield the actual restriction matrix.

Description 2.3.8. (7-point restriction and prolongation, -VC grids) *Another choice of transfer operators for 2d vertex centered grids are the 7-point transfer operators [8, 21]. The restriction operator for full coarsening in 2d is given by:*

$$I_{h}^{2h} \triangleq \frac{1}{8}\begin{bmatrix} 1 & 1 & \\ 1 & 2 & 1 \\ & 1 & 1 \end{bmatrix}_{h}^{2h}$$

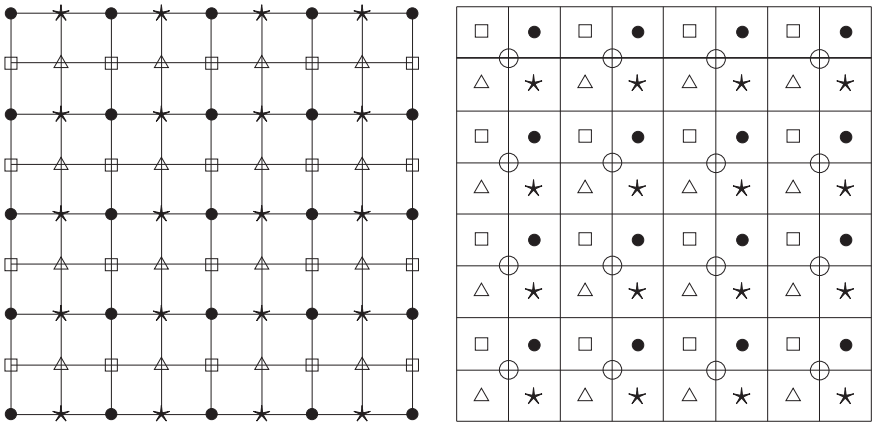
Similar to the FW restriction and bilinear prolongation, the prolongation operator in this case as well can be constructed as the adjoint of this restriction operator.

Description 2.3.9. (Four Point (FP) restriction, -CC grids) *The Four Point averaging restriction operator is more commonly used in a cell centered grid setting (employed in Chapter 6). Cell centered coarse grids are usually formed by agglomeration of fine grid cells, and therefore, coarse grid nodes do not form a subset of the fine grid nodes. The operator is given by:*

$$I_{h}^{2h} \triangleq \frac{1}{4}\begin{bmatrix} 1 & & 1 \\ & \cdot & \\ 1 & & 1 \end{bmatrix}_{h}^{2h}$$

This is also known as the piecewise constant restriction operator.

Description 2.3.10. (2^{nd} order prolongation and restriction -CC grids) *Compared to the vertex centered case, the situation is slightly different in the cell centered case shown in Figure 2.5(b). Here, coarse node positions ◯ are not shared by the fine grid. Fine grid values at positions marked by •, ⋆, △, and □, have to be interpolated by*



(a) Vertex centered Prolongation (b) Cell centered Prolongation

Figure 2.5: Differently marked grid nodes are interpolated differently owing to their geometric proximity to the coarse grid node.

the surrounding coarse grid values at ◯. The averaging weights for each position are given by the following stencil:

$$I_{2h}^{h} \triangleq \frac{1}{16}\left[\begin{array}{ccc|ccc} 1 & & 3 & 3 & & 1 \\ & \square & & & \bullet & \\ 3 & & 9 & 9 & & 3 \\ \hline 3 & & 9 & 9 & & 3 \\ & \triangle & & & \star & \\ 1 & & 3 & 3 & & 1 \end{array}\right]^{h}_{2h}$$

With the adjoint rule given in Equation 2.15, a 2^{nd} order restriction operator can be easily constructed through this prolongation operator.

Remark 2.3.2. (Order of restriction and prolongation) *It is usual practice to employ the FW restriction and bilinear prolongation in multigrid treatment of a wide class of problems on structured grids. The order of interpolation is defined as 1 more than the degree of the highest degree polynomial class that the interpolation procedure can retrieve exactly. Therefore, the order of both these operators is 2. [22] shows that for a successful multigrid treatment the sum of the orders of the transfer operator, should be larger than the order of the differential operator. The drawback of this rule is that smoothness of the error after relaxation is not taken into consideration, which might allow for cheaper restriction operators. This explains why Injection sometimes works with bilinear interpolation for nicely elliptic operators even if this rule is slightly violated.*

2.3.3 Coarse Grid Operators.

In many cases, the re-discretization of the differential operator on the coarse grid serves well as the operator approximation on the coarse grid and is traditionally known as the *discretization coarse grid operator (DCG)*. However, with situations such as jumping (or strongly varying) PDE coefficients, this approximation is often inadequate.

A more robust choice for the coarse grid operator stems from the following observation:

$$A_H e_H = r_H = I_h^H r_h = I_h^H A_h e_h = I_h^H A_h I_H^h e_H$$

which leads to the general recursion:

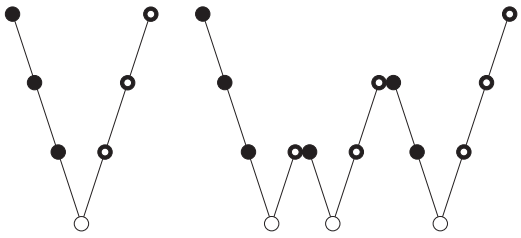
$$A_{k+1} = I_k^{k+1} A_k I_{k+1}^k$$

This is popularly known as the *Galerkin coarse-grid operator GCG*. Although GCG is more general in its range of applicability, it has an associated expense in terms of growing stencils. For $O(h^2)$ FDM discretization, on structured $2d$ grids, GCG stays within a 9-point difference stencil, but for unstructured grids, the growth can be quite large. For this reason, it often pays to check out if DCG can be satisfactorily applied in a given situation.

Remark 2.3.3. (5-point GCG for $2d$ applications) *In view of $2d$ applications, we note here, is that if the 7-point transfer operators given in Section 2.3.2 are employed in conjunction with the usual 5-point FDM discretization of the Laplacian, the GCG operator stays within a 5-point connectivity stencil [8], i.e. it has the same sparsity as the original operator. This is a very attractive property, but unfortunately, a $1d$ version of these transfer operators (which might be extended to arbitrary d by Kronecker tensor products) is not readily available.*

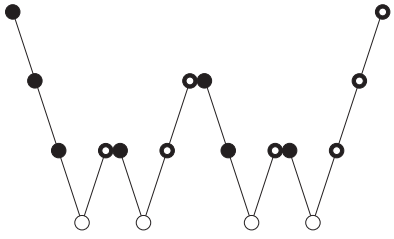
2.3.4 Multigrid Cycle-types and Optimality

In the multigrid method, the coarse grid defect equation can be solved approximately as we saw in Section 2.2.3, by recursively coarsening the grid further and using γ cycles on the coarse grid. γ is conventionally called *the cycle index* [8]. In many practical situations, where the multigrid components can be optimally chosen, $\gamma = 1$ works very well [4, 23], and results in the so-called V-cycle, shown in Figure 2.6(a). However, in some situations, particularly, where using GCG is not an option (due to algorithmic restrictions), and DCG is somewhat inadequate, $\gamma = 1$, i.e. the V-cycle isn't sufficient. As a remedy, $\gamma = 2$ can be used and results in the so-called W-cycle as shown in Figure 2.6(c). In a W-cycle multigrid (resulting from $\gamma = 2$), 2 cycles are used on each coarse level, which increases the computational expense, and therefore, constant values of $\gamma > 2$ are seldom employed in practice. A variable value of γ on different coarse levels is more promising. The most frequently employed cycle , i.e. the F-cycle, is constructed by calling multigrid recursively γ times in a loop, such that in the first iteration of the loop, user defined γ is used (usually $\gamma = 2$), while in the rest, $\gamma = 1$ is employed. An F-cycle constructed with $\gamma = 2$ is shown in Figure 2.6(b).



(a) A 4-grid multigrid V-cycle

(b) A 4-grid multigrid F-cycle



(c) A 4-grid multigrid W-cycle

Figure 2.6: Different multigrid cycles for a 4-grid method. ● = presmoothing, ⊙ = postsmoothing, ○ = exact solution, \ = Restriction, / = Prolongation

A unified multigrid method, incorporating all the three popular cycle types, is listed in Algorithm 2.

One of the most attractive features of a multigrid algorithm is its mesh size independent convergence rate, in contrast with simple stationary methods (such as GS), which exhibits convergence rate deterioration on finer meshes compared to coarser ones. Besides, the computational complexity of a *full multigrid method* is $O(M_0)$, where M_0 is the total number of unknowns in the finest (level 0) system. The optimal complexity, and the mesh free convergence rate of multigrid, make this technique very attractive for general use. In Chapter 4, Section 4.3.5 we derive work estimates for multigrid in a general d-dimensional setting.

At this stage, we present multigrid in a unified algorithm where all the three cycle types discussed above can be achieved through parameter variation. l represents a particular multigrid level. Larger l stand for coarser grids, and is used as a subscript with grid functions to indicate the level number. u^m represents the solution after the m^{th} iterate. *cycle* is a text indicator for specifying the particular cycle type, this specification is accompanied by the cycle index γ which specifies how many basic V-type recursions the algorithm should take. C is the coarsest grid level that the algorithm is allowed to attain.

Algorithm 2 Multigrid pseudocode

$$u_l^{m+1} = \mathbf{MG}(l, \gamma, cycle, u_l^m, \boldsymbol{A}_l, b_l, \nu_1, \nu_2).$$

(0) Initialization

– If $l = C$, $u_l^{m+1} =$ **exact** $(\boldsymbol{A}_l, b_l)$; Bail out; endif
– Build the coarse-grid operator $\boldsymbol{A}_{l+1}$, and the restriction I_l^{l+1}, and prolongation I_{l+1}^l operators.

(1) Pre–smoothing

– Compute $\overline{u}_l^m$ by applying $\nu_1 (\geq 0)$ smoothing steps to u_l^m :
$\overline{u}_l^m =$ **smooth**$^{\nu_1}(u_l^m, \boldsymbol{A}_l, b_l)$.

(2) Coarse grid correction

– Compute the residual $\overline{r}_l^m = b_l - \boldsymbol{A}_l \overline{u}_l^m$.
– Restrict the residual $\overline{r}_{l+1}^m = I_l^{l+1} \ \overline{r}_l^m$.
– Compute the approximate error $\widehat{e}_{l+1}^m$ from the *defect equation*. $\boldsymbol{A}_{l+1} \ \widehat{e}_{l+1}^m = \overline{r}_{l+1}^m$

by the following mini algorithm

> If $l = C$, $\hat{e}_{l+1}^m =$ **exact** $(\boldsymbol{A}_{l+1}, \overline{r}_{l+1}^m)$; endif
> If $l < C$, solve for $\hat{e}_{l+1}^m$ approximately by the recursion:
> $\hat{e}_{l+1}^{m,1} = \overline{0}$;
> do $i = 1$ to γ
> If $cycle = f$ and $i \neq 1$,
> $\hat{e}_{l+1}^{m,i+1} =$ **MG**$(l+1, 1, cycle, \hat{e}_{l+1}^{m,i}, \boldsymbol{A}_{l+1}, \overline{r}_{l+1}^m, \nu_1, \nu_2)$
> else
> $\hat{e}_{l+1}^{m,i+1} =$ **MG**$(l+1, \gamma, cycle, \hat{e}_{l+1}^{m,i}, \boldsymbol{A}_{l+1}, \overline{r}_{l+1}^m, \nu_1, \nu_2)$
> endif
> continue i
> endif

– Interpolate the correction $\widehat{e}_l^m = I_{l+1}^l \ \widehat{e}_{l+1}^m$.
– Compute the corrected approximation on Ω_l $u_l^{m+\frac{1}{2}} = \overline{u}_l^m + \widehat{e}_l^m$.

(3) Post–smoothing

– Compute u_l^{m+1} by applying $\nu_2 \ (\geq 0)$ smoothing steps to $u_l^{m+\frac{1}{2}}$:
$u_l^{m+1} =$ **smooth**$^{\nu_2}(u_l^{m+\frac{1}{2}}, \boldsymbol{A}_l, b_l)$.

2.4 Multigrid Analysis

2.4.1 Qualitative / Quantitative Approaches.

Trottenberg et al list four substantially different approaches to analyzing multigrid [8]. They are:

- The classical (qualitative) multigrid theory.
- The classical multigrid theory based on subspace splitting.
- Rigorous Fourier analysis.
- Local Fourier analysis.

The first is a qualitative approach proposed primarily by Hackbusch [4] and Braess, and developed by others such as Bramble [24]. The second is also a (different) qualitative approach proposed by Xu [23]. Qualitative approaches validate the multigrid idea in classical formulations, and are aimed at providing convergence proofs. As multigrid has been applied to a host of diverse problems, with a multitude of different components, the developments in the qualitative approach doesn't provide a qualitative versus quantitative coverage. In this thesis, we remain focused on improving multigrid convergence factors through coarse grid correction techniques, and therefore, do not explore the qualitative approaches any further.

Multigrid is a complex algorithm and is analyzed by studying its constituent parts. Although the eventual decay of the error depends on the complete algorithm (which includes the transfer operators), the component that plays a central role in it is the relaxation method. In turn, a relaxation method is characterized by its iteration matrix which is responsible for reducing the numerical error. We already discussed in Section 2.1.2 that the initial error can be represented as a linear combination of the eigenvectors of the iteration matrix. These eigenvectors span a vector space within the confines of which the iteration matrix operates. In other words, this means that this vector space is invariant with respect to the linear mapping defined by the iteration matrix. As it turns out, an invariant space is essential for analyzing multigrid operators (or components) quantitatively, and is not always readily available in the rigorous sense.

The Rigorous Fourier analysis is a quantitative approach, which is also rather limited in its range of applicability. It is characterized by considerations of the actual finite domain, with prescribed boundary conditions of Dirichlet, Neumann or periodic -types. It can give precise estimates, and predict 2-grid convergence factors to a good accuracy. The steps involved in this analysis are (1) to evaluate the actual eigenvectors of (various) iteration operators and (2) to impose bounds on their associated eigenvalues. We presented a slight flavor of it in Section 2.1.2. It is important to realize that not all discrete operators and boundary conditions lend themselves to an accessible rigorous approach, and therefore its coverage of interesting problems, is inadequate as well. The most widely employed tool for multigrid analysis is known as *Local Fourier Analysis* LFA. We deal with this topic in Chapter 3.

2.4.2 h-ellipticity and Implications for Anisotropy

Here we do not provide a mathematical definition of h-ellipticity. Our purpose here is only to indicate that h-ellipticity is the *amount of ellipticity* that a particular discrete operator possesses. A strictly positive measure indicates in turn, that when this operator is employed in a particular multigrid cycle, the oscillatory error components would be *local*, and can possibly be corrected through an appropriate point smoother. This measure can be quantified [1, 2, 8] through Fourier symbols[1] of the discrete operator, and takes into account the coarsening strategy. E.g., the h-ellipticity of the 5-point FDM discretization (of the Laplacian) on regular equidistant meshes (under standard coarsening) is 0.25. The h-ellipticity of the same discretization and coarsening scheme for the anisotropic diffusion equation in Equation 2.13, $\epsilon/(2+2\epsilon) \rightarrow 0$ for $\epsilon \rightarrow 0$. However, if the coarsening strategy only coarsens the grid in the y-direction, then the h-ellipticity is $1/(2+2\epsilon) \rightarrow 1/2$, which suggests one way of dealing with this equation.

2.4.3 Towards Point Smoothing and Adaptive Coarsening

In Chapters 4 and 5, we address the problem of the so-called grid-aligned anisotropies. There are two different classes of remedy available to handle discrete anisotropies. They can roughly be classified as *adaptive smoothing* and *adaptive coarsening*. Both remedies have their disadvantages and drawbacks. Most adaptive smoothing remedies are based on block smoothing, which usually means exact solution of a subset of variables (represented by a sub-block) in the system, at every smoothing sweep. Line smoothing in $2d$, and plane smoothing[2] in $3d$ falls under this category. For anisotropies (in a d-dimensional operator) introduced due to discretization on non-equidistant grids (such as used in Chapter 5, in a sparse grid setting), smoothing along hyper-planes becomes expensive. We propose an alternate solution based on adaptive coarsening. We coarsening only along those dimensions that satisfy a certain constraint on the coupling measure, presented in Chapter 4, Equation 4.13. Through this strategy we can retain point-smoothing and still achieve good multigrid convergence factors.

In Chapter 6, we take on the problem of efficient multigrid solution for logarithmically stretched grids. These grids have a monotonically increasing mesh size along both directions, and geometric multigrid methods work poorly for them, due to ineffective smoothing. Again, we solve this problem via the adaptive coarsening route. Some adaptive coarsening methods already exist for tackling these problems. One of them is called *Multiple Semicoarsened Grids (MSG)* [25, 26, 27]. This approach is based on generating multiple coarse grids, and interpolating the solutions in a manner that resembles the sparse grid technique. Grids constructed from an MSG sequence are shown in Figure 2.7.

MSG is easy to implement, and constructs coarse grids along all dimensions. That already hints that the technique may not be very useful for higher dimensions. Another technique which depends on *Conditional Semicoarsening* [28] depends on choosing

[1] Fourier symbols of the continuous and discrete d-dimensional Laplacian are given in Appendix B.

[2] Typically inexact plane smoothing which gives a 10% residual reduction, is sufficient in most cases.

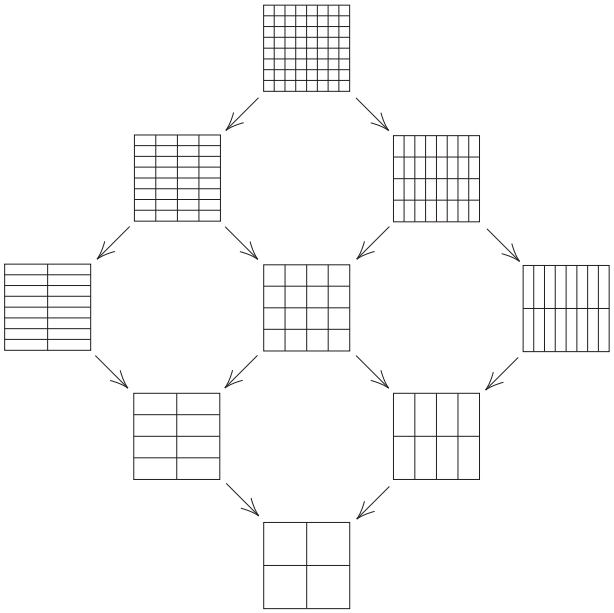


Figure 2.7: Grid sequence resulting from the MSG method

removable grid lines through a Fourier Analysis. Depending on the actual stretch of the grid, this results in coarsening within a tightly coupled subdomain only, and therefore has issues with computational complexity. We, in Chapter 6, also use Fourier analysis, but do not seek ideal smoothing factors for making a coarsening choice. During our research for obtaining the optimal coarsening strategy for non-equidistant grids, we found that any dimension with a mesh size variation that stays within 1.3 times the size of the coarsening candidate, can be coarsened together with it. This led to the idea of optimizing individual mesh aspect ratios for designing a coarsening strategy for stretched grids. A detailed exposition of this venture is the subject of Chapter 6.

The work in Chapter 6 unveiled an alternative technique for locally refined grids. These grids are usually coarsened only in the refined subdomain. We found that standard coarsening throughout the grid, along with bilinear interpolation at layer interfaces works well, even for quite demanding situations. There, in Chapter 7, we provide a detailed overview of this strategy.

Chapter 3

LFA of ω-RB Jacobi in d-dimensions

LFA was first introduced by Achi Brandt, as local mode analysis [1]. Here we will introduce LFA and cover only those aspects of it that are directly related to the work in this thesis, such as, the multigrid treatment of anisotropy for d-dimensional PDEs. See [8, 9] for a bigger exposition. In Chapter 4, we employ partial and full coarsening strategies to handle anisotropies resulting from discretization on non-equidistant grids. These coarsening strategies employ both doubling (i.e. $h \rightarrow 2h$) as well as quadrupling (i.e. $h \rightarrow 4h$) transfers. Therefore, in this chapter we present LFA with this requirement in view. Part of this work was published in our paper [29].

3.1 Local Fourier Analysis (LFA)

3.1.1 LFA Assumptions

The prominent differences of LFA with the other Rigorous Fourier approach are:

- The numerical error in LFA consists of the general Fourier mode, $\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = e^{\iota\boldsymbol{\theta}\cdot\boldsymbol{x}/h}$, where the frequency of this Fourier mode is $\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_d)$, such that θ_i varies *continuously* in the interval $(-\pi, \pi]$. ($\iota = \sqrt{-1}$)

- The LFA is valid only in the context of an infinite grid defined by:

$$G_h := \left\{ \boldsymbol{x} = (x_1, \ldots, x_d)^T = \boldsymbol{\kappa} h = h(\kappa_1, \ldots, \kappa_d)^T \quad : \quad \boldsymbol{\kappa} \in \mathbb{Z} \right\}$$

  which implies that all boundary considerations have been neglected, and all operators are extended to this infinite grid. The coarse infinite grid G_{2h} is defined similarly.

3.1.2 The Harmonics

In LFA, we distinguish between two distinct kinds of Fourier components, the *high frequency components* and the *low frequency components*. *The categorization of the frequencies $\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_d)$ as high or low (in a 2-grid setting) depends on the coarsening strategy that is employed.* We will come back to this point in greater detail in Chapter 4, Section 4.4. For now, it suffices to discuss the case of standard coarsening. As 2π is the period of φ, we are led to the identity:

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) \equiv \varphi(\boldsymbol{\theta}', \boldsymbol{x}) \qquad \text{for } \boldsymbol{x} \in G_h \quad \text{iff } \boldsymbol{\theta} = \boldsymbol{\theta}' (\text{mod} \quad 2\pi). \tag{3.1}$$

where this difference of -multiples of 2π- is between all the components of the d-tuples, ($\boldsymbol{\theta}$ & $\boldsymbol{\theta}'$), thus it suffices to consider these functions only for $\boldsymbol{\theta} \in [-\pi, \pi)^d$.

The distinction between these two kinds of Fourier components depends on the coarsening scheme. For full coarsening ($h \to 2h$ transfers along all dimensions):

$$\begin{aligned} \varphi \text{ is a low freq. component } &\iff \boldsymbol{\theta} \in T^{low} := \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^d. \\ \varphi \text{ is a high freq. component } &\iff \boldsymbol{\theta} \in T^{high} := [-\pi, \pi)^d \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^d \end{aligned} \tag{3.2}$$

the underlying fact being the high frequency error components are those that are not *visible* on the coarse grid. It is straight-forward to see that:

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) \equiv \varphi(\boldsymbol{\theta}', \boldsymbol{x}) \qquad \text{for } \boldsymbol{x} \in G_{2h} \quad \text{iff } \boldsymbol{\theta} = \boldsymbol{\theta}' (\text{mod} \quad \pi). \tag{3.3}$$

and therefore we can define the 2^d dimensional spaces of *harmonics* for any $\boldsymbol{\theta} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^d$; E_h^θ as:

$$\begin{aligned} E_h^\theta &= \text{span} \left\{ \varphi_h(\boldsymbol{\theta}', \boldsymbol{x}) : \boldsymbol{\theta}' \in [-\pi, \pi)^d \right\} \\ &= \text{span} \left\{ \varphi_h(\boldsymbol{\theta}_{\boldsymbol{\alpha}}, \boldsymbol{x}) : \boldsymbol{\alpha} = (\alpha^1, \alpha^2, \cdots, \alpha^d), \quad \alpha^j \in \{0, 1\} \right\} \end{aligned}$$

with

$$\boldsymbol{\theta}_{\boldsymbol{\alpha}} := \boldsymbol{\theta} - (\boldsymbol{\alpha} \diamond \text{sign}\,(\boldsymbol{\theta}))\,\pi, \quad \text{sign}\,(\boldsymbol{\theta}) := \begin{cases} 1 & (\boldsymbol{\theta} > 0) \\ -1 & (\boldsymbol{\theta} \leqslant 0) \end{cases}; \quad \left(-\frac{\pi}{2} \leqslant \boldsymbol{\theta} < \frac{\pi}{2}\right) \tag{3.4}$$

We use $\diamond$ to denote component-wise product between two vectors, such that, for arbitrary vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ of length d, $\boldsymbol{u} \diamond \boldsymbol{v}$ is a vector of d components $u_i v_i$. Henceforth, we use the following notation for brevity:

$$\begin{aligned} \varphi_{\boldsymbol{\alpha}} = \varphi_h(\boldsymbol{\theta}_{\boldsymbol{\alpha}}, \boldsymbol{x}), \quad (\boldsymbol{x} \in G); \\ \boldsymbol{\theta}_{\alpha_1 \alpha_2 \cdots \alpha_d} = \boldsymbol{\theta}_{(\alpha_1, \alpha_2, \cdots, \alpha_d)} \end{aligned} \tag{3.5}$$

3.2 Fourier Representation of Relaxation Operators

Let S_h represent a general (local) relaxation process, encapsulating commonly used smoothers such as ω-Jacobi, GS, and SOR. It suffices to mention that under these relaxation operations the *space of harmonics* remains invariant, i.e.:

$$S_h : E_h^{\boldsymbol{\theta}} \longrightarrow E_h^{\boldsymbol{\theta}} \qquad\qquad -\pi/2 < \theta_i < \pi/2 \tag{3.6}$$

Consider a division of the infinite discrete domain G_h into three disjoint sets G_h°, G_h^+, G_h^-; so that:

$$G_h = G_h^\circ \cup G_h^+ \cup G_h^-$$

G_h° represents those points on the grid, where values would be updated simultaneously. G_h^+ represents those points where new values are available, and G_h^- represents those points where old values would be used within the relaxation step. In order to develop a general representation of the relaxation operator, consider the regular splitting of L_h, given by:

$$L_h = L_h^M - L_h^N$$

Following the discrete form given in Equation 2.4, an iteration of the relaxation process can be written as:

$$u_h^{i+1} = (L_h^M)^{-1} L_h^N u_h^i + (L_h^M)^{-1} f_h \tag{3.7}$$

We now define the L_h^M and L_h^N, such that:

$$\begin{aligned} L_h^M &= \frac{1}{\omega}(L_h^\circ + \omega L_h^+) \\ L_h^N &= \left(\frac{1}{\omega} - 1\right) L_h^\circ + L_h^- \end{aligned}$$

where ω is the relaxation parameter. We readily observe that this definition is equivalent to $L_h = L_h^\circ + L_h^+ + L_h^-$, where the partitions L_h°, L_h^+, and L_h^- correspond to G_h°, G_h^+, and G_h^- respectively. Equation 3.7, under this definition gives:

$$\begin{aligned} u_h^{i+1} &= \omega\Big(L_h^\circ + \omega L_h^+\Big)^{-1} \left\{\left(\frac{1}{\omega} - 1\right) L_h^\circ + L_h^-\right\} u_h^i + \omega\Big(L_h^\circ + \omega L_h^+\Big)^{-1} f_h \\ &= \Big(L_h^\circ + \omega L_h^+\Big)^{-1} \Big[(1-\omega) L_h^\circ + L_h^-\Big] u_h^i + \omega\Big(L_h^\circ + \omega L_h^+\Big)^{-1} f_h \end{aligned} \tag{3.8}$$

Equation 3.8 embodies one iteration of a simple relaxation method. The difference of two successive iterates gives the error iteration operator S_h as:

$$S_h = \Big(L_h^\circ + \omega L_h^+\Big)^{-1} \Big[(1-\omega) L_h^\circ + L_h^-\Big] \tag{3.9}$$

with corresponding Fourier symbols given as:

$$A(\boldsymbol{\theta}, h;\ \omega) = \frac{(1-\omega)\widetilde{L}_h^\circ(\boldsymbol{\theta}) - \omega\widetilde{L}_h^-(\boldsymbol{\theta})}{\widetilde{L}_h^\circ(\boldsymbol{\theta}) + \omega\widetilde{L}_h^+(\boldsymbol{\theta})} \tag{3.10}$$

where $\widetilde{L}_h^\circ$, $\widetilde{L}_h^+$, and $\widetilde{L}_h^-$ are the Fourier symbols of L_h°, L_h^+, and L_h^- respectively.

The symbols $A(\boldsymbol{\theta}, h;\ \omega)$ given in Equation 3.10 represent the *amplification factor* by which the Fourier component $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ amplifies/reduces after successive relaxation sweeps of the mentioned methods (i.e. ω-Jacobi, GS or SOR). Therefore, for these methods, the amplification factor is also equivalent to their eigensymbols, i.e.

$\widetilde{S}_h(\boldsymbol{\theta};\omega) = A(\boldsymbol{\theta}, h;\ \omega)$. This is due to the fact that these methods do not mix Fourier components. However, pattern relaxation methods do not preserve Fourier components, and therefore, the amplification factor does not represent their eigensymbol. The complete eigensymbol for ω-RB Jacobi is represented by an amplification matrix with amplification factors of aliasing components coupled in 2×2 matrix blocks. In Chapter 3, we clarify this further, through a $3d$ derivation of the eigensymbols for ω-RB Jacobi, and subsequent extension of the results to general d-dimensions.

3.3 Smoothing Analysis of ω-RB Jacobi

In this section, we model the analysis of ω-RB Jacobi on a $3d$ system. $h \longrightarrow 2h$ transfer is implicitly assumed. Later, we show how to extend this analysis to d dimensions, as well as how to incorporate quadrupling, i.e., $h \longrightarrow 4h$ transfer. A smoothing analysis for a similar case appears in [9], that follows a particular prescribed ordering of the basis functions φ_{α}, a generalization to d dimensions is therefore not clear. The ordering considered here is the simple "binary"order, which can immediately be extended to d-dimensions. This means that for a $3d$ system the multi-index $\boldsymbol{\alpha}$ reads:

$$(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)$$

Table 3.1: This table shows aliasing of the Fourier modes φ_{α} with φ_{000} on the coarse grid (standard coarsening), for $\boldsymbol{x}$ belonging to each of the distinct 2^d odd-even categories. This is a direct consequence of Equations 3.3 and 3.4.

$\boldsymbol{x} \in$	φ_{000}	φ_{001}	φ_{010}	φ_{011}	φ_{100}	φ_{101}	φ_{110}	φ_{111}
G_h^{000}	φ_{000}	φ_{000}	φ_{000}	φ_{000}	φ_{000}	φ_{000}	φ_{000}	φ_{000}
G_h^{001}	φ_{000}	$-\varphi_{000}$	φ_{000}	$-\varphi_{000}$	φ_{000}	$-\varphi_{000}$	φ_{000}	$-\varphi_{000}$
G_h^{010}	φ_{000}	φ_{000}	$-\varphi_{000}$	$-\varphi_{000}$	φ_{000}	φ_{000}	$-\varphi_{000}$	$-\varphi_{000}$
G_h^{011}	φ_{000}	$-\varphi_{000}$	$-\varphi_{000}$	φ_{000}	φ_{000}	$-\varphi_{000}$	$-\varphi_{000}$	φ_{000}
G_h^{100}	φ_{000}	φ_{000}	φ_{000}	φ_{000}	$-\varphi_{000}$	$-\varphi_{000}$	$-\varphi_{000}$	$-\varphi_{000}$
G_h^{101}	φ_{000}	$-\varphi_{000}$	φ_{000}	$-\varphi_{000}$	$-\varphi_{000}$	φ_{000}	$-\varphi_{000}$	φ_{000}
G_h^{110}	φ_{000}	φ_{000}	$-\varphi_{000}$	$-\varphi_{000}$	$-\varphi_{000}$	$-\varphi_{000}$	φ_{000}	φ_{000}
G_h^{111}	φ_{000}	$-\varphi_{000}$	$-\varphi_{000}$	φ_{000}	$-\varphi_{000}$	φ_{000}	φ_{000}	$-\varphi_{000}$

We distinguish between 2^d different categories of grid-points given by:

$$G_h^{\alpha} := \{\boldsymbol{x} = \boldsymbol{\kappa} h, \quad \boldsymbol{\kappa} \in \mathbb{Z}^d, \ \ \boldsymbol{\kappa} = \boldsymbol{\alpha} \ \ (mod\ \ 2)\} \tag{3.11}$$

Note that in $3d$ there would be in all 8 different categories of grid-points from an odd-even perspective. Table 3.1 gives the aliasing of the Fourier modes for each of these categories.

For the purpose of representing a general grid function (such as the computational error) in terms of the Fourier components, we construct 8 functions ψ_i. Each ψ_i is

Table 3.2: The functions ψ_i for each category of grid points.

$\boldsymbol{x} \in \rightarrow$	G_h^{000}	G_h^{001}	G_h^{010}	G_h^{011}	G_h^{100}	G_h^{101}	G_h^{110}	G_h^{111}
ψ_1	φ_{000}	0	0	0	0	0	0	0
ψ_2	0	φ_{000}	0	0	0	0	0	0
ψ_3	0	0	φ_{000}	0	0	0	0	0
ψ_4	0	0	0	φ_{000}	0	0	0	0
ψ_5	0	0	0	0	φ_{000}	0	0	0
ψ_6	0	0	0	0	0	φ_{000}	0	0
ψ_7	0	0	0	0	0	0	φ_{000}	0
ψ_8	0	0	0	0	0	0	0	φ_{000}

defined by summing up the elements of the i^{th} row of Table 3.1 and substituting the appropriate aliasing mode φ_α from the corresponding entry in the top row. This manipulation leads to the following set of equations defining ψ_i.

$$\begin{aligned}
\psi_1 &= \{\varphi_{000} + \varphi_{001} + \varphi_{010} + \varphi_{011} + \varphi_{100} + \varphi_{101} + \varphi_{110} + \varphi_{111}\}/8 \\
\psi_2 &= \{\varphi_{000} - \varphi_{001} + \varphi_{010} - \varphi_{011} + \varphi_{100} - \varphi_{101} + \varphi_{110} - \varphi_{111}\}/8 \\
\psi_3 &= \{\varphi_{000} + \varphi_{001} - \varphi_{010} - \varphi_{011} + \varphi_{100} + \varphi_{101} - \varphi_{110} - \varphi_{111}\}/8 \\
\psi_4 &= \{\varphi_{000} - \varphi_{001} - \varphi_{010} + \varphi_{011} + \varphi_{100} - \varphi_{101} - \varphi_{110} + \varphi_{111}\}/8 \\
\psi_5 &= \{\varphi_{000} + \varphi_{001} + \varphi_{010} + \varphi_{011} - \varphi_{100} - \varphi_{101} - \varphi_{110} - \varphi_{111}\}/8 \\
\psi_6 &= \{\varphi_{000} - \varphi_{001} + \varphi_{010} - \varphi_{011} - \varphi_{100} + \varphi_{101} - \varphi_{110} + \varphi_{111}\}/8 \\
\psi_7 &= \{\varphi_{000} + \varphi_{001} - \varphi_{010} - \varphi_{011} - \varphi_{100} - \varphi_{101} + \varphi_{110} + \varphi_{111}\}/8 \\
\psi_8 &= \{\varphi_{000} - \varphi_{001} - \varphi_{010} + \varphi_{011} - \varphi_{100} + \varphi_{101} + \varphi_{110} - \varphi_{111}\}/8
\end{aligned} \tag{3.12}$$

Observe from the above definitions that each ψ_i is non-zero for a distinct G_h^α. For convenience this fact is represented in Table 3.2. From this table it is clearly evident that any Fourier mode in the 8-dimensional Fourier space can be written as a linear combination of ψ_i. Forming linear combinations with an arbitrary $c^\alpha \in \mathbb{C}$ and $\boldsymbol{x} \in G_h^\alpha$, we have the linear combinations:

$$\begin{aligned}
c^{000}\psi_1 + c^{001}\psi_2 + c^{010}\psi_3 + c^{011}\psi_4 + c^{100}\psi_5 + c^{101}\psi_6 + c^{110}\psi_7 + c^{111}\psi_8 &= c^\alpha\varphi_{000}, \\
c^{000}\psi_1 - c^{001}\psi_2 + c^{010}\psi_3 - c^{011}\psi_4 + c^{100}\psi_5 - c^{101}\psi_6 + c^{110}\psi_7 - c^{111}\psi_8 &= c^\alpha\varphi_{001}, \\
c^{000}\psi_1 + c^{001}\psi_2 - c^{010}\psi_3 - c^{011}\psi_4 + c^{100}\psi_5 + c^{101}\psi_6 - c^{110}\psi_7 - c^{111}\psi_8 &= c^\alpha\varphi_{010}, \\
c^{000}\psi_1 - c^{001}\psi_2 - c^{010}\psi_3 + c^{011}\psi_4 + c^{100}\psi_5 - c^{101}\psi_6 - c^{110}\psi_7 + c^{111}\psi_8 &= c^\alpha\varphi_{011}, \\
c^{000}\psi_1 + c^{001}\psi_2 + c^{010}\psi_3 + c^{011}\psi_4 - c^{100}\psi_5 - c^{101}\psi_6 - c^{110}\psi_7 - c^{111}\psi_8 &= c^\alpha\varphi_{100}, \\
c^{000}\psi_1 - c^{001}\psi_2 + c^{010}\psi_3 - c^{011}\psi_4 - c^{100}\psi_5 + c^{101}\psi_6 - c^{110}\psi_7 + c^{111}\psi_8 &= c^\alpha\varphi_{101},
\end{aligned}$$

$$
\begin{aligned}
c^{000}\psi_1 + c^{001}\psi_2 - c^{010}\psi_3 - c^{011}\psi_4 - c^{100}\psi_5 - c^{101}\psi_6 + c^{110}\psi_7 + c^{111}\psi_8 &= c^{\alpha}\varphi_{110}, \\
c^{000}\psi_1 - c^{001}\psi_2 - c^{010}\psi_3 + c^{011}\psi_4 - c^{100}\psi_5 + c^{101}\psi_6 + c^{110}\psi_7 - c^{111}\psi_8 &= c^{\alpha}\varphi_{111},
\end{aligned}
\tag{3.13}
$$

It is important to point out that the index set α is chosen as a superscript for the constant c merely for notational convenience and does *not* imply association to particular grid points. Also, the definition of each of these modes in terms of ψ_i is absolutely independent of each other.

As the ω-RB Jacobi relaxation consists of the two partial ω-Jacobi steps in succession, the relaxation operator can be written as:

$$
\begin{aligned}
S_h^{RB} &= S_h^R S_h^B \\
\Rightarrow \quad \widetilde{S}_h^{RB} &= \widetilde{S}_h^R \widetilde{S}_h^B
\end{aligned}
\tag{3.14}
$$

In d dimensions, each of the 2^d category of grid-points (given by Equation 3.11) can be categorized as either belonging to the red color or else to the black color by the following rule:

$$
\begin{aligned}
\boldsymbol{G}_h^{\boldsymbol{\alpha}} &= \boldsymbol{G}_h^{red}, \text{ if } (\sum_{j=1}^{d} \alpha^j) \ \ mod \ \ 2 = 0 \\
\boldsymbol{G}_h^{\boldsymbol{\alpha}} &= \boldsymbol{G}_h^{blk}, \text{ if } (\sum_{j=1}^{d} \alpha^j) \ \ mod \ \ 2 = 1
\end{aligned}
$$

Thus, we recognize $\boldsymbol{G}_h^{red}$ as $\{\boldsymbol{G}_h^{000} \cup \boldsymbol{G}_h^{011} \cup \boldsymbol{G}_h^{101} \cup \boldsymbol{G}_h^{110}\}$ and $\boldsymbol{G}_h^{blk}$ as $\{\boldsymbol{G}_h^{001} \cup \boldsymbol{G}_h^{100} \cup \boldsymbol{G}_h^{010} \cup \boldsymbol{G}_h^{111}\}$ in $3d$. Abbreviating $A(\boldsymbol{\theta}_{\boldsymbol{\alpha}}, h; \omega)$ as $A_{\boldsymbol{\alpha}}$, the partial sweep over the red points gives :

$$
S_h^R \varphi_{\boldsymbol{\alpha}} = \begin{cases} A_{\boldsymbol{\alpha}}\varphi_{\boldsymbol{\alpha}} & \text{for } \boldsymbol{x} \in \boldsymbol{G}_h^{red} \\ \varphi_{\boldsymbol{\alpha}} & \text{for } \boldsymbol{x} \in \boldsymbol{G}_h^{blk} \end{cases}
\tag{3.15}
$$

So writing out the linear combinations in Equation 3.13 for all $\boldsymbol{\alpha} = (i, j, k)$ (abbreviating $\boldsymbol{\alpha} = ijk$):

$$
S_h^R \varphi_{\boldsymbol{\alpha}} = \begin{cases}
A_{000}(\psi_1 + \psi_4 + \psi_6 + \psi_7) + \psi_2 + \psi_3 + \psi_5 + \psi_8; & (\boldsymbol{\alpha} = 000) \\
A_{001}(\psi_1 - \psi_4 - \psi_6 + \psi_7) - \psi_2 + \psi_3 + \psi_5 - \psi_8; & (\boldsymbol{\alpha} = 001) \\
A_{010}(\psi_1 - \psi_4 + \psi_6 - \psi_7) + \psi_2 - \psi_3 + \psi_5 - \psi_8; & (\boldsymbol{\alpha} = 010) \\
A_{011}(\psi_1 + \psi_4 - \psi_6 - \psi_7) - \psi_2 - \psi_3 + \psi_5 + \psi_8; & (\boldsymbol{\alpha} = 011) \\
A_{100}(\psi_1 + \psi_4 - \psi_6 - \psi_7) + \psi_2 + \psi_3 - \psi_5 - \psi_8; & (\boldsymbol{\alpha} = 100) \\
A_{101}(\psi_1 - \psi_4 + \psi_6 - \psi_7) - \psi_2 + \psi_3 - \psi_5 + \psi_8; & (\boldsymbol{\alpha} = 101) \\
A_{110}(\psi_1 - \psi_4 - \psi_6 + \psi_7) + \psi_2 - \psi_3 - \psi_5 + \psi_8; & (\boldsymbol{\alpha} = 110) \\
A_{111}(\psi_1 + \psi_4 + \psi_6 + \psi_7) - \psi_2 - \psi_3 - \psi_5 - \psi_8; & (\boldsymbol{\alpha} = 111)
\end{cases}
\tag{3.16}
$$

Substituting ψ_i from Equations 3.12, we get:

$$S_h^R \varphi_{\boldsymbol{\alpha}} = \frac{1}{2} \begin{cases} (A_{000}+1)\varphi_{000} + (A_{000}-1)\varphi_{111} \\ (A_{001}+1)\varphi_{001} + (A_{001}-1)\varphi_{110} \\ (A_{010}+1)\varphi_{010} + (A_{010}-1)\varphi_{101} \\ (A_{011}+1)\varphi_{011} + (A_{011}-1)\varphi_{100} \\ (A_{100}-1)\varphi_{011} + (A_{100}+1)\varphi_{100} \\ (A_{101}-1)\varphi_{010} + (A_{101}+1)\varphi_{101} \\ (A_{110}-1)\varphi_{001} + (A_{110}+1)\varphi_{110} \\ (A_{111}-1)\varphi_{000} + (A_{111}+1)\varphi_{111} \end{cases} \tag{3.17}$$

Let $w_h(\boldsymbol{x}) \in E_h^\theta$ be any arbitrary grid function, then it can be represented as a linear combination of the basis elements $\varphi_{\boldsymbol{\alpha}}$, i.e.

$$w_h(\boldsymbol{x}) = a_{000}\varphi_{000} + a_{001}\varphi_{001} + a_{010}\varphi_{010} + a_{011}\varphi_{011} + a_{100}\varphi_{100} + a_{101}\varphi_{101} + a_{110}\varphi_{110} + a_{111}\varphi_{111} \tag{3.18}$$

then

$$\begin{aligned} &S_h^R(\, a_{000}\varphi_{000} + a_{001}\varphi_{001} + a_{010}\varphi_{010} + a_{011}\varphi_{011} + a_{100}\varphi_{100} + a_{101}\varphi_{101} + a_{110}\varphi_{110} + a_{111}\varphi_{111}) \\ &\quad = \widetilde{a}_{000}\varphi_{000} + \widetilde{a}_{001}\varphi_{001} + \widetilde{a}_{010}\varphi_{010} + \widetilde{a}_{011}\varphi_{011} + \widetilde{a}_{100}\varphi_{100} + \widetilde{a}_{101}\varphi_{101} + \widetilde{a}_{110}\varphi_{110} + \widetilde{a}_{111}\varphi_{111}. \end{aligned}$$

Thus, we have the relation:

$$\widetilde{S}_h^R a_{\boldsymbol{\alpha}} = \widetilde{a}_{\boldsymbol{\alpha}} \quad (\because \quad S_h^R : E_h^\theta \to E_h^\theta; \;\; \boldsymbol{\theta} \in T^{low}) \tag{3.19}$$

with the *representation matrix* $\widetilde{S}_h^R$, given as:

$$\widetilde{S}_h^R = \frac{1}{2} \begin{bmatrix} A_{000}+1 & & & & & & & A_{111}-1 \\ & A_{001}+1 & & & & & A_{110}-1 & \\ & & A_{010}+1 & & & A_{101}-1 & & \\ & & & A_{011}+1 & A_{100}-1 & & & \\ & & & A_{011}-1 & A_{100}+1 & & & \\ & & A_{010}-1 & & & A_{101}+1 & & \\ & A_{001}-1 & & & & & A_{110}+1 & \\ A_{000}-1 & & & & & & & A_{111}+1 \end{bmatrix} \tag{3.20}$$

This matrix converts to the conventional block matrix through a restructuring by a permutation similarity transform. Note that similarity transforms with permutation matrices are always eigenvalue-invariant. This leads us to:

$$\widetilde{S'}_h^R = \frac{1}{2} \begin{bmatrix} A_{000}+1 & A_{111}-1 & & & & & & \\ A_{000}-1 & A_{111}+1 & & & & & & \\ & & A_{001}+1 & A_{110}-1 & & & & \\ & & A_{001}-1 & A_{110}+1 & & & & \\ & & & & A_{010}+1 & A_{101}-1 & & \\ & & & & A_{010}-1 & A_{101}+1 & & \\ & & & & & & A_{011}+1 & A_{100}-1 \\ & & & & & & A_{011}-1 & A_{100}+1 \end{bmatrix} \tag{3.21}$$

with $\widetilde{S'}_h^R = P\,\widetilde{S}_h^R\,P^T$, where P is an appropriate permutation matrix. Likewise wading our way through similar derivations we obtain $\widetilde{S'}_h^B$ as:

$$\frac{1}{2}\begin{bmatrix} A_{000}+1 & -A_{111}+1 & & & & & & & & & & & & & & \\ -A_{000}+1 & A_{111}+1 & & & & & & & & & & & & & & \\ & & A_{001}+1 & -A_{110}+1 & & & & & & & & & & & & \\ & & -A_{001}+1 & A_{110}+1 & & & & & & & & & & & & \\ & & & & A_{010}+1 & -A_{101}+1 & & & & & & & & & & \\ & & & & -A_{010}+1 & A_{101}+1 & & & & & & & & & & \\ & & & & & & A_{011}+1 & -A_{100}+1 & & & & & & & & \\ & & & & & & -A_{011}+1 & A_{100}+1 & & & & & & & & \end{bmatrix} \tag{3.22}$$

An extension to d-dimensions is now clear. Note that ω-RB Jacobi couples those Fourier components that alias on the coarse grid. Motivated by the eigenvalue invariance of this transformation, and without any loss of clarity, we now use the notation $\widetilde{S}_h^R$ and $\widetilde{S}_h^R$ for the representation matrices $\widetilde{S'}_h^R$ and $\widetilde{S'}_h^B$ given by Equations 3.21 and 3.22.

Thus the Fourier representations of the half sweep operators which stand for the smoothing steps over the red points (R) and the black points (B) w.r.t. the two-dimensional minimal invariant subspaces [30] are represented by 2^{d-1} blocks that read:

$$\widetilde{S}_h^R(\boldsymbol{\theta},\omega) = \frac{1}{2}\begin{pmatrix} A(\boldsymbol{\theta},\omega)+1 & A(\boldsymbol{\theta}',\omega)-1 \\ A(\boldsymbol{\theta},\omega)-1 & A(\boldsymbol{\theta}',\omega)+1 \end{pmatrix}$$

$$\text{and} \qquad \widetilde{S}_h^B(\boldsymbol{\theta},\omega) = \frac{1}{2}\begin{pmatrix} A(\boldsymbol{\theta},\omega)+1 & -A(\boldsymbol{\theta}',\omega)+1 \\ -A(\boldsymbol{\theta},\omega)+1 & A(\boldsymbol{\theta}',\omega)+1 \end{pmatrix}$$

$$\text{with} \quad A(\boldsymbol{\theta},\omega) = A^{2o}(\boldsymbol{\theta},\omega), A^{4o}(\boldsymbol{\theta},\omega) \quad \text{and} \quad A(\boldsymbol{\theta}',\omega) = A^{2o}(\boldsymbol{\theta}',\omega), A^{4o}(\boldsymbol{\theta}',\omega).$$

The amplification matrix $\widetilde{S}_h(\boldsymbol{\theta},\omega)$ is then given by:

$$\widetilde{S}_h(\boldsymbol{\theta},\omega) = \widetilde{S}_h^B(\boldsymbol{\theta},\omega)\,\widetilde{S}_h^R(\boldsymbol{\theta},\omega). \tag{3.23}$$

which is the d-dimensional Fourier representation of the ω-RB Jacobi operator, based on ($h \longrightarrow 2h$) transfers.

3.4 The Red-Black Smoothing Factor ω

As a first step towards defining the smoothing factor for ω-RB Jacobi we recall that since pattern relaxations intermix the high and the low frequencies, we need to filter out the low ones. For the purpose of filtering out the low frequencies we replace the real coarse-grid correction operator by an ideal coarse grid correction operator (a projection operator onto the space of high frequencies) Q_h^H as:

$$Q_h^H\varphi_h(\boldsymbol{\theta},\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{\theta} \in T^{low} \\ \varphi_h(\boldsymbol{\theta},\boldsymbol{x}) & \text{if } \boldsymbol{\theta} \in T^{high} \end{cases} \tag{3.24}$$

We now define the smoothing factor as follows:

Definition 3.4.1. Smoothing factor *The smoothing factor μ of the ω-RB Jacobi operator is defined as the worst factor by which the high frequency errors are reduced per iteration step. So with ν denoting the number of relaxation sweeps, and ρ the matrix spectral radius, we have:*

$$\mu(\omega, \nu) := \ sup \left\{ \sqrt[\nu]{\rho\left(\widetilde{Q}_h^H \widetilde{S}_h^{\nu}\right)} \right\} \quad \boldsymbol{\theta} \in T^{low} \tag{3.25}$$

or equivalently:

$$\mu(\omega, \nu) := \ sup \left\{ \sqrt[\nu]{\rho\left(\widetilde{Q}_h^H (\widetilde{S}_h^R \widetilde{S}_h^B)^{\nu}\right)} \right\} \quad \boldsymbol{\theta} \in T^{low} \tag{3.26}$$

3.5 Red-Black LFA Extension for Quadrupling in d-dimensions

3.5.1 Fourier High/Low frequencies for quadrupling

The Fourier smoothing analysis based on full and partial coarsenings that consist both of doubling ($h \to 2h$) and quadrupling ($h \to 4h$) -or only quadrupling- (as employed in Chapter 4) is built upon a different set of indices. Now the frequencies and the corresponding index-set would be defined as follows:

$$T^{low} = \left\{ \boldsymbol{\theta} : \boldsymbol{\theta} \in \left[-\frac{\pi}{4}, \frac{\pi}{4} \right)^d \right\}$$
$$T^{high} = \left\{ \boldsymbol{\theta} : \boldsymbol{\theta} \in [-\pi, \pi)^d \setminus \left[-\frac{\pi}{4}, \frac{\pi}{4} \right)^d \right\}$$

with the space of harmonics E_h^{θ} as:

$$E_h^{\theta} = \left\{ \varphi_h(\boldsymbol{\theta}_{\boldsymbol{\alpha}}, \boldsymbol{x}) : \boldsymbol{\alpha} = (\alpha^1, \alpha^2, \cdots, \alpha^d), \quad \alpha^j \in \{0, -1/2, 1/2, 1\} \right\} \tag{3.27}$$

This Fourier space is 4^d-dimensional. Regardless of the order of the basis elements, an appropriate Fourier representation of the relaxation operator would consist of 2^{d+1} (2×2) blocks. Each block couples Fourier components that alias on the coarse grid. The following example elaborates this extension:

Example 3.5.1. The $8\ (2\times2)$ **blocks for quadrupling in** $2d$ *The order of occurrence of these blocks in the representation is not important. In the case of quadrupling, the frequency component having the index $-1/2$ aliases with $1/2$, and the component with the index 0 aliases with 1 on the coarse grid. The possible index pairs in $2d$, i.e., $\boldsymbol{\alpha} = (\alpha^1, \alpha^2)$ may be enumerated as:*

$$\begin{aligned}\boldsymbol{\alpha} &= \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_{16}\} \\ &= \{(0,0), (0,-1/2), (0,1/2), (0,1), (-1/2,0), (-1/2,-1/2), (-1/2,1/2), (-1/2,1), (1/2,0), \\ &\quad (1/2,-1/2), (1/2,1/2), (1/2,1), (1,0), (1,-1/2), (1,1/2), (1,1)\}\end{aligned}$$

We get the 8 (2×2) *block matrices that turn up in* $\widetilde{S}_h^R$ *as:*

$$\begin{bmatrix} A^{\alpha_1+1} & A^{\alpha_{16}-1} \\ A^{\alpha_1-1} & A^{\alpha_{16}+1} \end{bmatrix}, \begin{bmatrix} A^{\alpha_2+1} & A^{\alpha_{15}-1} \\ A^{\alpha_2-1} & A^{\alpha_{15}+1} \end{bmatrix}, \begin{bmatrix} A^{\alpha_3+1} & A^{\alpha_{14}-1} \\ A^{\alpha_3-1} & A^{\alpha_{14}+1} \end{bmatrix}, \begin{bmatrix} A^{\alpha_4+1} & A^{\alpha_{13}-1} \\ A^{\alpha_4-1} & A^{\alpha_{13}+1} \end{bmatrix},$$
$$\begin{bmatrix} A^{\alpha_5+1} & A^{\alpha_{12}-1} \\ A^{\alpha_5-1} & A^{\alpha_{12}+1} \end{bmatrix}, \begin{bmatrix} A^{\alpha_6+1} & A^{\alpha_{11}-1} \\ A^{\alpha_6-1} & A^{\alpha_{11}+1} \end{bmatrix}, \begin{bmatrix} A^{\alpha_7+1} & A^{\alpha_{10}-1} \\ A^{\alpha_7-1} & A^{\alpha_{10}+1} \end{bmatrix}, \begin{bmatrix} A^{\alpha_8+1} & A^{\alpha_9-1} \\ A^{\alpha_8-1} & A^{\alpha_9+1} \end{bmatrix}$$

and likewise for $\widetilde{S}_h^B$*. An extension for constructing the half sweep amplification matrices in d-dimensions is therefore straight forward.*

The definition of the projection operator Q_h^H and the smoothing factor μ remains unchanged. Although comprehending the details is simple enough, the actual computation of optimal Red-Black relaxation parameters and corresponding convergence factors (for a case involving quadrupling) can be inefficient due to the huge number of blocks involved in d-dimensions. There is however, a neat simplification which allows a somewhat more efficient evaluation. This is illustrated in the following section.

3.5.2 Alternate Visualization of the Invariant Subspaces

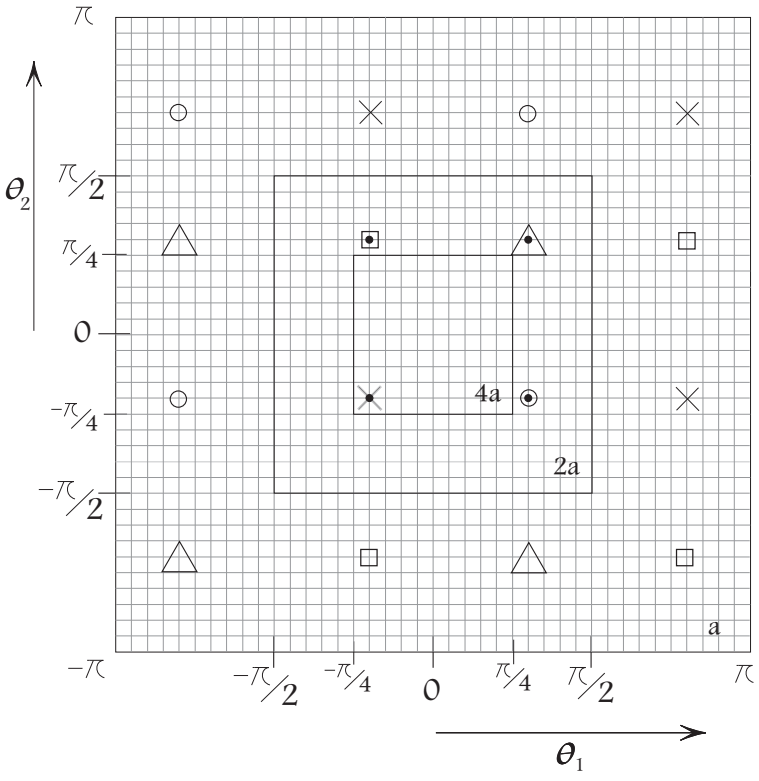


Figure 3.1: Aliasing frequencies in the frequency spaces a, 2a and 4a. Frequency indicators are •, ×, ○, △, □. Superimposed indicators show aliasing frequencies.

Example 3.5.1 shows the complete set of multi-indices α_i. The actual aliasing frequency pairs, can be obtained in this case by using Equation 3.28 for frequencies occurring in T^{low}.

$$\begin{pmatrix}\theta_1^{high}\\ \theta_2^{high}\end{pmatrix} = \begin{pmatrix}\theta_1^{low}\\ \theta_2^{low}\end{pmatrix} - \begin{pmatrix}\alpha^i \text{sign}\,(\theta_1^{low})\\ \alpha^j \text{sign}\,(\theta_2^{low})\end{pmatrix}\frac{\pi}{4} \tag{3.28}$$

An alternate formulation is also possible, considering only the old multi-index set where $\alpha_i \in \{0, 1\}$. To realize this, we slightly adjust our nomenclature to incorporate frequencies existing in areas a, 2a, and 4a, as shown in Figure 3.1.

We use θ^a, θ^{2a}, and θ^{4a} to denote frequencies existing in the areas demarked by a, 2a, and 4a, respectively, in Figure 3.1. Clearly, only θ^{4a} belongs to T^{low}. Nevertheless, the following relationships hold:

$$\begin{pmatrix}\theta_1^{2a}\\ \theta_2^{2a}\end{pmatrix} = \begin{pmatrix}\theta_1^{4a}\\ \theta_2^{4a}\end{pmatrix} - \begin{pmatrix}\alpha^i \text{sign}\,(\theta_1^{4a})\\ \alpha^j \text{sign}\,(\theta_2^{4a})\end{pmatrix}\frac{\pi}{2} \tag{3.29}$$

$$\begin{pmatrix}\theta_1^{a}\\ \theta_2^{a}\end{pmatrix} = \begin{pmatrix}\theta_1^{2a}\\ \theta_2^{2a}\end{pmatrix} - \begin{pmatrix}\alpha^i \text{sign}\,(\theta_1^{2a})\\ \alpha^j \text{sign}\,(\theta_2^{2a})\end{pmatrix}\pi \tag{3.30}$$

These relationships can be exploited to construct an efficient implementation of Fourier analysis software. The following example illustrates the process.

Example 3.5.2. Optimization of only 2^d minimal blocks *Consider a low frequency $\theta^{4a} = (-\pi/5, -\pi/5) \in T^{low}$. Using Equation 3.29, and the multi-index α, obtain 3 (high) frequencies in 2a. Enumerated with respect to α, these 4 frequencies are:*

$$\theta^{4a} = \theta_{00}^{2a} = (-\frac{\pi}{5}, -\frac{\pi}{5}), \quad \theta_{01}^{2a} = (-\frac{\pi}{5}, \frac{3\pi}{10}), \tag{3.31}$$

$$\theta_{10}^{2a} = (\frac{3\pi}{10}, -\frac{\pi}{5}), \quad \theta_{11}^{2a} = (\frac{3\pi}{10}, \frac{3\pi}{10}) \tag{3.32}$$

Next, for each of these 4 frequencies, form 3 more by using Equation 3.30. This leads to the 2d Red-Black block matrices. For an example case of frequency θ_{00}^{2a}, this gives $\left(\theta_{00}^{2a}\right)_{00}^{a}$, $\left(\theta_{00}^{2a}\right)_{01}^{a}$, $\left(\theta_{10}^{2a}\right)_{00}^{a}$, and $\left(\theta_{00}^{2a}\right)_{11}^{a}$. For clarity, θ_{00}^{2a} is now renamed as $\left(\theta_{00}^{2a}\right)_{00}^{a}$,

and leads to the determination of the spectral radius of the matrix $\widetilde{Q}_1\widetilde{S}_1$:

$$\widetilde{S}_1^R = \frac{1}{2}\begin{bmatrix} \left(A_{00}^{2a}\right)_{00}^a + 1 & \left(A_{00}^{2a}\right)_{11}^a - 1 & & \\ \left(A_{00}^{2a}\right)_{00}^a - 1 & \left(A_{00}^{2a}\right)_{11}^a + 1 & & \\ & & \left(A_{00}^{2a}\right)_{01}^a + 1 & \left(A_{00}^{2a}\right)_{10}^a - 1 \\ & & \left(A_{00}^{2a}\right)_{01}^a - 1 & \left(A_{00}^{2a}\right)_{10}^a + 1 \end{bmatrix} \tag{3.33}$$

$$\widetilde{S}_1^B = \frac{1}{2}\begin{bmatrix} \left(A_{00}^{2a}\right)_{00}^a + 1 & -\left(A_{00}^{2a}\right)_{11}^a + 1 & & \\ -\left(A_{00}^{2a}\right)_{00}^a + 1 & \left(A_{00}^{2a}\right)_{11}^a + 1 & & \\ & & \left(A_{00}^{2a}\right)_{01}^a + 1 & -\left(A_{00}^{2a}\right)_{10}^a + 1 \\ & & -\left(A_{00}^{2a}\right)_{01}^a + 1 & \left(A_{00}^{2a}\right)_{10}^a + 1 \end{bmatrix} \tag{3.34}$$

$$\widetilde{Q}_1 = [0, 1, 1, 1]^T$$
$$\widetilde{S}_1 = \widetilde{S}_1^R\widetilde{S}_1^B$$

This gives 1 spectral radius. Treating the other 3 frequencies, given in Equation set 3.31, in the same way, results in the other 3. As the context is of full quadrupling (for ease of discussion), this implies that the other three projection operators $\widetilde{Q}^2$, $\widetilde{Q}^3$, and $\widetilde{Q}^4$ will only contain ones. The maximum of these 4 spectral radii is the Red-Black convergence factor for quadrupling. This process is equivalent to the one illustrated in Example 3.5.2 and appears to be more efficient in d-dimensions.

This completes the discussion of Fourier analysis of Red-Black Jacobi.

Remark 3.5.1. Optimal smoothing parameters *The projection operator Q_h^H plays a very practical and important role in the evaluation of the optimal relaxation parameters ω. We elaborate its use in Chapter 4, and obtain ω- through the analysis presented in this chapter, for different kinds of partial and full, doubling and quadrupling transfers.*

Chapter 4

Multigrid for Multidimensional Elliptic PDEs

In the last chapter we presented the local Fourier analysis in d-dimensions, which sets the ground for the multigrid method and techniques presented in this chapter. The main focus here is on multigrid convergence for multidimensional partial differential equations (PDEs) on non-equidistant grids, such as may be encountered in a sparse grid solution. As a model problem we have chosen the anisotropic stationary diffusion equation, on a rectangular hypercube. We develop techniques for building the general d-dimensional adaptations of the multigrid components. We propose and develop partial grid coarsening strategies to handle anisotropies that are induced due to discretization on a non-equidistant grid. These techniques incorporate both $h \to 2h$, as well as $h \to 4h$ grid transfers. Apart from the d-dimensional formulae and techniques, we compute the optimal relaxation parameters -through LFA presented in the last chapter- of point ω-Red-Black Jacobi method for a general multidimensional case. These parameters are presented for the finite difference $O(h^2)$ and the $O(h^4)$ long-stencil operators. Numerical experiments based on the techniques proposed in this chapter are included.

4.1 Applications and Solution Techniques

Multidimensional partial differential equations have a diverse application in various fields, which amongst others, include financial engineering [31], molecular biology [32], and quantum dynamics [33, 34]. The existing literature on the multigrid treatment of various problems, rarely explores issues that arise out of growth in the dimensionality of the problem. The implications of dimensionality growth include deterioration of the multigrid convergence rate, impractical storage requirements and huge amounts of the CPU-time for single grid solution methods. Our main emphasis here is on the first challenge. We abbreviate *multigrid for d-dimensional PDEs* as *d-multigrid*.

A multigrid treatment of multidimensional PDEs based on hyperplane relaxation

has been proposed by Reisinger in [31]. We present a multigrid treatment based on point relaxation and partial coarsening schemes. We demonstrate how the multigrid convergence factor can be brought down for higher d by the suggested grid coarsening strategies and a proper choice of the relaxation parameters in the smoothing process.

The strategy that we suggest in this chapter for good multigrid convergence is that keeping the point smoothing method, we coarsen the grid simultaneously along all the dimensions where the errors are strongly coupled. We call this strategy *simultaneous partial coarsening*. This way we relax the anisotropy at each successive grid level, till the problem is isotropic to the point where full coarsening is feasible. Full coarsening from this stage onwards, brings the grid to the coarsest possible level, where we solve exactly. These strategies therefore consist of two main phases, the *partial coarsening phase* and the *full coarsening phase*. We show that multigrid based on *quadrupling* ($h \rightarrow 4h$) transfers during the partial coarsening phase of the scheme, renders a very efficient multigrid method if optimal relaxation parameters are used in the point smoothing process. We would like to point out that partial coarsening schemes have also been put forward by Larsson et al, identifying the conditions for partial coarsening through LFA [28]. Moreover, the utility of adaptive grid coarsening in multigrid has also been demonstrated by Elias et al and Horton and Vandewalle in [35] and [36], respectively.

We begin with the discretization of a d-dimensional PDE, and show in Section 4.2, how this might be done with Kronecker tensor products. Section 4.3 describes d-multigrid components that we use, i.e. point smoothing, transfer operators, coarsening strategies etc., and concludes with a computational complexity analysis. Next, in Section 4.4, we explain how optimal relaxation parameters for ω-RB Jacobi might be obtained efficiently through the smoothing analysis done in Chapter 5. This section highlights the role of the projection operator Q_h^H in general cases based on partial and full doubling and quadrupling transfers and concludes with a tabular presentation (up to 6d) of some optimal relaxation parameters. Finally, in Section 4.5, we present quite a few numerical experiments and demonstrate the fine multigrid convergence that we achieve.

4.2 The Discretization

We first recall that discrete operators can be implemented in two different ways. One is the stencil method and the other is the matrix method. The stencil method saves storage but is inherently difficult to implement due to the visual constraints due to the multidimensionality of the problem. Therefore, to circumvent the complicated implementation issues we use matrices (in sparse storage formats) and here we present matrix generation formulae based on Kronecker-tensor-products.

4.2.1 The Continuous Problem and FDM Discretization

For analysis and experimentation we choose the d-dimensional stationary diffusion equation, with Dirichlet boundary conditions, to serve as our model problem. As in Chapter 2, $\boldsymbol{x}$ is a d-tuple, so that $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$, and $\{a_i, b_i, \varepsilon_i\} \in \mathbb{R}$ with

$\varepsilon_i \in (0, \infty)$. The continuous problem then reads:

$$\begin{aligned} -Lu(\boldsymbol{x}) = -\sum_{i=1}^{d} \varepsilon_i \frac{\partial^2}{\partial x_i^2} u(\boldsymbol{x}) = f^{\Omega}(\boldsymbol{x}), \qquad & \boldsymbol{x} \in \Omega = \prod_{i=1}^{d} (a_i, b_i) \subset \mathbb{R}^d \\ u(\boldsymbol{x}) = f^{\Gamma}(\boldsymbol{x}), \qquad & \boldsymbol{x} \in \partial\Omega \end{aligned} \tag{4.1}$$

Subsequently, the discrete counterpart reads

$$\begin{aligned} -L_h u_h(\boldsymbol{x}) = f_h^{\Omega}(\boldsymbol{x}), \qquad & \boldsymbol{x} \in \Omega_h \\ u_h(\boldsymbol{x}) = f_h^{\Gamma}(\boldsymbol{x}), \qquad & \boldsymbol{x} \in \Gamma_h = \partial\Omega_h \end{aligned} \tag{4.2}$$

The finite difference discretization of the continuous operator L_h is chosen to be either $O(h^2)$ accurate, giving a $(2d+1)$-point stencil, or else $O(h^4)$ accurate, with a $(4d+1)$-point long-stencil for all *interior* points. The number of cells in the discretization grid along the i^{th} dimension -represented by N_i- *need not* be equal to the number of cells along (say) the j^{th} dimension. So with $h_i = (b_i - a_i)/N_i$ -the mesh size along the i^{th} dimension- the $1d$ variant of these multidimensional stencils are

$$(L_h)_i = \left(\frac{\partial^2}{\partial x_i^2}\right)_h \triangleq \frac{1}{h_i^2}[1 \quad \underline{-2} \quad 1] + O(h^2), \tag{4.3}$$

$$(L_h)_i = \left(\frac{\partial^2}{\partial x_i^2}\right)_h \triangleq \frac{1}{12h_i^2}[-1 \quad 16 \quad \underline{-30} \quad 16 \quad -1] + O(h^4). \tag{4.4}$$

It has to be noted that the $O(h^4)$ long stencil uses the so-called *ghost-points* (points outside the discretization grid) when applied to points *near* the boundary. To alleviate this problem we have the option of employing a different stencil, having shorter connections at the boundary. Thus we can either employ the simple $O(h^2)$ operator at the boundaries, or else use a different scheme with one-sided differencing. In this work, we use the second order stencil, for points near the boundaries. The discretization given by Equation 4.2 leads to the matrix equation $A_h u_h = b_h$ as indicated by Equation 2.5.

4.2.2 The Matrix A_h in Kronecker Tensor Products

We choose the so-called *eliminated boundary* scheme. In this scheme Dirichlet corrections from the domain boundaries are incorporated in the right hand side b_h and are therefore eliminated from the matrix A_h. This scheme results in a total of M unknowns, -$(M \times M)$ being the order of the discretization matrix A_h- with:

$$M = \prod_{i=1}^{d} (N_i - 1).$$

We represent the discretization grid size by $\boldsymbol{N}$, so that:

$$\boldsymbol{N} = [N_1, N_2, \cdots, N_d]. \tag{4.5}$$

The discrete matrix A_h can be built by the following tensor product formula:

$$A_h = \sum_{i=1}^{d} \left\{ \bigotimes_{j=i}^{d-1} I_{(d+i-j)} \otimes (\overline{L}_h)_i \otimes \bigotimes_{j=1}^{i-1} I_{(i-j)} \right\}. \tag{4.6}$$

$\otimes$ is the Kronecker-tensor-product of matrices. $\bigotimes$ is the cumulative Kronecker-tensor-product. For example:

$$\bigotimes_{i=1}^{3} P_i = P_1 \otimes P_2 \otimes P_3.$$

Kronecker-tensor-products are non-commutative and associative operations (see [37]). The order is determined by the subscripts here and the associative hierarchy does not matter.

In Equation 4.6 I_i, $i \in \{1, 2, \cdots, d\}$, is the identity matrix of order $(N_i - 1)$ and $(\overline{L}_h)_i$ is the $1d$ discrete Laplacian matrix, constructed through Equations 4.3 or 4.4 as illustrated by the following example. Suppose that $\boldsymbol{N} = [8, 6]$ is the prescribed grid size for a certain $2d$ problem, then we construct $(\overline{L}_h)_i$ by writing down the discrete stencil in Equation 4.4 for each point, including the boundaries. Then we isolate the left and the right boundary vectors (as shown below) and incorporate them in the right hand side b_h. For example $(\overline{L}_h)_1$ is the following matrix (without the isolated columns on the left and the right) according to the choice $O(h^4)$ of the computational accuracy:

$$\frac{\varepsilon_1}{12h_1^2} \begin{bmatrix} & & & & & & & & \\ 12 & -24 & 12 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 16 & -30 & 16 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 16 & -30 & 16 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 16 & -30 & 16 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 16 & -30 & 16 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 16 & -30 & 16 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12 & -24 & 12 \\ & \underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\mathbf{L}_1} & & & & & & & \end{bmatrix}.$$

$1d$ discrete operator matrices constructed in this way for each grid dimension, are to be substituted in Equation 4.6 for building up the discrete d-dimensional operator matrix A_h. (The $O(h^2)$-discretization is handled similarly.)

4.2.3 The Right-Hand-Side $\mathbf{b}_h$

The right hand side b_h consists of the source function f_h^{Ω} and the boundary function f_h^{Γ}. It is important to define a consistent enumeration of grid points in high dimensions. The grid index of an unknown is represented by a d-tuple $\boldsymbol{j} = (j_d, j_{d-1}, \cdots, j_1)$, where j_i runs in the range $1, 2, \cdots, N_i$. The complete set of all possible grid point indices can thus be modelled by a rectangular array $\boldsymbol{J}$ having d columns, and M rows. Each row can be considered to represent a particular grid index $\boldsymbol{j}$. In our scheme, the order in which the rows of this index array $\boldsymbol{J}$ are laid out, is the natural d-dimensional extension

of what we commonly know as the lexicographic ordering in 2 or 3 dimensions. The index array $\boldsymbol{J}$ can be represented as:

$$\boldsymbol{J} = [J_d \;\; J_{(d-1)} \cdots \;\; J_i \;\; \cdots \;\; J_2 \;\; J_1], \tag{4.7}$$

where each J_i is a column vector of length M. Also consider the following definitions which we require to build the index-set for the interior and the boundary points, Let;

$$\begin{aligned} \eta_i &= [1, 2, \cdots, (N_i - 1)]^T, \qquad i = 1, 2, \cdots, d \quad \text{(see Equation 4.5)}, \\ 1_i &= \underbrace{[1, 1, \cdots, 1]}_{\text{Total } N_i}^T \qquad\qquad T \text{ stands for transpose} \end{aligned} \tag{4.8}$$

We now formulate the columns of $\boldsymbol{J}$ as follows:

$$J_i = \bigotimes_{j=i}^{d-1} 1_{(d+i-j)} \otimes \eta_i \otimes \bigotimes_{j=1}^{i-1} 1_{(i-j)}. \tag{4.9}$$

At this stage the vector of source function values can be computed as the development of the index-set $\boldsymbol{J}$ is complete. Thus, computing the source function for each grid coordinate (row of the array $\boldsymbol{J}$), and denoting it by $\mathcal{S}$, we have $\mathcal{S} = f^{\Omega}_{\boldsymbol{J}}$.

Now for computing the contribution of boundaries in b_h, recall that we isolated two column vectors, namely the left and the right boundary coefficient vectors from the $1d$ discrete operators in each dimension. Considering the case of the i^{th} dimension if we denote these by l_i and r_i respectively, then we can define the i^{th} d-dimensional left and right boundary coefficient vectors, viz, $\mathcal{L}_i$ and $\mathcal{R}_i$ as follows:

$$\begin{aligned} \mathcal{L}_i &= \bigotimes_{j=i}^{d-1} 1_{(d+i-j)} \otimes l_i \otimes \bigotimes_{j=1}^{i-1} 1_{(i-j)}, \\ \mathcal{R}_i &= \bigotimes_{j=i}^{d-1} 1_{(d+i-j)} \otimes r_i \otimes \bigotimes_{j=1}^{i-1} 1_{(i-j)}. \end{aligned} \tag{4.10}$$

The contribution of the boundary-values in b_h has two parts, i.e., values from the left boundary and values from the right boundary. We denote the two by $\mathcal{B}_L$ and $\mathcal{B}_R$, respectively. $\mathcal{B}_L$ is the cumulative sum of the d left boundaries and likewise for the right. At this point a word about the boundary index set is just in order. If in Equation 4.7 any J_i is replaced by a column vector consisting of *the left Dirichlet boundary value,* a_i, we get a left boundary index set and if we replace it by a vector of *the right Dirichlet boundary value,* b_i, we get a right boundary index set. If the left and the right boundary index sets are given as:

$$\begin{aligned} \boldsymbol{J}_{L_i} &= [J_d \;\; J_{(d-1)} \cdots J_{(i-1)} \;\; \overline{a_i} \;\; J_{(i+1)} \cdots J_2 \;\; J_1], \\ \boldsymbol{J}_{R_i} &= [J_d \;\; J_{(d-1)} \cdots J_{(i-1)} \;\; \overline{b_i} \;\; J_{(i+1)} \cdots J_2 \;\; J_1], \end{aligned}$$

then

$$\mathcal{B}_L = \sum_{i=1}^{d} (\mathcal{L}_i \diamond f^{\Gamma}_{J_{L_i}}), \tag{4.11}$$
$$\mathcal{B}_R = \sum_{i=1}^{d} (\mathcal{R}_i \diamond f^{\Gamma}_{J_{R_i}}).$$

$\diamond$ represents component-wise multiplication of the operand column vectors.
Thus we have the right hand side b_h as

$$b_h = \mathcal{S} + \mathcal{B}_L + \mathcal{B}_R \tag{4.12}$$

and the discretization is complete.

4.3 *d*-Multigrid Based on Point Smoothing

The core of this chapter is the coarsening strategies proposed here which are based on a mixture of doubling ($h \rightarrow 2h$) and quadrupling ($h \rightarrow 4h$) grid transfers and which -coupled with point based relaxation- yield very efficient multigrid methods for problems on non-equidistant grids. The aim is to close upon a robust method that applies for general grid-aligned anisotropies in d dimensions.

The essential components of d-multigrid are the smoothing method and the coarse grid correction. For anisotropic problems it is a choice to keep the point smoothing method and to coarsen only along a sub-set of the dimensions, precisely those that are strongly coupled. This ensures that coarsening takes place only where the errors are smooth. For *nearly* isotropic problems the best strategy is to combine the point smoothing method with doubling based full coarsening and to use the optimal relaxation parameters obtained for d dimensions.

Remark 4.3.1. (Multigrid algorithm in the literature) *Multigrid algorithm as presented in Chapter 2 does not change for the higher-dimensional case, however, the components have to be generalized to match this new situation. General multigrid algorithms with slight variations are also presented in the literature, notably in [2, 6, 8, 9, 38, 39].*

4.3.1 The Relaxation Method

Of the many available point smoothing based relaxation methods, we choose the ω-Red-Black Jacobi method due to its excellent smoothing effect for problems of the Poisson type as described in Chapter 2, Section 2.3.1, Description 2.3.5. From an implementational point of view, this Red-Black smoothing procedure which is based on partial steps depends upon a partitioning process by which the index-set (represented by Equation 4.7) of the interior grid-points, can be dissected into the *red* part ($\boldsymbol{J}_r$), and the *black* part ($\boldsymbol{J}_b$). The grid point enumeration that we employ in our implementation scheme is such that the points are arranged linearly (in a column vector), counted out

in the lexicographical order for a d-dimensional grid. The unequal number of cells along different dimensions of the grid makes this partitioning process somewhat non-trivial. In Appendix C we present a way to bring about this segregation of odd and even points from a purely implementational aspect.

We also employ optimal relaxation parameters ω_{opt} in the relaxation process. It is well known that $\omega = 1$ serves as a good choice for the $2d$ isotropic case. In the case of anisotropy and higher dimensions the error smoothing effect of the relaxation method can be enhanced by the use of optimal relaxation parameters [30]. This implies that a search for ω_{opt} pays off. We employ d-dimensional Local Fourier Smoothing Analysis for this purpose, see Section 4.4.

4.3.2 Coarsening Strategies to Handle Anisotropies

We present two *grid-adaptive* coarsening procedures as our test cases, both of which have shown excellent convergence results. We call them *Strategy-1* and *Strategy-2*. Both the strategies are similar in the sense that they consist of two distinct coarsening phases, the partial coarsening phase and the full coarsening phase. The difference is in the partial coarsening phase, where the *transfer-type* in Strategy-1 is doubling (h $\rightarrow$ 2h) and in Strategy-2 is quadrupling (h $\rightarrow$ 4h). In the full coarsening phase both the strategies are identical and consist of doubling transfers only.

In Strategy-1 we first identify the dimension(s) having the strongest coupling. This is indicated by the magnitude of the *coupling factor* $\tilde{c}_i$ defined as:

$$\tilde{c}_i = \varepsilon_i \times \left(\frac{N_i}{b_i - a_i} \right)^2 \tag{4.13}$$

see Equation 4.1 and 4.5. The larger the coupling factor $\tilde{c}_i$ the stronger the coupling. Fourier Analysis suggests that all dimensions having a coupling factor $\tilde{c}_i$ within a range of 1.3 times the largest coupling factor identified, can be doubled simultaneously. This decision is made (at each successive grid level) and identifies all those dimensions along which doubling will take place. Employing this strategy recursively makes the discrete problem isotropic inasmuch as all coupling factors are within this range, onwards from here full-doubling takes over. Once the coarsest possible grid is reached an exact solution takes place. In Section 4.5 we evaluate this strategy for the $O(h^2)$ $(2d + 1)$-point stencil and for the $O(h^4)$ $(4d + 1)$-point long stencil.

In Strategy-2 the threshold value is 1.0, which means that we only quadruple along the dimension(s) which are identified as having the strongest coupling (having the largest $\tilde{c}_i$). This ensures that quadrupling (in comparison with doubling) takes place along fewer dimensions. This strategy gives good convergence when employed in conjunction with optimal relaxation parameters, and, is cheaper than Strategy-1 because of quadrupling in the partial phase. Moreover, we suggest that a strategy based on quadrupling in the full coarsening phase should not be employed in a general multidimensional case as full quadrupling always loses against full doubling, and hence is quite apt to hamper multigrid convergence.

We take the grid size along each dimension always as a power of 2. When the anisotropy stems only from discretization on non-equidistant grids, as encountered in

the sparse-grid solution, the sequence of coarse grids generated by the two strategies are as follows.

Example 4.3.1. Grid coarsening with both strategies *Suppose that a particular discretization grid for a certain 5d problem is* $\mathbf{G} = [\ 32\ \ 8\ \ 8\ \ 128\ \ 32\ ]$. $\varepsilon_i = 1, \forall\ i$ *and* $\Omega = (0,1)^5$. *Then the sequence of grids that we get is the following:*

$Strategy-1$	$Strategy-2$
$\Omega_6 = [\ 32\ \ 8\ \ 8\ \ 128\ \ 32\ ]$	
$\Omega_5 = [\ 32\ \ 8\ \ 8\ \ 64\ \ 32\ ]$	$\Omega_4 = [\ 32\ \ 8\ \ 8\ \ 128\ \ 32\ ]$
$\Omega_4 = [\ 32\ \ 8\ \ 8\ \ 32\ \ 32\ ]$	$\Omega_3 = [\ 32\ \ 8\ \ 8\ \ 32\ \ 32\ ]$
$\Omega_3 = [\ 16\ \ 8\ \ 8\ \ 16\ \ 16\ ]$	$\Omega_2 = [\ 8\ \ 8\ \ 8\ \ 8\ \ 8\ ]$
$\Omega_2 = [\ 8\ \ 8\ \ 8\ \ 8\ \ 8\ ]$	$\Omega_1 = [\ 4\ \ 4\ \ 4\ \ 4\ \ 4\ ]$
$\Omega_1 = [\ 4\ \ 4\ \ 4\ \ 4\ \ 4\ ]$	$\Omega_0 = [\ 2\ \ 2\ \ 2\ \ 2\ \ 2\ ]$
$\Omega_0 = [\ 2\ \ 2\ \ 2\ \ 2\ \ 2\ ]$	

(4.14)

A similar experiment is shown available in Section 4.5, Table 4.7.

4.3.3 Coarse Grid Discretization

An important component in the coarse grid correction process is the choice of the coarse grid operator L_H. In this chapter we use the coarse grid analog of the discrete operator on the fine grid. Once the next coarser grid is decided we discretize the anisotropic diffusion operator using the same discrete stencils as presented in Section 4.2.

A particularly good choice of the coarse grid operator for the $O(h^4)$ fine grid discretization is to employ the $O(h^4)$ long stencil only along the non-coarsened dimensions of the grid and to discretize with the $O(h^2)$ stencil on the coarse grids along the dimensions where partial coarsening takes place. This has the marked advantage of saving CPU-time as now the coarse-grid operator has increased sparsity. Moreover, on very coarse grids this is advantageous because at grid points adjacent to boundary points the long stencil cannot be applied since it has entries which lie outside the discrete domain, whereas the $O(h^2)$-discretization can be applied throughout the domain. The overall accuracy remains fourth order as we have the fourth order accuracy on the finest grid. This coarse grid discretization scheme fits very nicely with the numerical experiments and *2-grid and 3-grid analysis* (not shown here) confirm this.

We do not use the Galerkin operator because of its disadvantage of being usually more dense than the simple coarse grid analog of the fine grid operator (unless special transfer operators are employed to generate the coarse grid operators). In high d dimensions this disadvantage becomes more serious and impractical.

4.3.4 The Transfer Operators

We employ the d-dimensional analogs of the Full-Weighting (FW) restriction operator and of the bilinear interpolation operator in two dimensions for the intergrid transfers of the grid functions. In this section we present a tensor formulation to generate

the restriction and prolongation operator matrices. The $2d$ FW restriction operator given in Description 2.3.7 is the Kronecker tensor product of the following x_1 and x_2 directional $1d$ FW operators:

$$(I_h^{2h})_{x_1} \triangleq \frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix}, \quad (I_h^{2h})_{x_2} \triangleq \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix}.$$

A formula based on Kronecker tensor products for building up a FW restriction operator matrix $\mathbf{R}$, reads

$$\mathbf{R} = \prod_{i=1}^{d} (\mathbf{R}_i)^{k_i}, \tag{4.15}$$

$$(\mathbf{R}_i)^{k_i} = \prod_{l=0}^{k_i-1}\left[\bigotimes_{j=i}^{d-1} \mathbf{I}_{N_{(d+i-j)}} \otimes \mathbf{O}_{\frac{N_i}{2^{(k_i-l-1)}}} \otimes \bigotimes_{j=1}^{i-1} \mathbf{I}_{\frac{N_{(i-j)}}{2^{k_{(i-j)}}}}\right].$$

The quantities involved in Equation 4.15 for the dummy subscript a, are:

$\mathbf{I}_a$ is the identity matrix of order $(a-1)\times(a-1)$.
$\mathbf{O}_a$ is the $1d$ FW restriction operator matrix, order $= (\frac{a}{2}-1)\times(a-1)$.
$\boldsymbol{N} = [N_1, N_2, \cdots, N_d]$, as in Equation 4.5.
$\boldsymbol{K} = [k_1, k_2, \cdots, k_d]$ is the coarsening request, k_i is the number of $(h \to 2h)$ transfers along the i^{th} dimension.

Example 4.3.2. Quadrupling along one dimension *The above description of $\boldsymbol{K}$ means that $k_i = 2$ for quadrupling along the i^{th} dimension. Suppose that a certain four-dimensional grid is given by $\boldsymbol{N} = [32, 32, 32, 128]$, and a transfer operator for quadrupling along the 4^{th} dimension has to be obtained through Equation 4.15, then in this case $K = [0, 0, 0, 2]$.*

Once the FW restriction operator matrix in d-dimensions is set, the prolongation (d-linear interpolation) operator matrix can be obtained by the following relation:

$$\mathbf{P} = 2^{(\sum_{i=1}^{d} k_i)}(\mathbf{R}^T). \tag{4.16}$$

A generalized restriction operator Equation 4.15 gives us the freedom to experiment with different types of coarsening strategies depending on the grid. Note that the FW restriction operator given by Equation 4.15 provides the required matrix for any number of coarsenings along any number of dimensions for an abstract d-dimensional problem.

4.3.5 Computational Work for d-Multigrid

The practical feasibility of a d-multigrid method also has to take into account an estimate of the computational work that it involves. Differing slightly from the notation from [2, 8], we call this work-estimate W_0, assuming that a particular multigrid method

is based on a hierarchy of grids $(\Omega_0, \Omega_1, \cdots, \Omega_{l-1}, \Omega_l,)$ where Ω_l is the coarsest grid. The *computational-work* W_0 per multigrid cycle on Ω_0 is modeled by the recursion:

$$W_{l-1} = W_{l-1}^{l} + W_l, \qquad W_k = W_k^{k+1} + \gamma_{k+1} W_{k+1} \quad (k = l-2, l-3 \cdots, 2, 1). \tag{4.17}$$

W_k^{k+1} is the work estimate for a single 2-grid cycle (h_k, h_{k+1}) excluding the work needed to solve the defect equation on $\Omega_{h_{k+1}}$ and W_l is the work required to solve exactly on the coarsest grid. It is reasonable to assume that the multigrid components (relaxation, computation of residual and the transfer of grid functions) applied to a single unknown require a number of arithmetic operations which -independent of k- is bounded by a small constant C. With M_k the total number of unknowns at grid level k we have,

$$W_k^{k+1} \leqslant C M_k \qquad (k = l-1, l-2, \cdots, 0).$$

For a fixed cycle index γ (the number of times the coarse grids are cycled) and with the work on the coarsest grid neglected this leads to

$$W_0 \leqslant C \left[M_0 + \gamma M_1 + \gamma^2 M_2 + \cdots + \gamma^{l-1} M_{l-1} \right]. \tag{4.18}$$

Before we proceed further, we consider it important to point out at this stage that the coarsening strategies (algorithms) that we use do not employ the same transfers at all grid levels. Depending on the anisotropy of the system, Strategy-1 employs $(h \rightarrow 2h)$ transfers (along the strongly coupled dimensions) so that after application of this strategy at each level the grid comes closer to isotropy. When an acceptable isotropy is achieved Strategy-1 resorts to standard full coarsening till the grid is brought to the coarsest possible level which the particular discretization allows. Strategy-2 is *hybrid* in the sense that it is composed of a mixture of standard and quadrupling transfers, moreover, it is *adaptive* just like Strategy-1. In Strategy-2 we employ full $(h \rightarrow 2h)$ transfers when isotropy is achieved. Note that for the type of grid-induced anisotropy encountered in the sparse-grid solution method, the number of dimensions along which the grid is coarsened, increases (or remains constant in the case of an equidistant grid) at each successive level. This gives the following:

$$\begin{aligned} W_0 &\leqslant \frac{2^{j \cdot s}}{(2^{j \cdot s} - \gamma)} C M_0, \\ W_0 &\leqslant \frac{\tau}{(\tau - \gamma)} C M_0 \qquad \text{for} \quad \gamma < \tau = 2^{j \cdot s}. \end{aligned} \tag{4.19}$$

Here j represents the number of dimensions coarsened and $s = 1$ for $(h \rightarrow 2h)$ transfers and $s = 2$ for quadrupling. The worst case would be when the grid is highly stretched along a single dimension, which implies that coarsening takes place only along $j = 1$ dimension. With quadrupling, i.e. $s = 2$, this still renders $\tau = 4$, which implies that the complexity of the method is still $O(M_0)$ for $\gamma = 1, 2, 3$. This feature makes quadrupling particularly attractive in higher dimensions for non-equidistant grids. Strategy-1 also gives an $O(M_0)$ algorithm for $\gamma = 1, 2, 3$ as long as $j \geq 2$ whereas for $j = 1$ one has to apply a V-cycle.

If the problem is isotropic with an equidistant grid, we essentially have $s = 1$ and $j = d$, which gives the work estimates that appear in Table 4.1.

Table 4.1: Multigrid work estimates for isotropic problem on equidistant grids.

γ	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
1	$\frac{4}{3}\ CM_0$	$\frac{8}{7}\ CM_0$	$\frac{16}{15}\ CM_0$	$\frac{32}{31}\ CM_0$	$\frac{64}{63}\ CM_0$
2	$2\ CM_0$	$\frac{4}{3}\ CM_0$	$\frac{8}{7}\ CM_0$	$\frac{16}{15}\ CM_0$	$\frac{32}{31}\ CM_0$
4	$O(M_0\ log_2\ M_0)$	$2\ CM_0$	$\frac{4}{3}\ CM_0$	$\frac{8}{7}\ CM_0$	$\frac{16}{15}\ CM_0$

The $2d$ and $3d$ results are well-known [2], and an estimate for a general d (isotropic problem on equidistant grid) can be obtained by setting $j = d$ in Equation 4.19.

This leads us now to apply the standard multigrid procedure to our model problem as the multigrid components have been adjusted for a general multidimensional setting, along with a computational complexity estimate.

4.4 Optimal Relaxation Parameters for ω-RB Jacobi

In this section, we concentrate on the efficient evaluation of relaxation parameters in context of the coarsening strategies developed in Section 4.3.2. For the most part in this section we carry on the definitions and notations used earlier in Chapter 3, and as present in [8, 9] and [7, 30].

Consider the d-dimensional anisotropic diffusion operator

$$-\sum_{i=1}^{d} \varepsilon_i \frac{\partial^2}{\partial x_i^2} \qquad \text{with } \varepsilon_i > 0. \tag{4.20}$$

The anisotropy is further induced by the use of unequal mesh sizes as well as by the unequal dimensions of the domain. Therefore, the first step in the analysis is to simplify the operator, which makes the analysis more convenient. In this regard, we first get rid of the effect that spatial non-uniformity brings, by considering an equivalent operator on an equidistant (hyper) square grid, i.e.,

$$-\sum_{i=1}^{d} \bar{c}_i \frac{\partial^2}{\partial x_i^2} \qquad \text{with } \bar{c}_i \in (0, \infty) \tag{4.21}$$

(where $\bar{c}_i$ are given by Equation 4.13), and then scaling the real positive coefficients $\bar{c}_i$ as in [7, 30], as follows:

$$-\sum_{i=1}^{d} c_i \frac{\partial^2}{\partial x_i^2} \quad \text{with} \quad c_i = \bar{c}_i / \sum_{j=1}^{d} \bar{c}_j \quad \text{and hence} \quad \sum_{i=1}^{d} c_i = 1. \tag{4.22}$$

In this section, anisotropies are modeled by the coefficients c_i, which is more appropriate for the analysis.

Table 4.2: $\mu(1)$, ω_{opt} and associated $\mu(\omega_{opt})$ for $\nu = 1$ relaxation sweep.

Left: *Doub. along all dims., equidistant grid, isotropy,* $O(h^2)$.
Right: *Quad. along all dims., equidistant grid, isotropy,* $O(h^2)$.

Doubling ($h \to 2h$)					*Quadrupling* ($h \to 4h$)			
d	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	ω_{ub}	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	ω_{ub}
2	0.25	1.049	0.16	1.072	0.73	1.315	0.31	1.316
3	0.44	1.133	0.23	1.144	0.81	1.398	0.40	1.393
4	0.56	1.195	0.28	1.202	0.86	1.454	0.45	1.455
5	0.64	1.243	0.31	1.250	0.89	1.496	0.50	1.502
6	0.69	1.283	0.35	1.285	0.90	1.528	0.53	1.519

Left: *Doub. along 1 dim., non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^2)$.
Right: *Quad. along 1 dim., non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^2)$.

Doubling ($h \to 2h$)					*Quadrupling* ($h \to 4h$)			
d	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	ω_{ub}	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	ω_{ub}
2	0.125	0.946	0.08	1.033	0.52	1.182	0.22	1.181
3	0.125	0.963	0.10	1.033	0.55	1.190	0.25	1.197
4	0.125	0.980	0.11	1.033	0.57	1.199	0.27	1.207
5	0.125	0.997	0.122	1.033	0.59	1.209	0.29	1.219
6	0.15	1.013	0.13	1.04	0.60	1.219	0.30	1.225

The corresponding eigenvalues (*Fourier symbols*) of L_h^{2o} and L_h^{4o} read:

$$\widetilde{L}_h^{2o}(\boldsymbol{\theta}) = \frac{2}{h^2}\left(1 - \sum_{i=1}^{d} c_i \cos(\theta_i)\right) \quad \text{and}$$
$$\widetilde{L}_h^{4o}(\boldsymbol{\theta}) = \frac{1}{6h^2}\left(15 - \sum_{i=1}^{d} c_i \left(16\cos(\theta_i) - \cos(2\theta_i)\right)\right),$$

respectively.

Consequently, we derive the Fourier symbols of ω-Jacobi relaxation for L_h^{2o} and L_h^{4o} discrete operators from Equation 3.10 as:

$$A^{2o}(\boldsymbol{\theta}, \omega) = 1 - \omega\frac{h^2}{2}\widetilde{L}_h(\boldsymbol{\theta}) = 1 - \omega\left(1 - \sum_{i=1}^{d} c_i \cos(\theta_i)\right) \quad \text{and}$$
$$A^{4o}(\boldsymbol{\theta}, \omega) = 1 - \omega\frac{6h^2}{15}\widetilde{L}_h(\boldsymbol{\theta}) = 1 - \omega\left(1 - \frac{1}{15}\sum_{i=1}^{d} c_i \left(16\cos(\theta_i) - \cos(2\theta_i)\right)\right),$$

respectively, see [18, 7, 30]. A rigorous version is derived in Appendix B. From

Table 4.3: $\mu(1)$, ω_{opt} and associated $\mu(\omega_{opt})$ for $\nu = 2$ relaxation sweeps.

Left: *Doub. along all dims., equidistant grid, isotropy,* $O(h^2)$.
Right: *Quad. along all dims., equidistant grid, isotropy,* $O(h^2)$.

	Doubling ($h \to 2h$)			*Quadrupling* ($h \to 4h$)		
d	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$
2	0.25	1.0107	0.23	0.73	1.3062	0.39
3	0.44	1.1136	0.28	0.81	1.3928	0.46
4	0.56	1.1832	0.31	0.86	1.4507	0.50
5	0.64	1.2356	0.35	0.89	1.4934	0.53
6	0.69	1.2771	0.37	0.90	1.5266	0.56

Left: *Doub. along 1 dim., non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^2)$.
Right: *Quad. along 1 dim., non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^2)$.

	Doubling ($h \to 2h$)			*Quadrupling* ($h \to 4h$)		
d	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$
2	0.23	0.8490	0.18	0.52	1.1598	0.30
3	0.23	0.8682	0.19	0.55	1.1735	0.31
4	0.23	0.8858	0.19	0.57	1.1864	0.32
5	0.23	0.9022	0.20	0.59	1.1986	0.32
6	0.23	0.9174	0.20	0.60	1.2100	0.33

Left: *Doub. along all dims., equidistant grid, isotropy,* $O(h^4)$.
Right: *Quad. along all dims., equidistant grid, isotropy,* $O(h^4)$.

	Doubling ($h \to 2h$)			*Quadrupling* ($h \to 4h$)		
d	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$
2	0.28	1.0260	0.25	0.76	1.3110	0.40
3	0.46	1.1108	0.29	0.84	1.3782	0.47
4	0.57	1.1683	0.33	0.87	1.4238	0.52
5	0.65	1.2128	0.36	0.90	1.4579	0.55
6	0.70	1.2492	0.38	0.91	1.4847	0.58

Left: *Doub. along 1 dim., non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^4)$.
Right: *Quad. along 1 dim., non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^4)$.

	Doubling ($h \to 2h$)			*Quadrupling* ($h \to 4h$)		
d	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$
2	0.24	0.8859	0.20	0.59	1.1900	0.35
3	0.26	0.8957	0.21	0.61	1.1942	0.37
4	0.25	0.9122	0.22	0.62	1.2046	0.38
5	0.25	0.9268	0.22	0.64	1.2146	0.38
6	0.25	0.9399	0.23	0.65	1.2243	0.39

Chapter 3, Equation 3.26, we have

$$\mu(\omega) := \sup_{\boldsymbol{\theta} \in T^{low}} \left\{ \sqrt[\nu]{\rho\left(\widetilde{Q}_h^H(\boldsymbol{\theta})\widetilde{S}_h^{\nu}(\boldsymbol{\theta}, \omega)\right)} \right\}.$$

which we now use to determine the optimal relaxation parameters for ω-RB Jacobi in d-dimensions. Towards this end, the first step is to define and subsequently discretize the low frequency subdomain. Theoretically, the number of frequency divisions should match the number of divisions employed for discretizing the problem, [9]. However, much fewer points in the frequency domain usually suffice to determine near optimal parameters. Irrespective of doubling or quadrupling, the low frequency Fourier domain gets extended with partial coarsening, and theoretically all these frequencies should be accounted in the optimization code (that searches out the best ω). The practical situation is somewhat different. It is always sufficient to search within the set of frequencies in the smallest low frequency domain, i.e., the low frequency domain for full coarsening. This is due to the fact that all frequencies outside this domain alias with a frequency inside this domain, and therefore redundant optimization checks can be easily eliminated. This is not intended to convey, that all frequencies outside the low frequency domain (for full coarsening) are high frequencies. The actual filtering of low and high frequencies according to the coarsening strategy is done by the projection operator.

The next important step is to define the projection operator Q_h^H precisely for the coarsening strategy at hand. Partial coarsening (based on doubling) along a subset of the dimensions is also incorporated simply by modifying the projection operator $\widetilde{Q}_h^H$ in the optimization code. In this case all those indices $\boldsymbol{\alpha}$ would be considered as corresponding to a low frequency that have a zero at all the positions corresponding to the coarsened dimensions, regardless of the entry at the positions corresponding to the non-coarsened dimensions.

Partial quadrupling is incorporated as before by a modification of the projection operator only. The projection operator in any case (partial or full) should contain zeros for all indices $\boldsymbol{\alpha}$ that index a frequency as low, and ones for all indices that index a frequency as high. Indices $\boldsymbol{\alpha}$ are d-tuples. Each one of the d positions, within the d-tuple index, represents a dimension. We categorize an index (and subsequently the frequency it is attached with) as high -if- at the positions corresponding to quadrupled dimensions, the index contains a [$\pm 1/2$ or 1] OR a [1] at the positions corresponding to doubled dimensions; otherwise we categorize it as low.

This describes how the optimal Red-Black relaxation parameters can be computed through a computer program. Note that there are explicit (but very lengthy) analytical formulas for the optimal relaxation parameters ω_{opt} in case of full coarsening applied to the second order discretization [18, 30]. Moreover, there is a close to optimal upper bound ω_{ub} [30] for the optimal relaxation parameters which is given by the following handy expression:

$$\omega_{opt} < \omega_{ub} = \frac{2}{1 + \sqrt{1 - \mu(\omega = 1)}}.$$

For partial coarsening and especially for the fourth order discretization it seems to be very difficult to derive analytic expressions for ω_{opt}. However, for the second order

discretization it turned out that ω_{ub} is a satisfactory approximation for ω_{opt} even in case of partial coarsening and quadrupling (but not necessarily an upper bound anymore), see Table 4.2 which presents ω_{opt} (optimized for $\nu = 1$) and ω_{ub} for equidistant and non-equidistant grids. This is a nice generalization of the results from [30]. For the fourth order discretization this is no longer true and we have to stay with the numerically derived values.

From the values in Table 4.3 (optimized for $\nu = 2$) we see that the *smoothness enhancement effect* of using optimal relaxation parameters is more prominent and pronounced with Strategy-2. With Strategy-1 this enhancement becomes prominent in the case of *nearly isotropic problems* (with grids that are equidistant along $(d-2)$ or more dimensions). With anisotropic problems where anisotropy is induced by discretization on grids highly elongated along a single dimension (and dealt with Strategy-1), the choice $\omega = 1$ is more suitable -first- because the optimal values themselves are very close to 1 and therefore do not bring about a substantial enhancement, -and second- the cost of relaxation itself is cut down, which saves CPU-time.

4.5 Numerical Experiments

We now present the results of some of our numerical experiments. The emphasis is on grid-induced anisotropies. The PDE is discretized on various equidistant and non-equidistant grids and experimental results are presented. We measure the quality of our multigrid method by the so-called *contraction number*, defined as follows:

Definition 4.5.1. (Multigrid contraction number) *The only quantities available to estimate the quality of a multigrid method are the defects d_h^i. We use this to define the multigrid contraction number q^m, measured over the last few (say, n) cycles of the multigrid solution.*

$$q^m := \left(\frac{\|d_h^m\|}{\|d_h^n\|} \right)^{1/m-n}. \tag{4.23}$$

where m is the total number of cycles the multigrid algorithm takes to match the stopping criterion (described later). We use the discrete L_2 norm for measurements as presented in [8].

The contraction number q^m is a good empirical estimate of the multigrid asymptotic convergence factor $\rho(M_h)$, i.e., the spectral radius of the actual multigrid iteration operator (see Chapter 2, Section 2.2.3). The numerical results -presented in the Tables (4.4 -4.7)- depict a close match between the theoretical smoothing factor μ, and q^m. μ is adapted from Equation 3.26.

We compute specific values of μ for each experiment (having its own coarsening pattern), and display them for correspondence with the contraction numbers in all the results. All the experiments employ one pre and one post smoothing, and so $\nu = 2$ has already been incorporated in the displayed values of μ. The optimal relaxation parameters ω_{opt} that we employ are also computed for $\nu = 2$. For each order of accuracy and for each dimension (up to $d = 6$), we have chosen two kinds of grids, one equidistant and one non-equidistant (highly stretched in one dimension).

Table 4.4: Results of numerical experiments for $V(1,1)$ and $W(1,1)$ on equidistant grids. The contraction number with the number of cycles consumed i.e. (q^m/# *it.*) is presented. For a correspondence-comparison the smoothing factors $[\mu(1)]^2$ and $[\mu(\omega_{opt})]^2$ (computed for coarsening based on the finest grid), is also displayed. Results presented include experiments with $\omega = 1$ as well as $\omega = \omega_{opt}$.

$G = 128^2$	V	W	V	W
	$[\mu(1)]^2 = 0.06$		$[\mu(1.011)]^2 = 0.05$	
$O(h^2)C_2^2$	0.10/8	0.06/7	0.09/8	0.05/7
$G = 128^2$	$[\mu(1)]^2 = 0.08$		$[\mu(1.026)]^2 = 0.06$	
$O(h^4)C_4^4$	0.13/9	0.10/8	0.12/9	0.08/7
$O(h^4)C_2^4$	0.10/8	0.07/7	0.09/8	0.05/7
$G = 128^3$	V	W	V	W
	$[\mu(1)]^2 = 0.20$		$[\mu(1.114)]^2 = 0.08$	
$O(h^2)C_2^2$	0.22/11	0.18/10	0.12/9	0.07/7
$G = 64^3$	$[\mu(1)]^2 = 0.21$		$[\mu(1.111)]^2 = 0.08$	
$O(h^4)C_4^4$	0.26/12	0.22/11	0.16/10	0.09/8
$O(h^4)C_2^4$	0.24/12	0.21/11	0.13/9	0.07/7
$G = 64^4$	V	W	V	W
	$[\mu(1)]^2 = 0.32$		$[\mu(1.183)]^2 = 0.10$	
$O(h^2)C_2^2$	0.33/14	0.30/12	0.16/10	0.08/7
$G = 32^4$	$[\mu(1)]^2 = 0.33$		$[\mu(1.168)]^2 = 0.11$	
$O(h^4)C_4^4$	0.39/16	0.34/14	0.20/10	0.11/9
$O(h^4)C_2^4$	0.35/15	0.34/14	0.15/9	0.11/8
$G = 16^5$	V	W	V	W
	$[\mu(1)]^2 = 0.41$		$[\mu(1.236)]^2 = 0.12$	
$O(h^2)C_2^2$	0.38/16	0.38/15	0.18/10	0.09/8
$G = 8^6$	V	W	V	W
	$[\mu(1)]^2 = 0.48$		$[\mu(1.277)]^2 = 0.14$	
$O(h^2)C_2^2$	0.35/15	0.34/15	0.12/9	0.11/9

q^m is displayed against the number of multigrid cycles that the experiment took to reduce the following quotient:

$$\frac{\|b_h - A_h u_h^m\|}{\|b_h\|}$$

by seven orders of magnitude. This termination criterion is equivalent to the one based on relative residual reduction because our starting solution in these experiments is always an all zero vector. The test function employed for the experiments is:

$$u(\boldsymbol{x}) = \frac{\sum_{i=1}^{d} sin(d\pi^2 x_i)}{d\pi + \sum_{i=1}^{d} x_i}, \tag{4.24}$$

$\varepsilon_i = 1 \ \forall \ i$, and $\Omega = (0,1)^d$ for all the problems. The values of this function at the boundary are taken as Dirichlet boundary conditions and its analytically computed

Table 4.5: Results of numerical experiments for $V(1,1)$ and $W(1,1)$ with Strategy-1 on grids stretched along 1 dimension. The observed contraction number against the number of cycles consumed i.e. $(q^m/\#\ it.)$ is presented. For a correspondence-comparison the smoothing factors $[\mu(1)]^2$ and $[\mu(\omega_{opt})]^2$ (computed for coarsening based on the finest grid), is also displayed. Results presented include experiments with $\omega = 1$ as well as $\omega = \omega_{opt}$.

$G = 512 \times 32$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.835)]^2 = 0.03$	
$O(h^2)C_2^2$	0.06/8	0.003/4	0.06/8	0.03/6
$G = 512 \times 32$	$[\mu(1)]^2 = 0.06$		$[\mu(0.879)]^2 = 0.04$	
$O(h^4)C_4^4$	0.10/8	0.03/6	0.09/8	0.07/7
$O(h^4)C_2^4$	0.10/7	0.02/6	0.05/7	0.04/6
$G = 512 \times 32^2$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.835)]^2 = 0.03$	
$O(h^2)C_2^2$	0.20/11	0.005/4	0.12/9	0.03/6
$G = 128 \times 32^2$	$[\mu(1)]^2 = 0.06$		$[\mu(0.978)]^2 = 0.06$	
$O(h^4)C_4^4$	0.24/11	0.04/6	0.17/9	0.05/6
$O(h^4)C_2^4$	0.17/10	0.04/6	0.11/8	0.04/6
$G = 128 \times 8^3$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.836)]^2 = 0.03$	
$O(h^2)C_2^2$	0.20/11	0.007/4	0.11/8	0.04/5
$G = 128 \times 32^3$	$[\mu(1)]^2 = 0.06$		$[\mu(0.9121)]^2 = 0.05$	
$O(h^4)C_4^4$	0.33/13	0.04/6	0.21/9	0.06/6
$O(h^4)C_2^4$	0.27/12	0.05/6	0.15/9	0.03/6
$G = 128 \times 8^4$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.837)]^2 = 0.03$	
$O(h^2)C_2^2$	0.24/12	0.009/4	0.11/8	0.04/6
$G = 128 \times 8^5$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.837)]^2 = 0.03$	
$O(h^2)C_2^2$	0.27/13	0.009/5	0.12/9	0.04/6

Laplacian forms the source function for these experiments. The experiments include the V -and the W-cycles. In some of the experiments the grids used for the $O(h^2)$ operator are different from the ones for the $O(h^4)$ operator. This only serves the purpose of accumulating results for slightly *different-sized* experiments. C_2^2 indicates the use of the second order stencil along all coarsened and non-coarsened dimensions, likewise for C_4^4. C_2^4 indicates the use of the $O(h^4)$-long-stencil along all non-coarsened dimensions and the use of the $O(h^2)$-stencil on the coarse grids along the dimensions where coarsening takes place. This *hybrid* coarse grid discretization gives $4th$ order accuracy and converges *faster* than the conventional 4^{th} order long stencil.

Table 4.4 presents experimental results for equidistant grids. A comparison of the convergence results with and without optimal relaxation parameters indicates the

Table 4.6: Results of numerical experiments for $V(1,1)$ and $W(1,1)$ with Strategy-2 on grids stretched along 1 dimension. The contraction number with the number of cycles employed i.e. (q^m/# *it*.) is presented. For a correspondence-comparison the smoothing factors $[\mu(1)]^2$ and $[\mu(\omega_{opt})]^2$ (computed for coarsening based on the finest grid), is also displayed. Results presented include experiments with $\omega = 1$ as well as $\omega = \omega_{opt}$.

$G = 512 \times 32$	V	W	V	W
	$[\mu(1)]^2 = 0.25$		$[\mu(1.147)]^2 = 0.09$	
$O(h^2)C_2^2$	0.27/13	0.24/11	0.14/10	0.08/9
$G = 512 \times 32$	$[\mu(1)]^2 = 0.32$		$[\mu(1.190)]^2 = 0.12$	
$O(h^4)C_4^4$	0.34/14	0.31/13	0.20/10	0.13/8
$O(h^4)C_2^4$	0.31/13	0.30/13	0.12/9	0.09/7
$G = 512 \times 32^2$	V	W	V	W
	$[\mu(1)]^2 = 0.25$		$[\mu(1.147)]^2 = 0.09$	
$O(h^2)C_2^2$	0.30/13	0.24/11	0.17/13	0.08/10
$G = 128 \times 32^2$	$[\mu(1)]^2 = 0.37$		$[\mu(1.194)]^2 = 0.14$	
$O(h^4)C_4^4$	0.34/14	0.35/14	0.22/10	0.13/9
$O(h^4)C_2^4$	0.34/14	0.35/14	0.16/9	0.12/8
$G = 128 \times 8^3$	V	W	V	W
	$[\mu(1)]^2 = 0.25$		$[\mu(1.148)]^2 = 0.09$	
$O(h^2)C_2^2$	0.32	0.24/11	0.16/13	0.08/10
$G = 128 \times 32^3$	$[\mu(1)]^2 = 0.39$		$[\mu(1.205)]^2 = 0.14$	
$O(h^4)C_4^4$	0.38/16	0.37/15	0.24/11	0.14/9
$O(h^4)C_2^4$	0.38/16	0.36/14	0.22/11	0.12/8
$G = 128 \times 8^4$	V	W	V	W
	$[\mu(1)]^2 = 0.26$		$[\mu(1.149)]^2 = 0.09$	
$O(h^2)C_2^2$	0.35/15	0.24/11	0.17/13	0.08/10
$G = 128 \times 8^5$	V	W	V	W
	$[\mu(1)]^2 = 0.26$		$[\mu(1.150]^2 = 0.09$	
$O(h^2)C_2^2$	0.38/16	0.24/11	0.18/13	0.08/10

benefits of using them for multidimensional problems. The cut down in the multigrid convergence factor as well as in the number of required multigrid cycles is quite significant for $d \geq 3$.

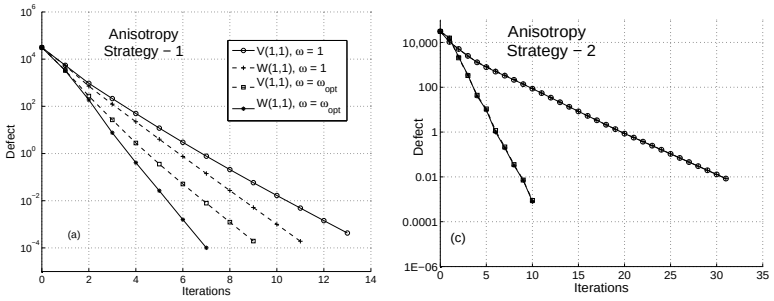
Table 4.5 presents experimental results for non-equidistant grids which we have chosen to be highly stretched in only one dimension. Because of this characteristic these experiments are computationally more expensive than any other as coarsening takes place only along the elongated dimension. This is exactly the opposite of the previous case, where coarsening took place along all dimensions and hence the cut down in the number of unknowns at each level was optimal. In this table we display the results that we get from Strategy-1 which is based purely on $h \rightarrow 2h$ transfers. Optimal relaxation parameters in this case pay off only with V-cycles, with $\omega = 1$

serving as a perfect compromise with W-cycles.

Table 4.7: A general 5*d* experiment on a non-equidistant grid. The finest grid is given by $G = [32 \;\; 8 \;\; 8 \;\; 128 \;\; 32]$. The domain is $\Omega = (0.11, 1.21)\times(0.5, 1.51)\times(0.25, 1.26)\times(0, 1) \times (0, 1.11)$, and the constant coefficients $\boldsymbol{c} = [200.33, 10^{-7}, 2, 1.5, 200.49]$. [H]

Strategy 1	V	W	V	W
	$[\mu(1)]^2 = 0.08$		$[\mu(1.031)]^2 = 0.058$	
$O(h^2)C_2^2$	0.13/11	0.07/9	0.08/9	0.058/8
Strategy 2	V	W	V	W
	$[\mu(1)]^2 = 0.55$		$[\mu(1.319)]^2 = 0.16$	
$O(h^2)C_2^2$	0.53/31	0.53/31	0.16/11	0.14/11

In Table 4.6 we have reworked the experiments of Table 4.5 but with Strategy-2 this time. Partial quadrupling (Strategy-2) ensures an $O(M_0)$ algorithm even with grids of this type. Note that the computational complexity of the experiments in Table 4.5 is $O(M_0\ log_2\ M_0)$, even though the multigrid convergence factor is quite impressive there. These results show the important role of optimal relaxation parameters in enhancing convergence of multigrid with quadrupling transfers.



(a) Strategy-1 (b) Strategy-2

Figure 4.1: Convergence behaviour of different multigrid cycles for a 4*d* problem, on a non-equidistant grid [64 64 64 16] (anisotropy with 3,750,705 unknowns)

To make the discussion complete we have included some more experiments on a *nearly equidistant grid*. The results are reported in Figure 4.1. The contraction number q^m is plotted against the number of cycles. A comparison of the results of Strategy-1 and Strategy-2 suggests that for these kind of grids, a combination of Strategy-2 with V-cycles and optimal relaxation parameters works very nicely.

We also performed a general 5*d* anisotropic experiment. The different domain dimensions are not equal. The results are displayed in Table 4.7. Clearly, V-cycles with optimal weighting in the relaxation process coupled with either coarsening strategy gives excellent results. We would like to point out that we expect Strategy-1 to be more dependable than Strategy-2 if the number of dimensions coarsened on the finest

grid is greater than $d/2$.

Remark 4.5.1. (Reaching *FMG* optimality) *The multigrid convergence factors displayed in the tables are mostly under* 0.1*, implying that a full multigrid algorithm starting on the coarsest grid is expected to reach an approximate solution up to the discretization accuracy in just one or two cycles.*

Remark 4.5.2. (Solving problems on very coarse grids) *It is also important to point out that for very coarse discretization grids (say 8 points along all dimensions), the asymptotic convergence of the iterative method ω-RB Jacobi is also quite satisfactory. A $6d$ problem (on a very coarse finest grid) takes the same CPU-time to asymptotically converge with ω-RB Jacobi as with multigrid.*

Chapter 5

d-Multigrid as Bi-CGSTAB preconditioner

In this chapter we present a robust solver based on the Krylov subspace method Bi-CGSTAB combined with the *d*-multigrid method as a preconditioner. Instead of developing *the perfect multigrid method*, as a stand-alone solver for a single problem discretized on a certain grid, we aim for a method that converges well for a wide class of discrete problems arising from discretization on various anisotropic grids. This is exactly what we encounter during a sparse grid computation of a multidimensional problem.

5.1 Overview: *d*-Dimensional PDEs & Methods

Multidimensional PDEs can be parabolic, and they have application in a variety of fields, as we discussed in Chapter 4, Section 4.1. In the context of their numerical approximation there are two main limiting factors on computing speed; one is the phenomenon of *temporal march* of successive solutions and the other is the multiple spatial-dimensionality of the problem which renders an exponential computational complexity on regular tensor product grids. Traditionally, this exponential growth in the number of discrete unknowns is known as the *curse of dimensionality* [40], and it has continually eluded and marred solution techniques. The *sparse grid* solution method [41, 42, 43] relieves this so-called curse to some extent (and for a limited kind of problems). The technique consists of discretizing the problem on many grids, each with lesser number of nodes (sparse grids), solving these subproblems, and then combining the solutions to obtain a *mimic* of the solution on the original *dense* grid. Different combination techniques can be employed; we use the technique proposed in [42].

The efficiency of the sparse grid solution method depends on the efficient solution of the underlying subproblems, some of which are highly anisotropic, and each of which has the same spatial dimensionality as the original problem. This was one of

the principal reasons for the development of d-multigrid in Chapter 4. In this chapter, we take on a challenging multidimensional application problem from mathematical finance, and develop a solver based on d-multigrid preconditioned Bi-CGSTAB.

Bi-CGSTAB [44] is well known and belongs to the Krylov family of general purpose iterative solvers. It is widely accepted that all iterative solvers based on Krylov subspaces require preconditioning for faster convergence. Preconditioning is a process aimed at clustering the scattered eigenvalues of the coefficient matrix. It is important to point out that the performance of stand alone multigrid is dependent significantly on the choice of optimal parameters and components, especially for high-dimensional problems; however, this is not quite the case when multigrid is used as a preconditioner. By choosing multigrid as a preconditioner we do not need to search for the ideal under-relaxation, which is grid anisotropy dependent, but can rather stay with fixed parameters. Important theoretical and experimental insights into multigrid preconditioning of Krylov subspace solvers can be had from earlier work in this context [45, 46, 14].

In Section 5.2 we point out that the Black-Scholes multi-asset option pricing PDE can be reduced to a standard d-dimensional diffusion equation, which underscores the need of an efficient solver for discrete diffusion systems. Next, in Section 5.3 we present the sparse grid technique along with the computation of accuracy bounds. Section 5.4 deals with the preconditioner and its components which include point smoothing and grid transfer strategies. These strategies are based on the idea of repeated partial coarsening in the direction(s) of strong coupling. In Section 5.5 we present experimental results based on the full grid solution method. This includes results for d-multigrid employed both as a stand alone solver as well as a preconditioner. Section 5.6 contains results from numerical experiments (on the model problem) based on the sparse grid technique with d-multigrid preconditioned Bi-CGSTAB. There we demonstrate in tables and figures the results that we get; and finally, in Section 5.7 we solve the same transformed Black-Scholes PDE of Section 5.2, exhibiting the convergence of the proposed method in a real application.

5.2 Financial Application with Model Problem

The multi-asset Black-Scholes option pricing (parabolic) PDE [47] is defined as:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} + \sum_{i=1}^{d}(r-\delta_i)S_i\frac{\partial V}{\partial S_i} - rV = 0, \qquad (5.1)$$
$$(0 < S_1, ..., S_d < \infty, \quad 0 \leqslant t < T).$$

V stands for the option price; S_i are the d underlying asset prices; t, the current time; ρ_{ij}, the correlation coefficients between the i^{th} and the j^{th} asset prices; σ_i, the volatility of the i^{th} asset price; r, the risk free interest rate and δ_i the continuous dividend yield for asset i. The equation comes with a final condition,

$$V(\mathbf{S}, T) = \max\{\sum_{k=1}^{d} w_k S_k - K, 0\}, \qquad (5.2)$$

where K is the exercise price and $0 \leq w_k \leq 1$ are the weights of the assets in the basket. The PDE represented by Equation 5.1 is a second order partial differential equation in d dimensions and $2d$ boundary conditions are mandatory. As the asset price domain is truncated $S_k \in [0, S_k^{\max}]$, we first of all need a boundary condition at $S_k = 0$. When using the reduced form of Equation 5.1, where every coefficient belonging to a derivative with respect to S_k vanishes at $S_k = 0$, a $(d-1)$-dimensional partial differential equation remains at the boundary. This is sometimes called the *natural boundary condition*. In particular, the boundary condition at $S_1 = 0$ or $S_2 = 0$ for a 2-asset option is represented by the well known 1D Black-Scholes equation for a vanilla option.

Also for $S_k = S_k^{\max}$ a boundary condition must be prescribed. If $S_k^{\max}$ is large enough, i.e. $w_k S_k^{\max} \gg K$, a linearity condition can be applied, which means that the option price can be assumed to show a linear growth in that coordinate direction. In this case we set the second derivative with respect to S_k equal to zero at that boundary. All other derivatives remain present. An appropriate size of the truncated domain is important for this boundary condition not to have a negative effect on the option prices at the spot price and/or at the exercise price K.

It has been shown [47, 48] that Equation 5.1 under the following simple log transform of the asset price S_i:

$$\begin{aligned} x_i &= \frac{1}{\sigma_i}\Big(r - \frac{{\sigma_i}^2}{2}\Big)\tau + \frac{1}{\sigma_i} ln S_i, \quad i = 1, 2, ..., d; \\ \tau &= T - t, \end{aligned} \tag{5.3}$$

converts into the d-dimensional diffusion equation, given by:

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}\sum_{i=1}^{d} \frac{\partial^2 V}{\partial {x_i}^2}, \qquad -\infty < x_i < \infty, \quad 0 < \tau \leqslant T, \tag{5.4}$$

Thus we choose the d-dimensional diffusion equation, with transformed initial conditions and Dirichlet boundary conditions, to serve as our model problem, in this chapter.

In what follows $\boldsymbol{x}$ is a d-tuple $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$. For $\{a_i, b_i, c_i, \tau_1, \tau_2\} \in \mathbb{R}$ and $c_i > 0$ the model problem reads:

$$\begin{aligned} \frac{\partial}{\partial t} u(\boldsymbol{x}, t) &= \sum_{i=1}^{d} c_i \frac{\partial^2}{\partial {x_i}^2} u(\boldsymbol{x}, t); \quad \boldsymbol{x} \in \left(\Omega = \prod_{i=1}^{d} [a_i, b_i]\right) \subset \mathbb{R}^d; \quad t \in [\tau_1, \tau_2]; \\ u(\boldsymbol{x}, t) &= f^{\Gamma}(\boldsymbol{x}, t); \qquad\qquad \boldsymbol{x} \in \Gamma = \partial\Omega; \quad x_i \in \{a_i, b_i\}. \end{aligned} \tag{5.5}$$

We use the implicit (second order) Crank-Nicolson time stepping scheme for the temporal discretization. The spatial part is handled by second order FDM, also used in Chapter 4. For the model problem we set Dirichlet boundary conditions at all spatial boundaries of the domain. In contrast with Chapter 4, we use the non-eliminated boundary scheme in this chapter. This is helpful for transition between the actual application and the model problem. The spatial discretization grid is given by $\boldsymbol{N} = [N_1, N_2, \cdots, N_d]$, N_i represents the number of divisions along the ith dimension, and

the total number of points in the finest grid (incl. boundaries) is $M = \prod_{i=1}^{d}(N_i + 1)$. As in Chapter 4, we represent the index of a grid point by a d-tuple $(j_d, j_{(d-1)}, \cdots, j_1)$

Written in terms of matrix operators, the Crank-Nicolson time stepping scheme reads:

$$(\boldsymbol{L}_h + \boldsymbol{D}_k)u_{\boldsymbol{h},k}(\boldsymbol{x}, t + k) = (-\boldsymbol{L}_h + \boldsymbol{D}_k)u_{\boldsymbol{h},k}(\boldsymbol{x}, t) \tag{5.6}$$

The matrix operators have the order $(M \times M)$. $\boldsymbol{D}_k$ is a diagonal matrix containing $1/k$ on the main diagonal, and $\boldsymbol{L}_h$ is the iteration matrix obtained similarly as in Chapter 4, Section 4.2.2. Kronecker tensor products are employed in this work for defining the d-dimensional operators. This completes the discretization.

5.3 The Sparse Grid Method

Consider the task of the numerical approximation of a parabolic d-dimensional problem discretized with 2^n points per spatial coordinate. Without loss of clarity we substitute N for this number. The grid thus formed is termed as a *full grid* and a full grid based solution process involves (at the minimal), vectors of the size $2^{n.d}$. For 6 space dimensions and only 32 divisions along each axis, the storage cost is around 9 gigabytes per vector, and grows worse for increasing d. The sparse grid approach, developed by Zenger and co-workers [41, 43] is a technique that splits the full grid problem of $M = N^d$ points up into layers of subgrids. Each sub-grid represents a coarsening in several coordinates up to a minimal required number of points. In the so-called *sparse grid combination technique*, the partial solutions that are computed on these grids, are combined a-posteriori by interpolation to a certain point or region.

Definition 5.3.1. Sparse grid multi-index *A multi-index for a d−dimensional grid is denoted by $\mathcal{I}_d$, and is a d−tuple n_i, $i = 1, \ldots, d$, which represents a d-dimensional grid with N_i grid points in coordinate i, with $N_i = 2^{n_i}$. The sum of a multi-index $|\mathcal{I}_d|$ is defined by:*

$$|\mathcal{I}_d| = \sum_{i=1}^{d} n_i \tag{5.7}$$

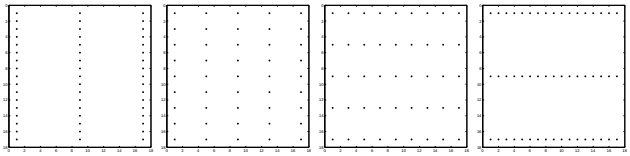
Example 5.3.1. Layer number *According to Definition 5.3.1 the multi-index $\mathcal{I}_d$ of a full grid with $N = 2^n$ points per coordinate reads $\mathcal{I}_d = \{n, n, \ldots, n\}$, with $|\mathcal{I}_d| = nd$ being the layer number.*

The full grid solution will be denoted by u_n^f; the sparse grid solution after the combination will be denoted by u_n^c and the exact solution by u_E. Now, we can define the sparse grid solution [42], as:

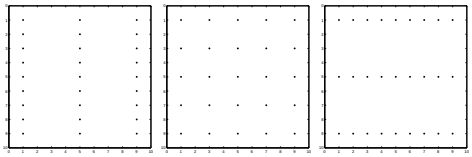
Definition 5.3.2. The sparse grid solution *The combined sparse grid solution u_n^c corresponding to a full grid solution u_n^f reads*

$$u_n^c = \sum_{k=n}^{n+d-1} (-1)^{k+1} \binom{d-1}{k-n} \sum_{|\mathcal{I}_d|=k} u_{\mathcal{I}_d}^f, \tag{5.8}$$

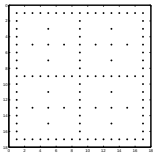
with $u_{\mathcal{I}_d}^f$ being the solution of the problem on a grid with multi-index $\mathcal{I}_d$ such that $|\mathcal{I}_d|$ equals k.



(a) 16×2 (b) 8×4 (c) 4×8 (d) 2×16



(e) 8×2 (f) 4×4 (g) 2×8



(h) Combined

Figure 5.1: Construction of a 2D sparse grid; (a)–(d): grids on layer 5, (e)–(g): grids on layer 4; (h) combined sparse grid solution

For solutions having bounded mixed derivatives, the sparse grid representation (for most practical purposes) can be used instead of the full grid solution. For a simple $2d$ case, the subgrids (as constructed by the sparse grid scheme) are depicted in Figure 5.1, Diagrams (a) - (g). Note that the *shape of the stretch* in all these grids is different, which implies that in each of these subproblems we have a different grid induced anisotropy. If the subgrids are simply combined without any interpolation, which means that all the evaluated points in every sub-grid are added with the binomial coefficients given in Equation 5.8; then the number of points in the full grid with $n_i = n$ reads $N_f = (2^n)^d$. From Equation 5.8 it follows that the number of problems to be solved in the sparse grid technique reads:

$$Z_{n,d} = \sum_{k=n}^{n+d-1} \binom{k-1}{d-1} = \frac{n}{d}\binom{n+d-1}{d-1} - \frac{n-d}{d}\binom{n-1}{d-1} \tag{5.9}$$

Furthermore the number of points N_n employed in a grid with $|\mathcal{I}_d| = n$ reads

$$N_n = 2^n. \tag{5.10}$$

Combining (5.9) and (5.10) results in the total number of points employed in the sparse grid technique

$$N_n^c = \sum_{k=n}^{n+d-1} N_k \binom{k-1}{d-1} = \sum_{k=n}^{n+d-1} \binom{k-1}{d-1} 2^k \tag{5.11}$$

It is known that the error of the discrete solution from a second order finite difference discretization of the 2D Laplacian can be split [49] as

$$u_n^f - u_E = C_1(x_1, h_1)h_1^2 + C_1(x_2, h_2)h_2^2 + D(x_1, h_1, x_2, h_2)h_1^2 h_2^2 \tag{5.12}$$

With the combination technique as in Definition 5.3.2 and the splitting in (5.12), the dimension dependent absolute error (for the Laplacian), reads, [50]:

$$\epsilon_n = |u_n^c - u_E| = \mathbf{O}(h_n^2 \left(\log_2 h_n^{-1}\right)^{d-1}), \tag{5.13}$$

with h_n the finest mesh size along one dimension.

5.4 The *d*-Multigrid Preconditioner

We employ d-multigrid (described in ample detail in Chapter 4) as a preconditioner for Bi-CGSTAB due to the robustness that the resulting solver possesses. The components that we typically use for the work in this chapter, are (1) pointwise RB Jacobi for smoothing, (2) DCG operator on the coarse grids, and (3) FW restriction and bilinear prolongation for grid transfer. We use either one $V(1,1)$, or one $F(1,1)$ cycle for each of the two preconditioning steps in standard Bi-CGSTAB.

An advantage of the pointwise smoothing method combined with the partial coarsening technique adopted here is that each problem in the sparse grid setting gets an individual treatment. The coarsening pattern is different for, -and automatically adapted to each anisotropic grid that is involved . This may not be the case for a multigrid method with a fixed coarsening and hyperplane smoothing strategy. Furthermore, a multidimensional multigrid method based on a parallel pointwise RB Jacobi smoothing method can be parallelized. Nevertheless, the parallelization of a multidimensional algorithm is essentially non-trivial.

5.5 Experiments Based on the *Full grid Solution* Method

In this chapter *full grid solution* refers to a solution on a regular tensor product grid (where the sparse grid technique is not used). As described in the previous section, we have quite a strong and robust multigrid preconditioner. Before we actually use it in the sparse grid setting, we would like to test its performance as a stand alone solver versus as a preconditioner for Bi-CGSTAB in a full grid solution process.

5.5.1 *d*-Multigrid Performance in Stationary Cases

A useful numerical insight for time marching solution processes (our ultimate aim here) comes from an insight into the stationary process per time step. We use the test function given by Equation 4.24. We have conducted a number of numerical experiments -isotropic and anisotropic- and have included the convergence graphs for them. These graphs show the residual reduction against multigrid cycles for d-multigrid used in these two contexts (solver and preconditioner). Multigrid is an $O(M)$ solver (where M is the number of unknowns on the finest grid) when optimal relaxation and ideal

coarse grid correction are available. In such a situation multigrid is extremely efficient. Some of the graphs here show a tough competition between multigrid as a solver against multigrid as a Bi-CGSTAB preconditioner. This happens due to the fact that for the model problem the employed relaxation method and the coarse grid correction form near optimal d-multigrid attributes. Evidently, in any situation where tuning multigrid with optimal attributes is not a choice, multigrid works better as a Krylov-preconditioner than as a stand alone solver.

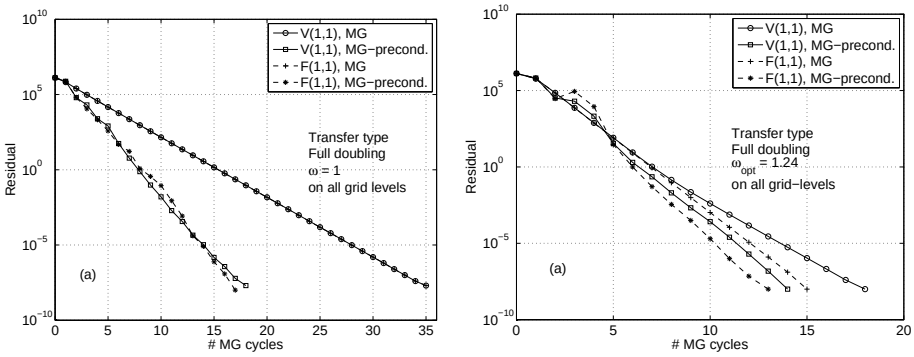


Figure 5.2: Convergence diagram for a 5-dimensional isotropic problem, 32 divisions along each dimension.

First of all, we check out d-multigrid performance for a 5-dimensional isotropic case, with 32 divisions along all dimensions of the domain. The number of unknowns in the system is 39, 135, 393. In this case the V(1,1) multigrid (multigrid method based on V cycles with 1 pre and 1 post smoothing steps) preconditioned Bi-CGSTAB far out performs the V(1,1) multigrid solver; Figure 5.2, (here we choose $\omega = 1$ in ω-RB Jacobi). However, with $\omega_{opt} = 1.24$ included in the game (a possibility for the model problem) the comparison is not as bright. This confirms that no great Krylov induced enhancement should be expected when multigrid (as a solver) approaches optimality.

Next we present some experiments based on problems with *discrete anisotropies* that result from discretization on a non-equidistant grid, i.e. a grid where the number of divisions is different along different dimensions of the hyper domain. We have selected 3 multidimensional problems, each with a different discrete anisotropy. The problems have been chosen with the aim of harvesting experimental results for grids highly stretched along 1 dimension as well as grids highly stretched along multiple dimensions. The anisotropies are handled with the partial coarsening schemes as illustrated in Chapter 4, Section 4.3.2. The results appear in Figures 5.3, 5.4 and 5.5. Here, we find that the F(1,1) MG cycles are more suited than V(1,1), both for the stand alone multigrid solver as well as a preconditioner for Bi-CGSTAB, if the coarsening strategy is based on doubling. Quadrupling suits the situation more when the grid is stretched along only a few dimensions, (preferably $< d/2$) and when optimal relaxation parameters are available. However, with quadrupling, V(2,2) and F(1,1) cycles seem to yield better results than V(1,1).

In Figure 5.6 we have only chosen the CPU-time scale (for presenting results)

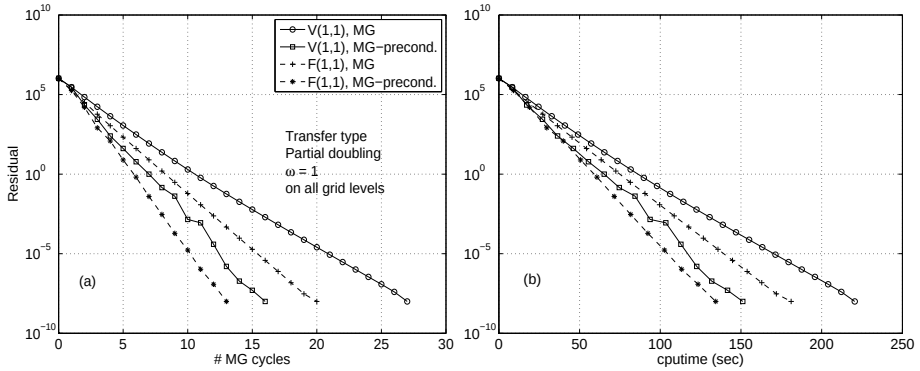


Figure 5.3: Convergence diagram for a 6-dimensional anisotropic problem, grid stretched along $d/2$ dimensions and given by $\boldsymbol{N} = [32, 8, 32, 8, 32, 8]$. # of unknowns is 26,198,073. Diagram (a) shows a comparison between multigrid as a solver and multigrid as a preconditioner, on the iteration scale, Diagram(b) on the CPU-time scale.

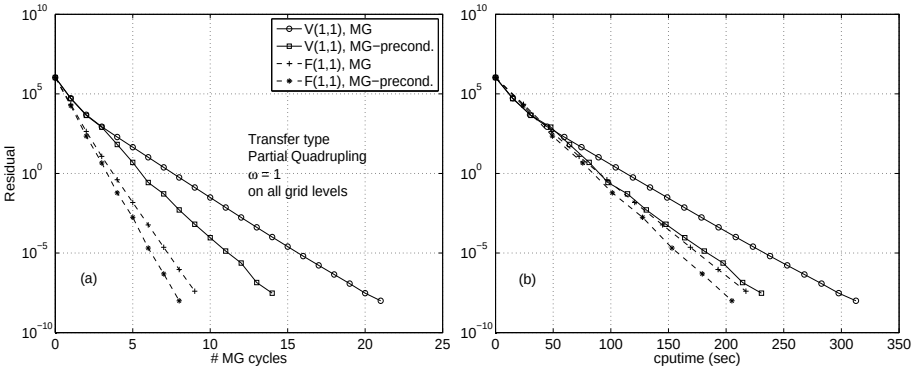


Figure 5.4: Convergence diagram for a 7 dimensional anisotropic problem, grid stretched along 1 dimension, and given by $\boldsymbol{N} = [8, 8, 8, 64, 8, 8, 8]$, # of unknowns is 34,543,665.

because with difference in the transfer scheme (doubling vs quadrupling), the number of multigrid cycles are not really comparable. Quadrupling -in contrast with doubling- relies on optimal relaxation to quite some extent; in fact, the better the relaxation process the shorter the CPU-time. This points us to the fact that if optimization in the relaxation process is an impossibility we might be better off with doubling for all kinds of grid based discrete anisotropies.

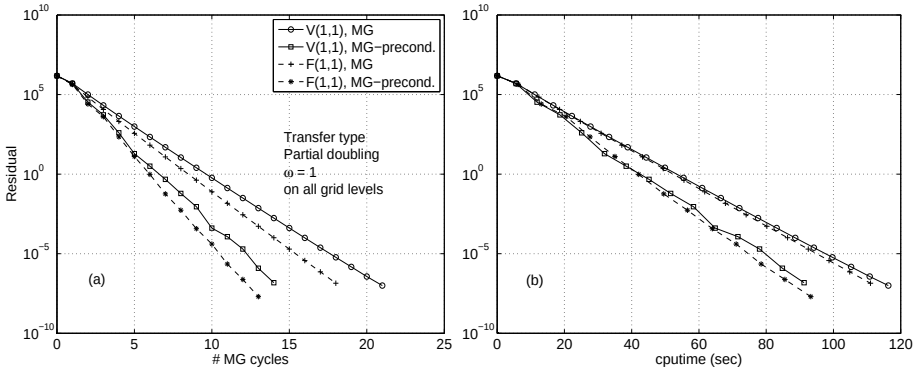


Figure 5.5: Convergence diagram for a 4 dimensional anisotropic problem, grid stretched along $(d-1)$ dimensions, and given by $\boldsymbol{N} = [128, 128, 128, 8]$, # of unknowns is 19,320,201

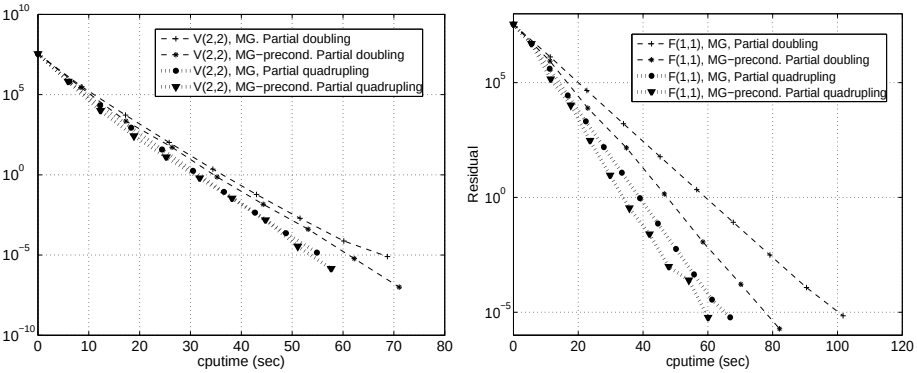


Figure 5.6: Convergence diagrams comparing d-multigrid based on doubling transfers against quadrupling transfers for V(2,2) -Diagram (a)- and F(1,1) -Diagram(b)- multigrid cycles. This 5 dimensional problem is discretized on $\boldsymbol{N} = [8, 8, 2048, 8, 8]$, # of unknowns is 13,443,489.

5.6 Experiments Based on the *Sparse grid Solution*

5.6.1 d-Multigrid as a Preconditioner in the Time-independent Case

As the solver works for every kind of grid that might arise within the sparse grid setting (as described in section 4), the aim is now to reach a reasonable number of dimensions. The test function for the sparse grid stationary experiment reads:

$$u(\boldsymbol{x}) = \prod_{i=1}^{d} e^{x_i^2} = \exp\left(\sum_{i=1}^{d} x_i^2\right), \tag{5.14}$$

with $\Omega = [0,1]^d$, $c_i = 1$. The approximation is done for $2 \leqslant d \leqslant 8$ with a mimic of the grid with 1024 cells per coordinate. In a full grid setting the maximum problem would

have a size of $1024^8 = 2^{80}$, i.e., 2^{53} GB of memory, which is - of course - immensely huge. The maximum size of an 8D problem in the sparse grid setting chosen here is

Table 5.1: Time independent experiments of problem (5.14) using sparse grids. TOP: is the $2d$ case. BOTTOM: $8d$ case. Column one gives n_{max}, the largest number of cells in one coordinate.

n_{max}	Value	Error	Conv	Time	#probl	Th. Conv
d=2						
16	1.65	$5.52 \cdot 10^{-3}$	3.05	0.04 sec.	7	3.00
32	1.65	$1.72 \cdot 10^{-3}$	3.21	0.07 sec.	9	3.20
64	1.65	$5.16 \cdot 10^{-4}$	3.34	0.10 sec.	11	3.33
128	1.65	$1.50 \cdot 10^{-4}$	3.43	0.12 sec.	13	3.43
256	1.65	$4.30 \cdot 10^{-5}$	3.50	0.16 sec.	15	3.50
512	1.65	$1.21 \cdot 10^{-5}$	3.55	0.25 sec.	17	3.56
1024	1.65	$3.36 \cdot 10^{-6}$	3.60	0.33 sec.	19	3.60
d=8						
16	7.63	$2.43 \cdot 10^{-1}$	1.50	18.51 sec.	165	0.53
32	7.54	$1.48 \cdot 10^{-1}$	1.64	1m 51s	495	0.84
64	7.47	$8.33 \cdot 10^{-2}$	1.77	9m 38s	1287	1.12
128	7.43	$4.40 \cdot 10^{-2}$	1.89	40m 4s	3003	1.36
256	7.41	$2.20 \cdot 10^{-2}$	2.00	2h 31m 7s	6435	1.57
512	7.40	$1.04 \cdot 10^{-2}$	2.10	9h 8m 45s	12869	1.75
1024	7.39	$4.76 \cdot 10^{-3}$	2.19	29h 40m 12s	24301	1.91

$1024 \times 2^7 = 2^{17}$, which is only 1 MB. The solution at the central point in the domain $x_i = 0.5$ is computed here. The results are described in detail for $d = 2$ and $d = 8$ in Table 5.1 and the error and convergence for all values of d are plotted in Figure 5.7. In the table, the time indicated is the total computational time for the sparse grid solution including the interpolation. The number of problems $\#probl$ is as in Equation 5.9 and the theoretical convergence $Th.Conv$ from Equation 5.13.

The table and figures show the dependence of the number of dimensions in the convergence according to the theoretical convergence ratio in equation (5.13). Although the theoretical convergence of the sparse grid method is low when d is high at small numbers of n_{max} (the largest number of cells in one direction), the convergence in this test experiment is reasonable. A possible reason may be the smoothness of the analytic solution.

Table 5.2 presents a comparison of the total number of multigrid cycles when multigrid is employed both as a solver as well as a preconditioner for Bi-CGSTAB for solving all sparse grid subproblems on a layer of a d-dimensional problem, with increasing d. Presented are the maximum, the minimum and the average numbers of

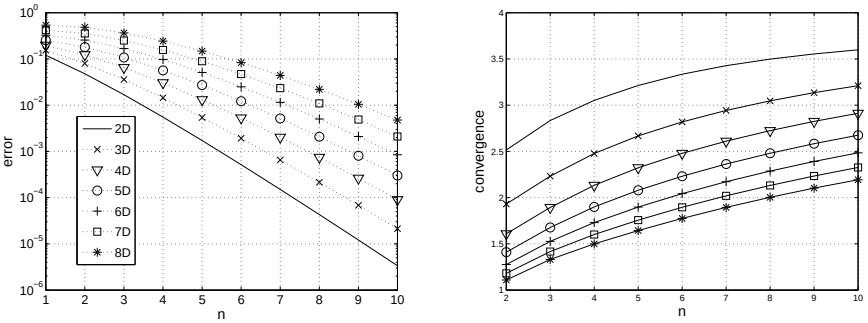


Figure 5.7: LEFT: Decay of the error $|u_n^c - u_E|$, with u_E the exact solution (5.14). RIGHT: Convergence of the error in the left picture.

iterations, as well as the total number of subproblems solved on which these numbers are based. The stopping criterion is the residual being smaller than 10^{-14}, which is severe but gives a good insight in the comparison. For both solvers we choose smoothing relaxation parameter $\omega = 1$. We see that the average number of multigrid cycles -when multigrid is used as a preconditioner instead of being used as a solver- does not reduce significantly for low values of d. However the cycle difference in the two usages does become significant for higher d because then the total number of subproblems also increase binomially. As the number of subproblems to be solved is more than 5000 for the 7D problem a gain in average of about 2 multigrid iterations is still interesting. The time for the highest level in the case $d = 8$ was over 10^5 seconds which is 28 hours. However, the number of subproblems is 24300, so the average computational time per grid is only 5 seconds. If the sparse grid method is parallelized over 10 machines, the time would be 2.8 hours in total, because the sparse grid is only a combination technique of subproblems.

5.6.2 d-Multigrid as a Preconditioner in the Time dependent Case

For the time dependent case, we choose as the test problem solution,

$$u(\boldsymbol{x}, t) = e^t \prod_{i=1}^{d} e^{\frac{x_i}{\sqrt{d}}} = \exp\left(t + \frac{1}{\sqrt{d}} \sum_{i=1}^{d} x_i\right), \tag{5.15}$$

with $\Omega = [0, 1]^d$, $t \geqslant 0$ and $c_i = 1$. The approximation is done for $2 \leqslant d \leqslant 5$ with 1024 cells as the maximum number per coordinate in the sparse grid technique. The solution of the central point in the domain $x_i = 0.5$ is computed at time $t = 0.1$. The number of time steps used is fixed at 400.

The *grid convergence* results are described in detail for $d = 2$ and $d = 5$ in Table 5.3 and the errors and convergence for all values of d are plotted in Figure 5.8. In the table, again "time" is the total computation time for the complete sparse grid solution including the interpolation and time integration. The number of problems is as in (5.9) and the theoretical convergence from equation (5.13).

Table 5.2: Comparison of the total number of pure multigrid and multigrid preconditioned Bi-CGSTAB iterations (maximum, minimum and average number), for all the subgrids on the finest layer of a d-dimensional problem. # grids is the number of subgrids involved in the sparse grid solution

d	Max	Min	Average	# grids
	Bi-CGSTAB with MG			
2	12.0	4.0	9.400	10
3	14.0	6.0	10.691	55
4	14.0	6.0	10.982	220
5	16.0	6.0	10.999	715
6	16.0	6.0	10.928	2002
7	16.0	6.0	10.791	5005
8	14.0	6.0	10.488	11440
	Pure Multigrid			
2	13.0	5.0	10.000	10
3	20.0	6.0	12.436	55
4	21.0	7.0	13.155	220
5	24.0	8.0	13.221	715
6	22.0	8.0	13.185	2002
7	21.0	8.0	12.997	5005
8	20.0	8.0	12.762	11440

Again, the results in the table and figures shows the dependence of the number of dimensions in the error. The total time for the $5d$ computation is again relatively small per grid and the accuracy results are satisfactory for the time dependent case. The multigrid convergence remains excellent, as in the stationary test case above.

5.7 Multi-Asset Option

The last experiment is the computation of the price of a basket option by solving equation (5.4) with the sparse grid technique. The corresponding initial condition reads:

$$u(\boldsymbol{x}, 0) = \max\left(\sum_{i=1}^{d} w_i e^{\sigma_i x_i} - K, 0\right), \tag{5.16}$$

with w_i percentages of the assets in the underlying basket, σ_i the volatility in asset i and K the exercise price. This payoff function has a non-differentiability in the hyperplane when the basket sum equals the strike price. This will be problematic for the sparse grid solution of this problem, as one requirement for sparse grid convergence is that numerical solutions have bounded mixed derivatives [41]. Still we are interested in the convergence of sparse grids for this option pricing problem.

The remaining problem parameters are set for each asset as

Table 5.3: Time dependent experiments with solution (5.15) using sparse grids. TOP: is the $2d$ case. BOTTOM: $5d$ case. Column one gives the maximum number of cells per coordinate.

	$X = 0.5^d$			Control		
Grid	Value	Error	Conv	Time	#probl	Th. Conv
d=2						
8	2.24	$1.37 \cdot 10^{-4}$	2.96	2.14 sec.	5	2.67
16	2.24	$4.36 \cdot 10^{-5}$	3.14	5.67 sec.	7	3.00
32	2.24	$1.33 \cdot 10^{-5}$	3.28	10.87 sec.	9	3.20
64	2.24	$3.93 \cdot 10^{-6}$	3.38	19.12 sec.	11	3.33
128	2.24	$1.14 \cdot 10^{-6}$	3.46	31.53 sec.	13	3.43
256	2.24	$3.23 \cdot 10^{-7}$	3.52	49.01 sec.	15	3.50
512	2.24	$9.07 \cdot 10^{-8}$	3.56	1m 10s	17	3.56
1024	2.24	$2.56 \cdot 10^{-8}$	3.54	1m 40s	19	3.60
d=5						
8	3.38	$9.67 \cdot 10^{-5}$	1.98	9.02 sec.	21	0.79
16	3.38	$4.45 \cdot 10^{-5}$	2.17	36.99 sec.	56	1.27
32	3.38	$1.92 \cdot 10^{-5}$	2.32	2m 10s	126	1.64
64	3.38	$7.83 \cdot 10^{-6}$	2.45	7m 17s	251	1.93
128	3.38	$3.06 \cdot 10^{-6}$	2.56	21m 28s	456	2.16
256	3.38	$1.15 \cdot 10^{-6}$	2.65	1h 58s	771	2.34
512	3.38	$4.24 \cdot 10^{-7}$	2.72	3h 8m 18s	1231	2.50
1024	3.38	$1.50 \cdot 10^{-7}$	2.83	8h 57m 46s	1876	2.62

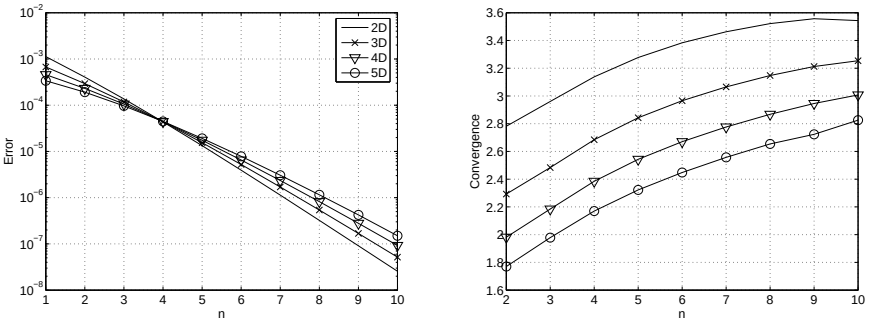


Figure 5.8: LEFT: Decay of the error $|u_n^c - u_E|$, u_E from (5.15). RIGHT: Convergence of the error in the left picture.

Table 5.4: Option prices of basket calls. TOP: Sparse grid option prices for $d = 2$ and $d = 3$. BOTTOM: Option prices for the higher dimensions. n represents the maximum number of point in one dimension

	d=2		d=3	
n	Value	Error	Value	Error
8	2.291	$5.11 \cdot 10^{-1}$	2.102	$4.51 \cdot 10^{-1}$
16	2.727	$7.60 \cdot 10^{-2}$	2.498	$5.46 \cdot 10^{-2}$
32	2.801	$1.10 \cdot 10^{-3}$	2.562	$1.00 \cdot 10^{-2}$
64	2.807	$4.23 \cdot 10^{-3}$	2.563	$9.04 \cdot 10^{-3}$
128	2.811	$8.56 \cdot 10^{-3}$	2.562	$9.87 \cdot 10^{-3}$
256	2.807	$4.65 \cdot 10^{-3}$	2.555	$3.28 \cdot 10^{-3}$
512	2.803	$7.72 \cdot 10^{-4}$	2.554	$1.39 \cdot 10^{-3}$
1024	2.803	$6.52 \cdot 10^{-4}$	2.553	$3.61 \cdot 10^{-4}$
	d=4		d=5	
8	1.983	$4.32 \cdot 10^{-1}$	1.896	$4.32 \cdot 10^{-1}$
16	2.364	$5.05 \cdot 10^{-2}$	2.273	$5.42 \cdot 10^{-2}$
32	2.429	$1.45 \cdot 10^{-2}$	2.343	$1.57 \cdot 10^{-2}$
64	2.427	$1.19 \cdot 10^{-2}$	2.341	$1.36 \cdot 10^{-2}$
128	2.422	$7.02 \cdot 10^{-3}$	2.331	$3.97 \cdot 10^{-3}$
256	2.420	$5.35 \cdot 10^{-3}$	2.335	$7.88 \cdot 10^{-3}$
512	2.417	$2.69 \cdot 10^{-3}$	2.330	$2.28 \cdot 10^{-3}$
1024	2.413	$1.29 \cdot 10^{-3}$	2.326	$1.77 \cdot 10^{-3}$

- $w_i = 1/d$, $1 \le i \le d$
- $K =$ € 40,
- $r = 6\%$,
- $T =$ one year,
- $\sigma_i = 20\%$, $1 \le i \le d$
- $\delta_i = 4\%$, $1 \le i \le d$
- $\rho_{ij} = 0.25$, $i \ne j$.

The price of the option is computed for $2 \leqslant d \leqslant 5$ where d represents the number of assets in the basket. The outer domain boundaries are placed at $S = 5K$ to mimic infinity in (5.1). In the $x-$domain, this means that $\Omega = [-\sigma_i^{-1} \log 5, \sigma_i^{-1} \log 5]^d$. The sparse grid approximation contains grids with at most 1024 cells per coordinate and with 128 time steps. The results of the experiments are summarized in Table 5.4.

In the table, satisfactory grid convergence is observed for the lower-dimensional cases, but it is no longer regular. In particular when d is increasing, the convergence becomes irregular. The reason may lie in the fact that in higher dimensions a large number of subgrids is included with only a very small number of grid points in many

dimensions ($n_i = 2$ in this test). An alternative is to use the sparse grid technique based on a larger number of points in each dimension (n_i at least 4 or 8) see, for example, [51]. Furthermore, the accuracy is hampered by the fact that the initial condition is non-differentiable.

The multigrid convergence, however, remains excellent.

Chapter 6

A geometric multigrid method for PDEs on stretched grids

We now switch themes in terms of application and shift our focus to efficient multigrid methods for logarithmically stretched grids. The emphasis is on geometric L-shaped coarsening techniques that we have developed in this context. The presented method is matrix free, in contrast with alternatives such as Algebraic Multigrid (AMG) or certain preconditioned Krylov subspace based solution methods. For a Poisson model problem, we explain, both visually and in a descriptive way, how the stretched fine grid may yield a sequence of coarser grids so as to maintain the complementarity between relaxation and coarse grid correction. We also present complexity estimates of the method, thus demonstrating its efficiency. Through figures and numerical experiment tables, we provide convergence histories for the model problem discretized -and solved- on various stretched grids with our method.

6.1 Introduction

Many real life application problems have discontinuities or kinks in specific regions of the domain and, therefore, the selection of a discretization grid is usually dictated by specific accuracy requirements in these regions. The regions are often restricted locally to certain parts of the domain, and require a higher concentration of grid points. This local concentration of grid points gives rise to anisotropy in the resulting linear system. As discussed in the preceding chapters, anisotropy -due to strongly varying connection strengths in the discrete operator- poses well known convergence problems for standard multigrid with pointwise smoothing [8, 52]. The well known remedial is to smooth -or coarsen, in such a way so that complementarity between the smoothing process and the coarse grid correction process is preserved. The former approach is well-developed and precisely known [8, 6], although its practical use is not very viable when problem dimensionality increases. The latter approach is open to development and numerous works have surfaced in this context, see [26, 27, 28, 53].

Higher grid resolution may be obtained from two geometrically different ideas, *AMR* (*Adaptive Mesh Refinement*) [54, 55, 56, 57] and *Grid Stretching*; both approaches may give rise to *structured* and *unstructured* grids. For geometric classification, we refer the grids arising from locally adapted mesh refinement as *amr-type* grids, while those resulting from the use of a global stretching parameter (detailed in Section 6.2.3) as *str-type* grids. While structured *str-type* grids form the main theme in this chapter, structured *amr-type* grids with a specific local refinement topology, are the subject of Chapter 7. Solution methods for unstructured grids usually make use of *quad* -or *oct* tree data structures [58, 59]. *Uniform* grid stretching occurs when mesh sizes are equidistant throughout a particular dimension but are non-equidistant across different dimensions. On the contrary, *non-uniform* grid stretching can be defined as the case where the grid has variable mesh sizes even within a single dimension. These grids are often the result of a coordinate transform of the grid variables.

In Chapter 4, we showed that for uniformly stretched multi-dimensional grids, partial coarsening along the stretched dimensions gave an optimal multigrid algorithm. One of the important inferences from the local Fourier analysis in that theme of work, was that if the dimensions of the grid (or equivalently, grid cells) stayed within a factor of 1.3 of each other, they could all be coarsened together. Beyond this factor, coarsening must only be done with respect to the highly elongated dimension(s). The reason is that the Fourier smoothing factor of the relaxation method deteriorates significantly, if standard coarsening is performed while one (or more) grid dimensions is out of proportion. In this chapter, we see that this rule can be applied successfully to grid cells on an individual basis as well. On the individual basis, this rule translates to a similar bound on the *mesh aspect ratio* of the cell. Based on this, we introduce a novel grid coarsening method (for $2d$ stretched grids), and perform numerical tests on a Poisson-type model problem. The specific type of grid stretching handled in this chapter is called, the *Power-law* grid stretching, and belongs to the family of *logarithmic stretching*.

A well known alternative to geometric multigrid treatment of grid stretching is the use of algebraic multigrid (*AMG*) [3, 60]. AMG starts out with a given matrix and constructs all the multigrid components algebraically, through variational principles, during the actual solution process which thus entails a setup phase with an associated cost. In this chapter, the presented multigrid method circumvents the drawbacks of AMG while still ensuring its excellent convergence factors for PDEs on stretched grids. We introduce a hybrid technique that uses an AMG-style coarsening, in a geometric multigrid setting. Coarse grids in the proposed method are formed by agglomeration of the fine grid cells depending on how square (in shape) a particular candidate agglomeration is, depending on a particular priority criterion. If the grid is traversed in L-shaped lines, this entire process retains a nice structure and allows for an efficient implementation of the process; which is how it gets the name *L-shaped coarsening*. We combine L-shaped coarsening with point smoothing, piecewise constant restriction, bilinear interpolation, and present a geometric method that converges very well for the Poisson model problem.

An outline of the chapter is as follows. In Section 6.2, we specify the model problem, the Power-law grid stretching scheme, and the cell centered finite volume discretization that we use. Section 6.3 follows, with details of the geometric multigrid

components for the new method. Here, we explain in precise detail the L-shaped grid coarsening technique, along with a visual display of the coarsened grids obtained. The discretization coarse grid (*DCG*) operator comes next, and is followed by the description of the transfer operators. In Section 6.4, we do a complexity analysis for this method. This is followed by Section 6.5, in which we provide numerical experiments based on 1d and 2d grid stretching. A simple jump discontinuity experiment and an experiment on *amr-type* grids are also performed to test the robustness of the method. Convergence histories are presented both in tabular and in visual displays; finally, in the last section, some conclusions are drawn from the work.

6.2 Cell Centered FVM, and Grid Stretching

6.2.1 The Model Problem

The $2d$ Poisson type equation (on a unit square, with Dirichlet boundary conditions) is given as:

$$\begin{aligned} -\nabla \cdot \mathbf{A}\nabla u(x,y) &= f^{\Omega}(x,y), \quad (x,y) \in \Omega = (0,1)\times(0,1) \\ u(x,y) &= f^{\Gamma}(x,y), \quad (x,y) \in S \Rightarrow x \in \{0,1\} \text{ or } y \in \{0,1\} \end{aligned} \tag{6.1}$$

where

$$\mathbf{A} = \begin{bmatrix} a_1(x,y) & 0 \\ 0 & a_2(x,y) \end{bmatrix}$$

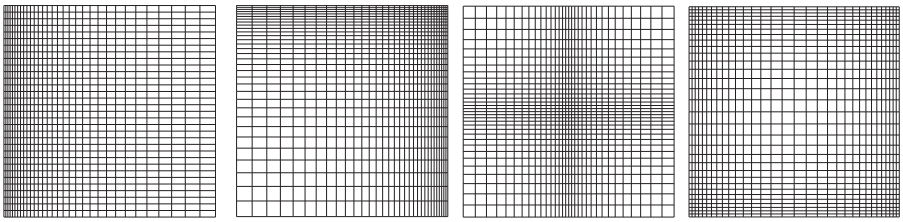
and a_1 and a_2 are smoothly varying functions.

6.2.2 The Cell Centered Finite Volume Scheme

To define a cell centered finite volume scheme, the entire domain is divided into rectangular control volumes, with nodes in the center of the cells. Each node represents the value of the unknown averaged over the control volume. Integrating both sides of Equation 6.1, and then applying the Gauss Divergence theorem, results in an integral over the boundary of the domain. After division of the domain into control volumes, the so-called flux balance equation per finite volume can be immediately deduced as:

$$-\sum_{k=0}^{3} \int_{S_{m_k}} (\mathbf{A}\nabla u_{m_k}) \cdot \hat{n}_k \, dS_{m_k} = \int_{\Omega_m} f^{\Omega} d\Omega_m \tag{6.2}$$

where m indexes a rectangular control volume, S_{m_k} denotes the k^{th} face of the boundary of control volume m, $\hat{n}_k$ is the outward unit normal from this face, u_m is the value of u averaged over the control volume, m, and (∇u_{m_k}) refers to the gradient of u_m computed at the midpoint of S_{m_k}. Ω_m is the volume of the m^{th} cell, such that $\Omega = \cup\, \Omega_m$. Equation 6.2 is the foundation equation of FVM here, which can be approximated in a number of ways, one of which is detailed in Section 6.3. When this is done for all values of m, the result is an $(m \times m)$ system of linear equations.



(a) West edge (b) North east corner (c) Center (d) All corners

Figure 6.1: (*str-type* grids): A depiction of different *str-type* concentrations of control volumes, obtained through Equation 6.3 with $\alpha = 1.07$

6.2.3 Grid Stretching and the Power-law Scheme

In this chapter, we treat multigrid for a specific variety of *str-type* grid stretching, called the Power-law grid stretching. In what follows, we use the terms *left* and *right* to indicate the decreasing and the increasing directions -respectively- along a particular dimension. x_{str} is the point from where the stretching ensues. Dirichlet boundaries are pinned down first, and then we divide the segment to the left and the right of the stretching point into a specified number of control volumes. The mesh sizes along each dimension are generated by the 1-dimensional formula presented in Equation 6.3. In the following, x_{min}, x_{max} are the domain boundaries along the i^{th} dimension, so that $x_{min} \leq x_{str} \leq x_{max}$ holds. N_L and N_R are the specified number of cells on the left and the right of x_{str}, respectively. Similarly α_L and α_R are the stretching parameters -to be used- respectively on the left and the right of x_{str}. When this is provided, we choose:

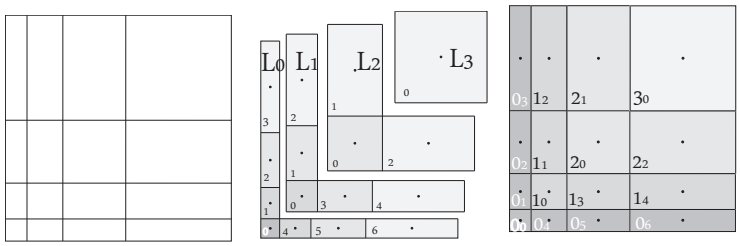
$$\begin{aligned} h_k &= \left\{ (x_{str} - x_{min}) / \left(\frac{\alpha_L^{N_L} - 1}{\alpha_L - 1} \right) \right\} \times \alpha_L^{(N_L - 1 - k)}, \quad k = 0 \cdots (N_L - 1); \\ h_k &= \left\{ (x_{max} - x_{str}) / \left(\frac{\alpha_R^{N_R} - 1}{\alpha_R - 1} \right) \right\} \times \alpha_R^{(k - N_L)}, \quad k = N_L \cdots (N_L + N_R - 1); \end{aligned} \tag{6.3}$$

Using this, we can generate specific stretched grids, some of which are depicted in Figure 6.1. In Section 6.5, we experiment both with $1d$ and $2d$ grid stretching. For $2d$ stretching, the method that we present in this chapter has been developed for grids having control volume concentration at a domain corner. However, we point out that the other $2d$ stretchings (i.e., those having concentrations either at the center or at all the four corners of the domain) are merely unions of the types that we treat here and, therefore, the results carry over to them as well.

6.3 L-shaped Coarsening and Multigrid Components

6.3.1 The Enumeration Scheme and L-shaped Coarsening

We focus on domains that are stretched with the same parameter for both x and y dimensions (i.e., $\alpha = \alpha_x = \alpha_y$), however, some general settings can also be easily



(a) A 4×4 stretched grid (b) Display of L-strips (c) Control volume ordering

Figure 6.2: Grid enumeration in L-shaped strips

dealt with. This allows us to use symmetry advantageously for storage purposes. The entire domain is divided into L-shaped control-volume strips. In general, an L-strip consists of a *vertex-cell*, a *vertical segment* consisting of control volumes, and likewise, a *horizontal segment*. The different L-strips in the discrete domain have different number of cells, each of which has a different volume. The different L-strips are ordered in priority of cell density, so that the most densely saturated strip comes first, and the strips with descending saturation of control-volumes follow. The last strip invariably consists of only the vertex control volume, being devoid of horizontal -and vertical segments.

Example 6.3.1. (Enumeration of the L-strips and ordering of the cells) *Consider Figure 6.2. (a) shows a simple stretched Cartesian grid with* 4×4 *control volumes. In (b), these control volumes are enumerated into the* 4 *L-strips,* $L_0 \cdots L_3$*, and nodes are placed in the center of each control volume. The disassembly is virtual and only shown as a depiction of the enumeration. Note the ordering of the control volumes. Each L-strip complies with the general description in this section. (c) shows the same grid as in (a) but with the grid enumerated in L-strips and control volumes carrying the described ordering. In the cell-indices in (c), the larger digit represents the index of the L-strip, while the smaller ones represent the indices of the control volumes within it.*

Grid coarsening in this setting is performed by agglomeration of control volumes on the fine grid [61, 62]. Selection of prospective fine grid cells to agglomerate is done by virtually isolating two L-lines and comparing the different mesh aspect ratios that we seek to improve through the coarsening process. The mesh aspect ratio, *mar*, of a cell is defined as:

$$mar = \frac{h}{w},$$

where h and w represents the height and the width of the cell, respectively.

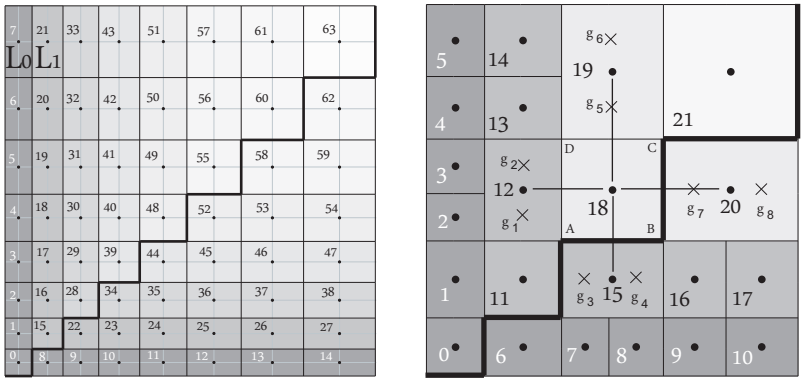
The global guiding principle in selecting fine grid cells to agglomerate is that the newly constructed coarse grid cell should reflect an improvement in the mesh-aspect ratio over other prospective fine cell agglomerations. A pair of L-strips is virtually isolated and inspected cell by cell. The process is guided by the enumeration scheme of

our method. Coarsening starts by agglomerating the vertex cells. This agglomeration always gives a perfectly square coarse cell. Then the vertical segment is inspected, 4 cells at a time, i.e., 2 cells of each adjacent vertical strip; and a choice is made between horizontal-semi-coarsening (2×1) or full-coarsening (2×2). This choice is not solely dependent on the mesh aspect ratios; in fact, it is *biased* in favour of full-coarsening if this comes within a certain threshold. The decisions are stored and automatically carried over to the horizontal segment due to symmetry. A pair of fine grid L-strips thus gives a coarse grid L-strip. This process of inspecting fine grid L-strips in pairs is continued until the grid is depleted. It is important to note that this particular coarsening process leads us to store a *connection structure* which has exactly half the number of elements as the control volumes in the given grid. The following example is provided to elaborate on this process in greater detail.

Example 6.3.2. (L-shaped coarsening) *Consider the grids in Figure 6.3. The grid in Figure 6.3(b) represents the coarse grid chosen for the fine grid in Figure 6.3(a). To illustrate the process, we inspect L_0 and L_1 together, in Figure 6.3(a). First, we agglomerate the vertex cells,* $0, 1, 8, 15$*, and then move up the vertical segment. Two different prospective agglomerations to consider next are either combining Cells* $2, 3, 16, 17$ *together, called prospect 1, or combining Cells* 2 *and* 16*, called prospect 2. The coarsening pattern that we employ is such that individually agglomerating Cell* 2 *with Cell* 3 *and Cell* 16 *with Cell* 17 *is not an option, nor is not agglomerating at all. Put simply, we rule that while traversing the vertical segment, the decision has to be made between semicoarsening in the x-direction or full-coarsening, depending on mar_1 and mar_2, which are the mesh aspect ratios of the two prospects, respectively. This ensures that nodes stay aligned along* 1 *dimension and reduces unnecessary book keeping. We define the difference, d_i, for the i^{th} coarsening prospect as $d_i = |1 - mar_i|$. Although it seems relatively simple to pick the prospect with the smaller difference, we point out that this does not lead to an optimal reduction of complexity. As mentioned earlier, we set a priority criterion in favour of prospect 1, as it allows a greater reduction of unknowns, compared with prospect 2. We use a threshold value, such that if d_1 is less than -or equal to it, then prospect 1 is selected and prospect 2 dropped, even if $d_2 < d_1$; however, if comparison with the threshold fails, then the prospect with the lesser d is selected straight away. On the coarser grids, there is an additional check; if a situation such as depicted by Cells* $2, 3, 12$ *of Figure 6.3(b) arises, then we simply carry out this 3-cell agglomeration and do not venture to fatten the prospective coarse grid cell any further. In particular, this check ensures that the boundary of the coarsened cell is shared with its neighbour in the adjacent L-strip.*

On the finest grid, the coarsening process detailed above results automatically in full-coarsening along a particular *diagonal* portion of the grid and in semi-coarsening as the proximity to the domain edges grow. All coarser grids add a band of (full-coarsened) cells to the right and the left of this diagonal portion, adding up to complexity reduction. Figure 6.4 gives the complete sequence of coarse grids generated for a 64^2 fine grid stretched with $\alpha = 1.03$. In Section 6.5, we solve the model problem on this sequence of grids and demonstrate the convergence of the resulting multigrid solver.

Remark 6.3.1. (Structure on coarse grids) *We would like to point out that the coarse*



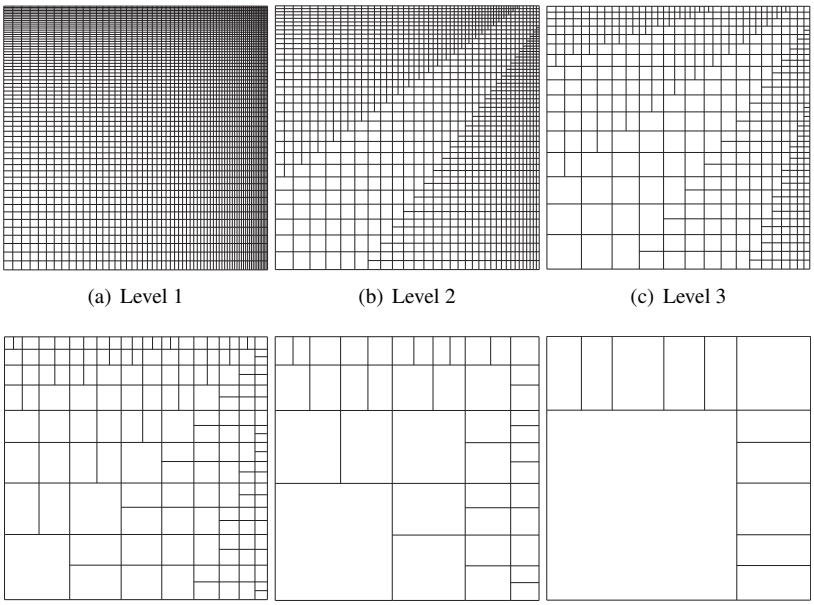
(a) (b)

Figure 6.3: (a) shows an (8×8) grid concentrated at the lower-left corner. The stretching parameter used along both dimensions is $\alpha = 1.07$ and $x_{str} = 0.0$. Thin lines connecting the nodes emphasize that -unlike coarser levels- all nodes on the finest grid are aligned horizontally and vertically (there are no hanging nodes). The heavy stair-case line shows the symmetry of the grid which is advantageous both from a storage as well as a computational point of view. (b) displays the first coarse grid constructed from (a) using the technique described in this section. Note that there may be hanging nodes on the coarser levels.

grids are structured along L-shaped strips, within which they are aligned horizontally and vertically. This is deliberate, in order to keep data access optimal and the implementation geometric. A multigrid method based on similar algebraic rules of coarsening, but without geometric constraints would be non-optimal (due to non-optimal data access) if implemented without storing matrices.

Remark 6.3.2. (Aspect-ratio bound for good multigrid convergence) *We would like to highlight that the aspect ratio of a particular cell is tolerable up to a value of 1.3 from the point of view of multigrid convergence [29] with pointwise smoothing, i.e., LFA smoothing factors do not deteriorate if dimensions have a size disparity within this range, and are coarsened together. Seeking the value 1.0 for prospective agglomerations is idealistic and impractical, often leading to coarsening choices that imply very poor reduction of unknowns per grid level. This observation leads us to use a threshold value of 0.3.*

Remark 6.3.3. (Nodal position on the coarsened grid) *After the coarsening process is over and the coarse grid is constructed, we place the nodes (i.e., unknowns) in the center of the coarse grid control volumes.*



(a) Level 1 (b) Level 2 (c) Level 3

(d) Level 4 (e) Level 5 (f) Level 6

Figure 6.4: The complete sequence of coarse grids obtained by coarsening a 64^2 grid stretched with $\alpha = 1.03$, by the described algorithm. Visually, the coarse grids seem unstructured but in fact they are aligned vertically and horizontally within each L-strip.

6.3.2 The DCG Operator and Pointwise relaxation

The DCG Operator

On the coarse grids, we have a matrix free discretization coarse grid *DCG* operator, which is a cheaper option compared to matrix based methods. The discretization scheme on the coarse grids is the same as that on the finest grid, i.e. the cell centered FVM. Consider Figure 6.3(b); if the m^{th} control volume is defined by the rectangle $\overline{ABCD}$, then;

$$\int_A^B a_2 \frac{\partial u_m}{\partial y} dx_m - \int_B^C a_1 \frac{\partial u_m}{\partial x} dy_m - \int_C^D a_2 \frac{\partial u_m}{\partial y} dx_m + \int_D^A a_1 \frac{\partial u_m}{\partial x} dy_m = \int_{\Omega_m} f^{\Omega} d\Omega_m \tag{6.4}$$

follows directly from Equation 6.2. The right hand side of Equation 6.4 can be approximated simply by the point value of the source function multiplied by the volume of the cell. Each first order derivative on the left hand side is approximated by the central $O(h^2)$ FDM if the particular k^{th} face of the control-volume boundary is not a portion of the domain boundary. However, if it is, then the derivative is approximated by the $O(h)$ one-sided FDM. For a uniform equidistant grid layout, this scheme results in second order accuracy [63].

For the finest grid, it is important to point out that the discretization of the deriva-

tives in Equation 6.4 -being through finite differences- is trivial due to the perfect alignment of the nodes with their horizontal and vertical neighbours. The only general guiding principle for constructing a successful DCG operator on the coarser grids is the conservation of flux through each face of the control volume. For an example that illustrates how the conservation of flux is actually maintained during computations, consult Appendix D. We define the flux as the *net flow* through a particular face of the cell; for example, the flux through the east face of the m^{th} control volume is:

$$F_m^{east} = \int_B^C a_1 \frac{\partial u_m}{\partial x} dy_m$$

Example 6.3.3. (coarse grid stencil for the hanging nodes) *In Figure 6.3(b), Nodes 2 and 3 do not have horizontal neighbours and, therefore, make use of ghost points g_1 and g_2, respectively, which, in turn, are linearly interpolated from the points directly above and beneath them. In effect, this means that (on the east side) Node 2 is connected with Nodes 11 and 12, while Node 3 is connected with Nodes 12 and 13. The same applies to Nodes 7 and 8 in the horizontal segment. As a general rule, whenever a node is missing, we linearly interpolate it from its collinear neighbours.*

Remark 6.3.4. (Alternative flux definition) *In contrast to our definition of the flux as the net flow through a boundary, flux is also often defined as the rate of flow, i.e., without incorporating the length of the boundary segment (dy_m in this case) in the definition. This is useful where flux has to be averaged across control volumes such as might be encountered in an AMR setting [8]. In our work, however, defining flux as the net flow is more helpful. Mathematically the two definitions are equally acceptable.*

Example 6.3.4. (Conservation of flux) *For Nodes 2, 3, and 12 of Figure 6.3(b), conservation of flux would mean that the following equality holds:*

$$F_{12}^{west} = -\left(F_2^{east} + F_3^{east}\right)$$

Pointwise relaxation

The relaxation process in our multigrid method is a variant of the lexicographical point-based Gauss-Seidel relaxation method. The variation is only in terms of the pattern in which the domain is traversed. The smoothing properties of a stationary iterative method, such as Gauss-Seidel, are not invariant of the relaxation pattern in which the unknowns are visited. The traversal pattern in this work, trivially, is in L-shaped strips following the enumeration of the domain in these structures, and the ordering of the control volumes within them. Each L-strip is visited in the enumeration order. Within each strip, first the node in the vertex control volume is relaxed, then each of the nodes (in enumeration order) in the control volumes in the vertical segment, and finally the nodes in the horizontal segment of an L-strip. The observed smoothing properties are superior to those of classical lexicographic Gauss-Seidel and slightly inferior to Red-Black Gauss Seidel; however, they are sufficiently good to provide excellent multigrid convergence in this set-up for Poisson's equation.

6.3.3 The Transfer Operators

The Restriction Operator

We use piecewise constant restriction to transfer grid functions from finer grid levels to coarser grid levels. In our cell centered setting, due to variable mesh sizes, this requires averaging across cells that are different in volume. Therefore, we use a volume-average based restriction, which is a generalization of the piecewise constant FP restriction in Chapter 2, Description 2.3.9.

Here, we explain the method through which fine grid cells are restricted to coarse grid cells. Let m_f denote a subset of indices of control volumes of the finer level, which would be agglomerated to form the control volume m_c, after the coarsening has taken place; i.e., $\Omega_{m_c} = \sum \Omega_i,\ i \in m_f$. v_i represents any grid function that has to be averaged, such as the residual.

Nodes contained in the subset m_f of fine level l contain values averaged over their respective control volumes, i.e.:

$$v_i = \frac{\int_{\Omega_i} v \, d\Omega_i}{|\Omega_i|}, \qquad i \in m_f \tag{6.5}$$

likewise, the coarse node m_c, should contain a value representing the average over the control volume m_c of level $(l+1)$,

$$v_{m_c} = \frac{\int_{\Omega_{m_c}} v \, d\Omega_{m_c}}{|\Omega_{m_c}|} \tag{6.6}$$

which gives:

$$v_{m_c} = \left(\sum_{\Omega_i, i \in m_c} \int_{\Omega_i} v \, d\Omega_i \right) / |\Omega_{m_c}| \tag{6.7}$$

$$\Rightarrow v_{m_c} = \left(\sum_{i \in m_f} v_i \, |\Omega_i| \right) / |\Omega_{m_c}| \tag{6.8}$$

This averaging formula contains a contribution from each of the fine grid cell in the subset m_f, respective to its volume. The cumulative contribution is then distributed over the coarse grid cell volume, to represent an average value within it. All fine to coarse transfers in this chapter make use of this restriction.

The Prolongation Operator

We use simple node-position based bilinear interpolation for transferring grid functions from coarse to fine levels. From a global perspective, the nodes are ordered into horizontal and vertical segments of L-strips. Each fine grid horizontal or vertical segment has a similarly oriented coarse grid segment to its left and right. In general, the nodes on these left and right neighbours are not aligned with the fine grid nodes and, therefore, first have to produce linearly interpolated values which are collinear with the fine grid nodes. After these values have been interpolated in one dimension,

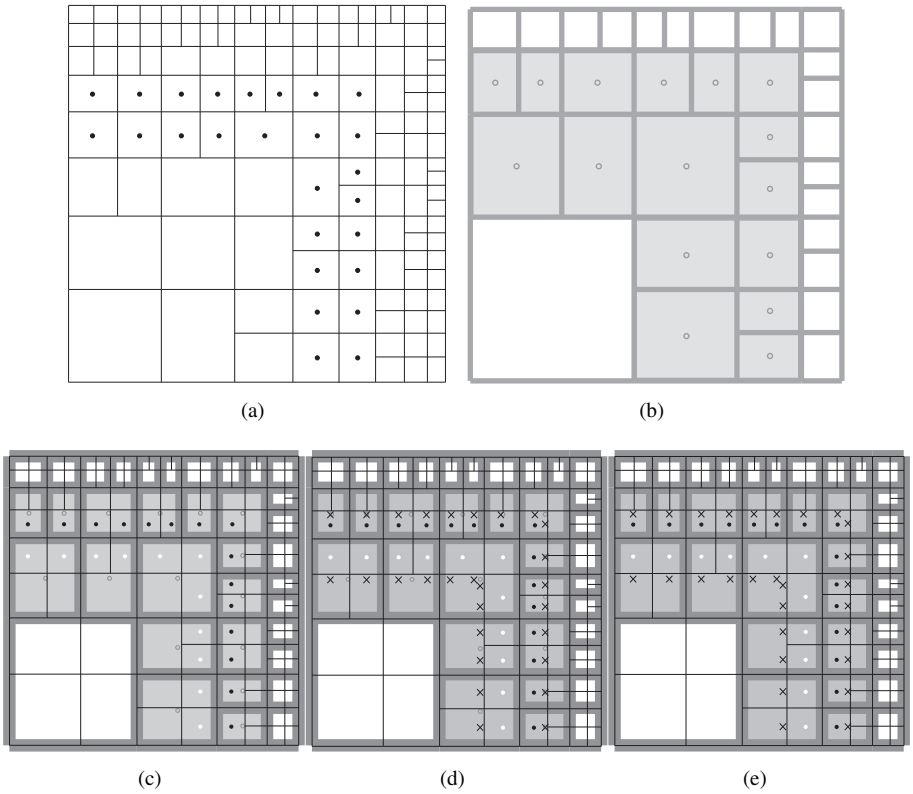


Figure 6.5: (a) and (b) represent the third and the fourth levels respectively of a 32^2 grid stretched with $\alpha = 1.06$, respectively. In (c), (d), and (e), these levels are superimposed to demonstrate the prolongation for level 3 from level 4. Each fine grid node is interpolated by information from all 4 surrounding sides.

the fine grid nodes are subsequently interpolated from them. The interpolation (analogous to restriction) is based on the actual relative distancing of the nodes, and not on the fixed component prolongation stencils. This process ensures that each fine grid node is interpolated with 4 coarse nodes (from all four surrounding sides), and that the interpolation process is never uni-directional.

Example 6.3.5. (Interpolating a fine grid L-strip) *Consider Figure 6.5. Grid-levels* 3 *and* 4 *of a particular grid sequence are superimposed to elaborate on how bilinear interpolation takes place in the L-shaped setting. The coarse grid is represented by thick grey lines, and the fine grid by fine black lines. L_1 and L_2 strips of the coarse grid are shown shaded along with their nodes, represented by hollow grey circles. These coarse grid L-strips enclose L_3 and L_4 strips of the fine grid, containing solid black and solid white -shaded nodes, respectively. In the figure, the interpolation for the black nodes is demonstrated; white nodes are only there to emphasize that they too would have to be interpolated from the same set of coarse nodes. The black crosses*

represent ghost positions, collinear both with the coarse grid nodes and (black) fine grid nodes. The interpolation takes place in two stages. Proceeding from (c), first the crosses are interpolated linearly from the coarse grid nodes (hollow grey circles). Once these ghost points are populated, the original coarse grid nodes have no role left. They are deliberately not shown in (e) to depict this. At this stage, the black fine grid nodes are finally interpolated linearly from the crosses with which they are collinear. This scheme gives bilinear accuracy. Note that interpolation of the solid white fine grid nodes would employ the same coarse grid nodes, but the position of the crosses would change and reflect collinearity with the white nodes.

The interpolation coefficients are only computed for the vertical segment and are retained for use with the horizontal segment to take advantage of symmetry. Each L-strip (and, subsequently, each control volume in it) is treated in the enumeration order. The nodal values adjacent to the domain boundary are interpolated from coarse grid values on one side and from the boundary value on the other side. This scheme turns out to be a better transfer than interpolating the boundary nodes linearly from only one side.

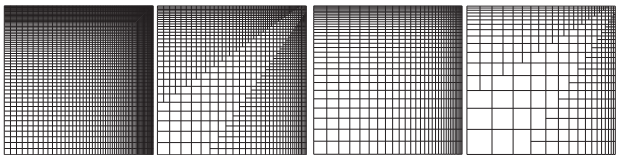
Remark 6.3.5. (Final assembly into the multigrid algorithm) *The actual solution process has a small setup phase in which all of the coarse grids are constructed and stored. The storage complexity for any grid never exceeds the number of control volumes in it. This is managed through a data structure embodying the L-shaped enumerated grid, which yields all grid parameters, including the mesh sizes, the nodal positions, symmetry information, etc., and is, therefore, indexed to point to a particular member within a family of grids. It is important to point out that the natural form of discretization of FVM or FEM (unlike FDM) has the right-hand side scaled. This implies that the residual also has the same scaling, and, therefore, must be neutralized and re-scaled both before and after restriction to the coarse grid.*

6.4 Complexity

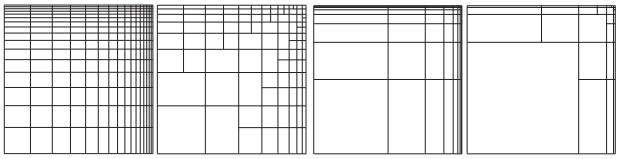
Carrying over the notation from Chapter 4, we define a work unit, *wu* as:

$$1\ wu = C\ M_0 \tag{6.9}$$

where C is a small constant. We measure the complexity of our multigrid method in terms of the computational work W_0. The basic complexity relationship given in Chapter 4, Inequality 4.18 holds, and observing the method from a 2-grid perspective helps in estimating its complexity. In Figure 6.6, the grid reduction in a 2-grid setting is displayed. The specific tie-shapes are the regions where 4-cell agglomerations take place, whereas in the darker regions close to the domain edges, semicoarsening in either direction is performed. It has already been discussed that this hybrid behaviour pays off in the form of better complexity values. In this particular figure, the grid sizes range from 8^2 to 256^2, and α ranges from 1.01 to 2.0. Although these grids appear greatly different in their layout, they have a common denominator, as they share a common measure of the worst aspect ratio (which has been deliberately brought about



(a) 64^2 grid, $\alpha = 1.04$ (b) 32^2 grid, $\alpha = 1.09$



(c) 16^2 grid, $\alpha = 1.20$ (d) 8^2 grid, $\alpha = 2.00$

Figure 6.6: A variety of fine and coarse grids with a reduction factor ≈ 2.34

by specific combinations of grid sizes and stretching parameters). In turn, all of them display a grid reduction factor around 2.34. This directly suggests that the coarsening factor depends greatly on the maximal aspect ratio. For a closed form estimate of the complexity in a 2-grid setting, we consider the first two grids. The control volumes on the fine grid are aligned both horizontally and vertically and, therefore, the cell index i runs from 0 to ($\sqrt{M_0}-1$) in both directions. The mesh-sizes are governed by Equation 6.3, and after a slight modification, are given by;

$$h_i = \frac{x_{max}}{(\frac{\alpha^{M_0}-1}{\alpha-1})}\alpha^i, \qquad i = 0, \cdots, n, \cdots, (M_0 - 1)$$

During the inspection of the vertical segment for coarsening decisions (as outlined in Section 6.3.1), the index of the last cell (on the first strip L_0) chosen for full (2×2) coarsening, is represented by n. After selecting n, it is fairly straight forward to connect the first two grids in a closed form formula.

Let c_1 denote the number of cells in the coarse grid that were obtained by agglomerating 4 cells (2×2 coarsening) of the fine grid, and c_2 denote the remaining coarse grid cells, obtained from semi-coarsening. c_1 depends on the value of n and is always found to be of the form given in Equation 6.10. The connection between the first two grids is given as;

$$\begin{aligned} c_1 &= \left\{(M_0 - n + 1)n + \sum_{i=1}^{n-1}(n-i)\right\} \\ c_2 &= 2(M_0 - 4c_1) \\ M_0 &= (4c_1 + 2c_2)M_1 \end{aligned} \tag{6.10}$$

For estimating the computational complexity of a multigrid method with a sequence of l grids, i.e., $(\Omega_0, \Omega_1, \cdots, \Omega_{l-1})$, all grids in the sequence have to be taken into account. However, in this situation, a grid reduction function, $\tau(M_0, \alpha)$, in closed form is not apparent. The relation between the fine grid level, k, and the coarse grid

Table 6.1: A closed form for $\tau(M_0, \alpha)$ is not readily available and, therefore, discrete (empirically observed) values -averaged over a sequence of l grids- are shown for a diverse combination of grid sizes and α

	α					
M_0	1.00	1.02	1.04	1.06	1.08	1.10
4^2	4	4	4	4	4	4
8^2	4	4	4	4	3.28	3.29
16^2	4	4	3.50	3.51	3.58	3.23
32^2	4	3.62	3.40	3.22	2.96	2.87
64^2	4	3.52	3.17	2.72	2.64	2.50
128^2	4	3.22	2.72	2.45	2.36	2.34
256^2	4	2.77	2.37	2.29	2.27	2.26

Table 6.2: Work estimates, W, computed from 6.12, and measured in work units, *wu*.

	α					
M_0	1.00	1.02	1.04	1.06	1.08	1.10
4^2	1.33	1.33	1.33	1.33	1.33	1.33
8^2	1.33	1.33	1.33	1.33	1.44	1.44
16^2	1.33	1.33	1.40	1.40	1.39	1.45
32^2	1.33	1.38	1.42	1.45	1.51	1.53
64^2	1.33	1.40	1.46	1.58	1.61	1.67
128^2	1.33	1.45	1.58	1.69	1.73	1.75
256^2	1.33	1.56	1.73	1.77	1.79	1.79

level, $(k+1)$, is as follows:

$$M_k = \tau(M_0, \alpha)\, M_{k+1}, \qquad k = 0, 1, 2 \cdots, l-1 \tag{6.11}$$

where τ is independent of k. Thus Equations 6.11 and 4.18 yield:

$$W_0 \leq \frac{\tau(M_0, \alpha)}{\tau(M_0, \alpha) - \gamma}\, C\, M_0 \tag{6.12}$$

which can be used to evaluate the required work units for a given discrete problem.

Values of $\tau(M_0, \alpha)$ averaged over l grids, for varying combinations of grid sizes and stretching parameters are shown in Table 6.1.

For a geometric multigrid method (with standard coarsening) on an equidistant 2d grid, the computational work-per-cycle is bounded by $\frac{4}{3}$. In contrast, a method that only employs semi-coarsening throughout the grid will yield work-per-cycle equal to 2. It is then natural to expect that the method presented here lies between these bounds, as parts of the grid undergo 2×2 agglomeration while other parts only 2×1. In Table 6.2 and Figure 6.7, we present the work estimates for V-cycles ($\gamma = 1$) obtained from substituting the averaged values of the grid reduction factor τ in Equation 6.12. The unit of measurement for these estimates is a work unit, *wu*, as defined in Equation 6.9. Work units for grid sizes ranging from 8^2 to 256^2 are shown against values of the stretching parameter α ranging from 1.0 (i.e. no stretching)

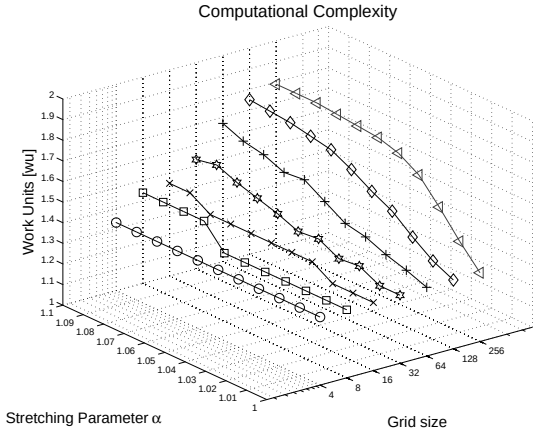


Figure 6.7: A 3d representation of the complexity of the presented method against the grid size and the stretching parameter.

to 1.1. It is well known that with semi-coarsening-only multigrid algorithms, *W*-cycle methods do not yield an optimal complexity [8]; however, we see that although our method gets relatively expensive with severe stretching (and consequently bad mesh aspect ratios), the average value of the grid reduction factor, $\tau(M_0, \alpha)$, always stays above 2.00. This is due to the fact that a semi-coarsening-only strategy is never employed when a grid is being reduced. The computational complexity grows directly with the mesh aspect ratio, which -in our case- is directly related to α as well as the grid size. Quite apparently for a 256^2 grid, stretching with $\alpha = 1.01$ and $\alpha = 1.02$ is quite sufficient for local grid refinements in practical applications. The complexity in this situation is around 1.5, whereas for an unrealistically high stretching with $\alpha = 1.1$ on 256^2 grid, the complexity is 1.79, still well under the bound set by algorithms that only semi-coarsen.

6.5 Numerical Experiments and Results

In this section, we demonstrate the presented method at work, by solving boundary value problems based on the model problem described in Section 6.2. Through the Poisson equation, we approximate the following test function:

$$u(x,\ y) = \frac{sin(2\pi^2 x) + sin(2\pi^2 y)}{(2\pi + x + y)} \tag{6.13}$$

The results include the residual decay and the achieved multigrid convergence factors. These quantities are displayed against the number of multigrid *V*-cycles required for convergence. The stopping criterion for convergence is defined by the relative residual going below the tolerance value set at 10^{-8}.

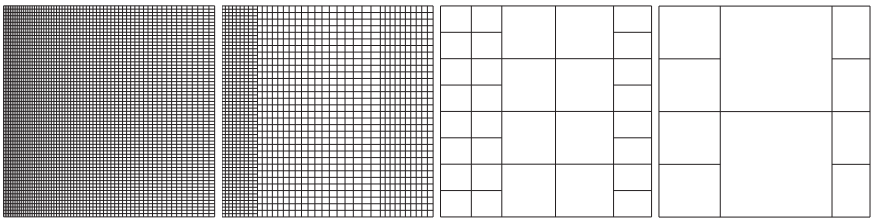
Experiments with 2 kinds of grid stretching are performed. *1-dimensional stretching*, i.e., grids stretched only along the *x*-axis and equidistant along the *y*-axis; and *2-*

Table 6.3: Multigrid convergence factors against the number of V-cycles (separated by obliques) for different values of α. Grid stretching in these experiments was only along the x-axis.

$\alpha_x \rightarrow$ Grid $\downarrow$	1.00	1.02	1.04	1.06	1.08	1.10
16^2	0.13 / 7	0.13 / 7	0.14 / 7	0.15 / 7	0.13 / 7	0.13 / 7
32^2	0.12 / 7	0.15 / 8	0.14 / 7	0.13 / 7	0.13 / 7	0.11 / 7
64^2	0.12 / 7	0.15 / 8	0.12 / 7	0.12 / 7	0.09 / 6	0.08 / 6
128^2	0.11 / 7	0.13 / 7	0.09 / 6	0.09 / 6	0.08 / 6	0.09 / 6

dimensional stretching, i.e., grids stretched along both the axes with the same stretching parameter, α. The multigrid convergence factor is estimated empirically by the contraction number q^m (see Definition 4.5.1), which depicts the average defect reduction over m cycles.

6.5.1 Experiments With 1-Dimensional Grid Stretching



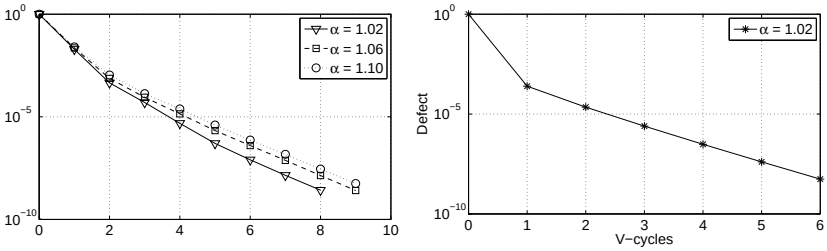
(a) Level 1 (b) Level 2 (c) Level 5 (d) Level 6

Figure 6.8: The first two and the last two grids of the sequence for a 64^2 grid stretched only along the x-axis with $\alpha = 1.01$

The results of 1-dimensional stretching are presented first. The coarsening technique, as described earlier for the 2-dimensional grid stretching, virtually isolates two vertical strips and makes decisions about the different prospective agglomerations. Figure 6.8 gives the complete grid sequence for an example problem with a 64^2 grid and $\alpha = 1.01$. The guiding principle for building the coarse grids is the same as described in Section 6.3.1; a prospective agglomeration should *relax* the *tense* aspect ratios. The only notable difference (with the 2-dimensional grid stretching case) is that the control-volume strips are vertical and not L-shaped. The rest of the process is similar. Multigrid convergence factors for a variety of stretching parameters, α, are displayed in Table 6.3.

Table 6.4: Multigrid convergence factors and the number of $V(1,1)$-cycles (separated by obliques) for different values of α. These experiments are based on 2-dimensional grid stretching

$\alpha \rightarrow$ Grid $\downarrow$	1.02	1.04	1.06	1.08	1.10
8^2	0.13 / 6	0.13 / 7	0.14 / 7	0.14 / 7	0.14 / 7
16^2	0.17 / 8	0.17 / 8	0.17 / 7	0.15 / 7	0.15 / 7
32^2	0.18 / 8	0.16 / 7	0.16 / 7	0.15 / 7	0.14 / 7
64^2	0.15 / 7	0.14 / 7	0.12 / 6	0.11 / 6	0.11 / 6
128^2	0.15 / 7	0.12 / 6	0.12 / 7	0.15 / 7	0.17 / 8
256^2	0.15 / 8	0.13 / 8	0.17 / 9	0.18 / 9	0.19 / 9



(a) *L − shaped* grid (b) *AMG* grid

Figure 6.9: Residual decay for two-dimensional stretching, against the number of V cycles employed. Figure (a) represents the decay achieved for different α with the L-shaped coarsening scheme. Figure (b) represents the same solution achieved by AMG.

6.5.2 Experiments With 2-Dimensional Grid Stretching

The Model Problem

The results of experiments with 2-dimensional grid stretching are shown in Table 6.4. The multigrid convergence factors are quite satisfactory for different α-values giving a range of moderate to severe stretchings. The residual decay for 2-dimensional grid stretching experiments are presented in Figure 6.9. Besides good convergence, all of the experiments display a good linear reduction of the residual (measured in the discrete $L2$-norm), around an order of magnitude per V-cycle.

Remark 6.5.1. (On accuracy, timing and comparison with AMG) *We verified both the accuracy of the discretization scheme as well as of this multigrid scheme in approximating the exact discrete solution, to be quadratic. The next important test performed was measuring CPU time, for a problem on a 256^2 grid (and $\alpha = 1.02$) solved both with the proposed method as well as AMG with standard components. AMG reports*

0.41 seconds and our method, 0.5 seconds; which is satisfactory considering the low storage cost of the proposed method.

Remark 6.5.2. (Stress test) *Figure 6.10 shows a kind of stress-test. A* 32^2 *grid is stretched with* $\alpha = 5.0$*, and the model problem is solved on this grid with the presented method for the purpose of checking if the method withstands such severe grid stretching. The residual decay is, however, exceptional, around* 20 *orders of magnitude in* 25 *iterations, confirming that the method does not break down.*

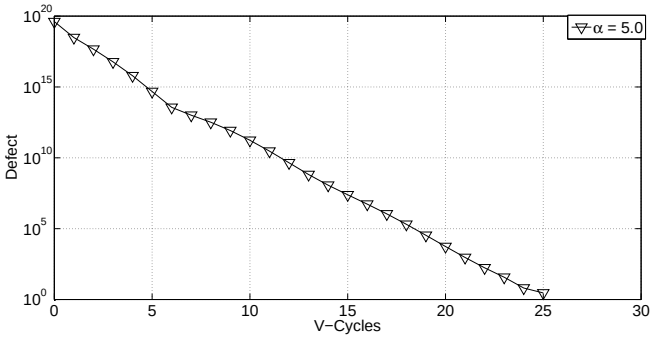


Figure 6.10: Residual decay for 2-dimensional stretching, against the number of V(1,1) cycles employed. Note the particularly high stretching for the 32^2 grid

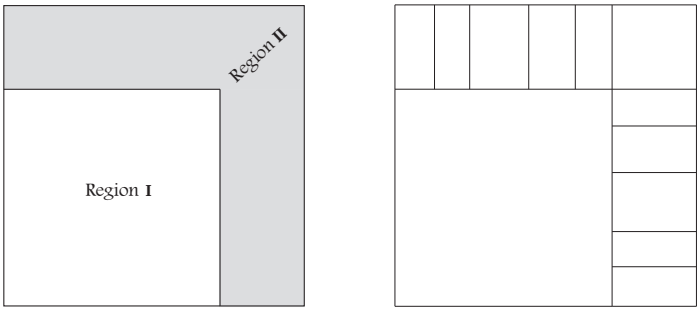
The Model Problem With a Simple Jump Discontinuity

In this part, we perturb the model problem slightly, by dividing the domain into two parts, as shown in Figure 6.11(a). The domain is partitioned into an L-shaped subdomain, and the remaining square portion, as demarked by the interface line (solid L-line in Figure 6.11(a)); so that the Poisson-type equation has different constant coefficients in the different parts of the domain but the same analytic solution everywhere. From Equation 6.1;

$$a_1 = a_2 = d_1, \qquad (x, y) \in \text{Region I}$$
$$a_1 = a_2 = d_2, \qquad (x, y) \in \text{Region II}$$

The interface line is used in forming the coarsest grid L-strip, which thereby ensures that the interface line never gets annihilated during the construction of coarse grids. As a consequence, prolongation and restriction across this bounding line (discontinuity) takes place at each grid level, moreover, we stay with the prolongation and restriction described in Section 6.3.3 and do not use operator-dependent transfer operators; neither do we need the Galerkin coarse grid operator. Table 6.5 shows the $V(1, 1)$-cycle results of the presented method when tested with this problem.

We see that the method performs fairly well even with severe jumps across the interface line.



(a) Domain divided into a square, and an L-shaped sub-domains

(b) The coarsest grid L-strip is constructed with the interface line

Figure 6.11: The subdomains used in the test with a jump discontinuity

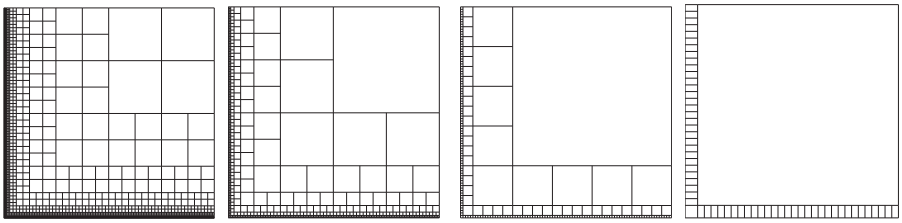
Table 6.5: Multigrid convergence factor for 64^2 and 128^2 problem with jump discontinuity across sub-domain interface

$d_1/d_2 \rightarrow$ Grid / α $\downarrow$	2	10^1	10^2	10^3	10^4	10^5
64^2 / 1.03	0.16 / 7	0.33 / 13	0.40 / 16	0.40 / 16	0.40 / 16	0.40 / 16
256^2 / 1.009	0.15 / 7	0.30 / 12	0.35 / 14	0.29 / 12	0.29 / 12	0.29 / 12

The Model Problem on *Amr-type* Grids

Here, we experiment with the model problem on some *amr-type* grids. We show that the methods developed in this chapter for *str-type* grids work nicely for certain variants of the *amr-type* grids as well. In these *amr-type* grids, there are no perturbed mesh aspect ratios to improve, and, therefore, the coarsening is always standard, implying a grid reduction factor of at least 4 on all levels. In the present work, we do not perform any implementational adjustments, and simply run the problem on a different grid-sequence. The enumeration is L-shaped and no multigrid components have been altered. The convergence pattern is almost identical to that of *str-type* grids.

The shape of the *amr-type* grids that we use depends on 3 parameters. The number of layers (of different-sized control volumes), the number of control volumes tiled between adjacent layers, and the proportion of cell-volume between cells of adjacent layers; denoted respectively, by n, c, and p. Thus the representation, $(n \times c,\ p)$, describes a particular *amr-type* grid completely. In Figure 6.12, the complete grid sequence is shown for the finest grid represented by $(8 \times 2,\ 2)$. The coarse grids look quite similar to the ones in Figure 6.4, the only difference being that, except on the coarsest grid, all the grids in the *amr-type* sequence have the same cell volume within each L-strip. The results of the experiments on these grids are displayed in Table 6.6, and appear quite satisfactory and stable. Moreover, we use amr-type grid saturation in



(a) Level 1 (b) Level 2 (c) Level 3 (d) Level 4

Figure 6.12: The grid sequence for the *amr-type* grid, $(8 \times 2,\ 2)$. The finest grid has 8 layers, with 2 cells tiled in each. Cells in adjacent layers are at a volume proportion of 4.

Table 6.6: Multigrid convergence factors for different *amr-type* grids.

$(n \times c,\ p) \rightarrow$	$(4 \times 2,\ 2)$	$(8 \times 2,\ 2)$	$(8 \times 4,\ 2)$	$(4 \times 32,\ 2)$	$(2 \times 64,\ 2)$
$\hat{q}^{(m)}$ / $m \rightarrow$	0.16 / 7	0.16 / 7	0.16 / 7	0.16 / 7	0.16 / 7

Chapter 7 to handle strongly varying wavenumbers in 2d Helmholtz equations.

Chapter 7

Multigrid Preconditioning for the Indefinite Helmholtz

In this chapter we develop multigrid preconditioners for the high-frequency indefinite Helmholtz equation. The wavenumbers of these equations have a strongly varying spatial dependence. A peculiar property of the solution is that it has the so-called *evanescent waves* (with extremely steep gradients at the domain edges) which perturb the problems and they get very difficult to solve efficiently. We propose and test multigrid preconditioning schemes for these problems. One of the proposed schemes is based on the multigrid ideas developed in Chapter 6 and customized here for the problem at hand. We analyze the model problems for their requirement of minimal numerical resolution due to accuracy constraints, and discretize them on locally refined grids that are customized according to this requirement. These grids are piecewise equidistant with two different layers of cells, having a specific ratio between their mesh sizes. The multigrid method is then employed to approximately invert the Krylov-preconditioner, which is the complex shifted Helmholtz operator as in [14].

7.1 Application, and the Model Problems

2*d* Helmholtz problems with strongly varying Helmholtz terms arise as simplified models of equations originating from quantum mechanics [12]. The solutions also exhibit evanescent waves at the south and the east edges of the domain. One of the ways to tackle these problems efficiently is to saturate grid cells (up to a margin) near these boundaries, and to use a larger mesh size (at least double) elsewhere in the domain. Another method is to use the same mesh size throughout and to employ a sophisticated multigrid method (for preconditioner inversion). In this chapter, we explore the merits of both these approaches. Cell-centered Finite Volume Method is used for discretizing the continuous problem. This scheme has an anticipated second order accuracy over both grid topologies that we use, i.e., the full-grid as well as customized grids with local refinement.

7.1.1 Application Problem

We now give a short introduction to the application which leads to these model problems. The scattering of the elementary particles is governed by the fundamental equation of quantum mechanics, i.e., the Schrödinger equation:

$$\iota\hbar\frac{\partial}{\partial t}\psi(r,t) = -\frac{\hbar^2}{2m}\nabla^2\psi(r,t) + V(r,t)\psi(r,t) \tag{7.1}$$

where $\iota = \sqrt{-1}$, $\hbar$ is the Planck's constant, m is the mass of the particle, and r represents the radial displacement. The wave function ψ and the potential energy V, have both spatial as well as temporal dependence. In general, ψ does not have a separable form, i.e., the spatial and temporal parts cannot be isolated. However, in the case of certain potentials this is possible and results in the so-called time-independent Schrödinger equation, commonly abbreviated as TISE [64, 65]:

$$-\frac{\hbar^2}{2m}\nabla^2\psi(r) + V(r)\psi(r) = E\psi(r) \tag{7.2}$$

where E is constant. The TISE is obviously a Helmholtz-type equation.

McCurdy et al, in [12], describe a scattering problem, where they study the collisional breakup of a system of charged particles. The typical scenario is that an electron hits a Hydrogen molecule (H_2) and ejects out from it, two electrons leaving a positive charged Hydrogen ion. The mechanics of this collision involves the Schrödinger equation for this problem in six variables. They decompose this problem in sets of coupled $2d$ second order differential equations, and solve them numerically through finite differences. The real part of the radial functions in $2d$ displays evanescent waves that decay exponentially and are eventually absorbed.

In a similar research scenario, Vanroose[1] et al, in [66], report on the photo-induced breakup of the H_2 molecule while studying molecular electron correlation. They study similar phenomena as in [12], where both electrons are ejected out of the Hydrogen molecule by the absorption of a single photon. The mechanics of the ejected electrons is described by the TISE with spatially dependent wavenumber, and leads to the model problems.

7.1.2 The Model Problems

The general form of the Helmholtz boundary value problem representing the two model problems is:

$$\begin{aligned}
\left[-\nabla^2 - \phi(x,y) \right] u(x,y) &= f(x,y) && (x,y) \in \Omega = (0,L)^2 \\
u(0,y) = u(x,0) &= 0 && \text{Dirichlet at South/West edges} \\
\frac{\partial u}{\partial x} &= -\iota K u(L,y) && 1^{st} \text{ order Sommerfeld at East edge} \\
\frac{\partial u}{\partial y} &= -\iota K u(x,L) && 1^{st} \text{ order Sommerfeld at North edge}
\end{aligned} \tag{7.3}$$

[1] The Helmholtz model problems of this chapter originate from Wim Vanroose, University of Antwerpen; and stem from the research conducted in [66]

Model problems 1 and 2 are henceforth abbreviated as MP1 and MP2. For both model problems, K ranges between 0 and 5, and $f(x,y) = \frac{1}{e^{x^2+y^2}}$. The model problems are further characterized by:

$$\text{MP1:} \quad \phi(x,y) = \lambda\left(\frac{1}{e^{x^2}} + \frac{1}{e^{y^2}}\right) + K^2; \qquad 0 < \lambda < 10 \ \ \& \ \ L = 50 \tag{7.4}$$

$$\text{MP2:} \quad \phi(x,y) = \frac{1}{x} + \frac{1}{y} + K^2; \qquad 50 < L < 200 \tag{7.5}$$

For some specific λ, K, and L, the solution appears in Figure 7.1.

7.2 Mesh Size Analysis of the Model Problems

To ensure acceptable numerical solution, the discretization of the indefinite Helmholtz equation has to satisfy certain mesh size constraints. These constraints appear in [13, 67] and are based on finite element formulations. One such constraint is that k^2h^3 should stay constant (where k is the constant wavenumber in $(0,1)^2$ domain). We consider instead the accuracy criterion used in [14], i.e., the mesh size constraint $kh < 0.625(= 5/8)$ on the finest grid, which ensures acceptable accuracy (10 discrete points per wave) of the computed solution. This constraint is a worst-case limit of the one in [13], and is therefore sufficient. To speculate about the mesh sizes required for MP1 and MP2, we first transform these equations to the dimensionless form, i.e., we scale the entire system to the unit square. For this, let $\widetilde{x} = x/L$, and $\widetilde{y} = y/L$. Evaluating the second derivatives, we are led to the following transformation to $(0,1)^2$.

$$\text{MP1} \Rightarrow -\frac{\partial^2 u}{\partial \widetilde{x}^2} - \frac{\partial^2 u}{\partial \widetilde{y}^2} - L^2\left(\lambda\left[\frac{1}{e^{L^2\widetilde{x}^2}} + \frac{1}{e^{L^2\widetilde{y}^2}}\right] + K^2\right)u(\widetilde{x},\widetilde{y}) = \frac{L^2}{e^{L^2(\widetilde{x}^2+\widetilde{y}^2)}}$$

$$\text{MP2} \Rightarrow -\frac{\partial^2 u}{\partial \widetilde{x}^2} - \frac{\partial^2 u}{\partial \widetilde{y}^2} - L^2\left(\frac{1}{\widetilde{x}L} + \frac{1}{\widetilde{y}L} + K^2\right)u(\widetilde{x},\widetilde{y}) = \frac{L^2}{e^{L^2(\widetilde{x}^2+\widetilde{y}^2)}}$$

Comparing these equations with the standard indefinite Helmholtz equation (with constant wavenumber k), we have:

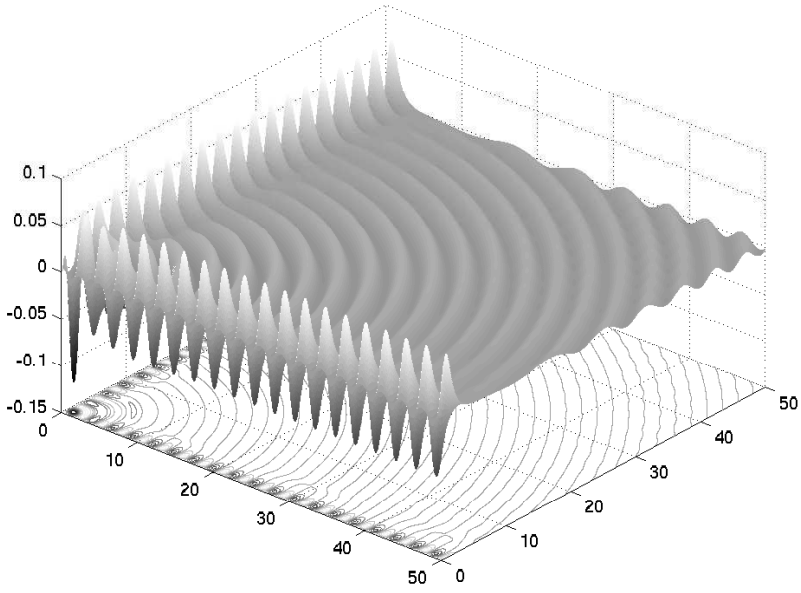
$$\text{MP1} \Rightarrow \quad k = L\sqrt{\left(\lambda\left[\frac{1}{e^{L^2\widetilde{x}^2}} + \frac{1}{e^{L^2\widetilde{y}^2}}\right] + K^2\right)}; \ \ L = 50 \tag{7.6}$$

$$\text{MP2} \Rightarrow \quad k = L\sqrt{\left(\frac{1}{\widetilde{x}L} + \frac{1}{\widetilde{y}L} + K^2\right)} \tag{7.7}$$

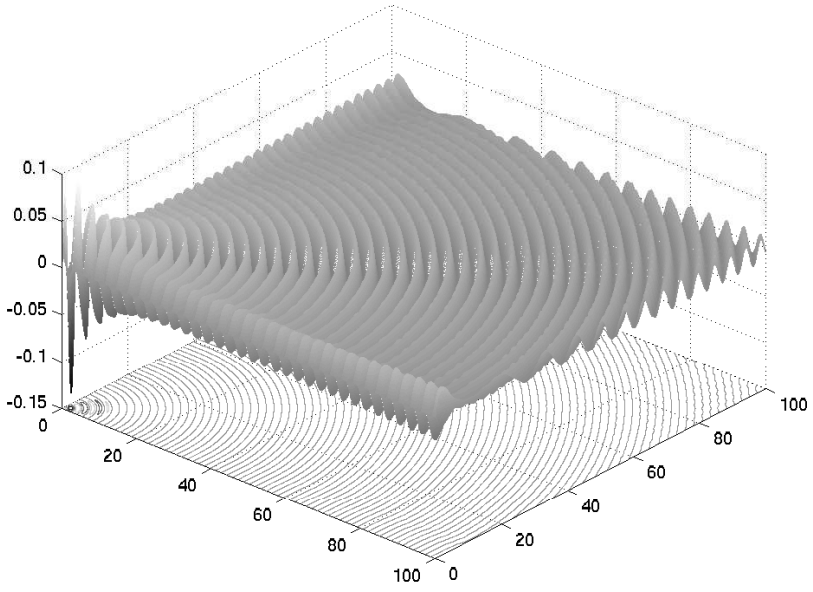
7.2.1 The Mesh Size Analysis for MP1

The mesh size analysis of MP1 is particularly straightforward. We can obtain an upper limit on the wavenumber in $(0,1)^2$ for MP1 (for given values of λ and K), by evaluating the supremum of k from Equation 7.6. It is straight forward to see that this supremum occurs at the origin, i.e., at $(\widetilde{x},\widetilde{y}) = (0,0)$. Therefore, for MP1 the highest possible wavenumber in the unit square domain is given by:

$$k = 50\sqrt{2\lambda + K^2}$$



(a) MP1 with $\lambda = 7$, $K = 2$



(b) MP2 with $L = 100$, $K = 5$

Figure 7.1: The surface generated by the real part of the model problem solution for specified values of the parameters.

Table 7.1: Maximum mesh size limits for MP1

λ \ K	1	2	3	4	5
0	0.625	0.312	0.208	0.156	0.125
1	0.360	0.255	0.188	0.147	0.120
2	0.279	0.221	0.173	0.140	0.116
3	0.236	0.197	0.161	0.133	0.112
4	0.208	0.180	0.151	0.127	0.109
5	0.188	0.167	0.143	0.122	0.105
6	0.173	0.156	0.136	0.118	0.102
7	0.161	0.147	0.130	0.114	0.100
8	0.151	0.139	0.125	0.110	0.097
9	0.143	0.133	0.120	0.107	0.095
10	0.136	0.127	0.116	0.104	0.093

Substituting this expression in $kh = 5/8$, we can calculate the maximum mesh size limit to be observed for MP1 in the unit square. As the scaling to the unit square was only for the purpose of analysis, we multiply the obtained mesh sizes by $L = 50$, which can then be directly used in the experiments. After multiplication by L, this gives:

$$h = \frac{5}{8\sqrt{2\lambda + K^2}}$$

For varying values of λ and K, these mesh sizes limits are given in Table 7.1

As an example, consider the task of solving MP1 for $(\lambda, K) = (9, 2)$. Reading from Table 7.1, we must have a mesh size of 0.133 which amounts to 376 cells along each dimension. To have multilevel compatibility, we realize this limit as 384^2, which after 8 times of successive halving, can be solved exactly at the size of 3^2. We immediately see that if we use a uniform equidistant grid, this results in 147456 unknowns, in order to ensure sufficient accuracy of the computed solution. An experiment based on this speculation is carried out in Section 7.5.

It is immediately apparent that this mesh size constraint is due to the strong variation of the wavenumber at the origin. This gives rise to the idea of grid saturation near the origin and along the edges. Care must be exercised however, in marking the dense and the sparse regions of the grid. To ensure that the dense area is wide enough to engulf the entire subdomain where evanescent waves are expected, we mark off the domain by the straight lines $y = 1/3$, and $x = 1/3$. See Figure 7.2. We find that the mesh size requirements in the sparse region of the grid (depicted as the white region in Figure 7.2), is practically dependent only on K, and the small contribution from the λ term in Equation 7.6 can be neglected. This gives the mesh sizes in the first row of Table 7.1. For MP1 with $(\lambda, K) = (9, 2)$, the mesh size required (in the remaining white region) is conveniently more than two times the fine mesh size required near the origin. This local mesh refinement topology agrees well with the constraint $kh < 5/8$, and therefore provides sufficient confidence in the numerical solution.

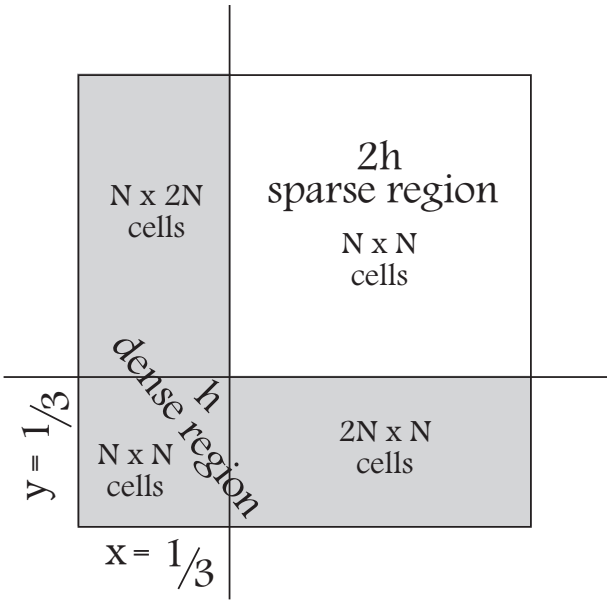


Figure 7.2: The figure depicts the isolation of the dense (gray) and the sparse (white) regions of the domain

7.2.2 The Mesh Size Analysis of MP2

Compared to MP1, the mesh size analysis proceeds somewhat differently for MP2. We realize that for MP2, the wavenumber supremum cannot directly be obtained in the unit square, as the function ϕ has a singularity at the origin. We therefore choose to stay with the maximum discrete wavenumber that our cell-centered discretization scheme allows. In the unit square, the maximum discrete wavenumber k is obtained by substituting $(\widetilde{x}, \widetilde{y}) = (h/2, h/2)$ in Equation 7.7. Subsequently, substitution of this k in $kh = 5/8$, gives a quadratic expression, from which we pick the positive mesh size, and discard the negative. We take care to scale back the mesh sizes to $(0, L)^2$.

$$
\begin{aligned}
k &= L\sqrt{\frac{4}{hL} + K^2} \\
\Rightarrow h &= \frac{1}{2}\left\{-\frac{4}{K^2} + \sqrt{\left(\frac{4}{K^2}\right)^2 + \left(\frac{5}{4K}\right)^2}\right\}
\end{aligned}
\tag{7.8}
$$

Equation 7.8, gives the maximum mesh size limit required in the proximity of the origin. We chop off the domain at the lines $y = 1/3$ and $x = 1/3$. The mesh size requirement in the truncated domain (white area in Figure 7.2) is much less for moderate values of K, as expected. h for the truncated domain is given by Equation 7.9, and is scaled to

Table 7.2: Upper limit on the mesh size for MP2 with respect to $(0, L)^2$

K / L	1	2	3	4	5
$\forall\, L$	0.095	0.089	0.082	0.075	0.068

Table 7.3: Upper limit on the mesh size for MP2 with respect to $(L/3, L)^2$

K / L	1	2	3	4	5
50	0.591	0.308	0.207	0.156	0.125
100	0.607	0.310	0.208	0.156	0.125
150	0.613	0.311	0.208	0.156	0.125
200	0.616	0.311	0.208	0.156	0.125

the $(0, L)^2$ domain as before.

$$\begin{aligned} k &= \sqrt{L(6 + LK^2)} \\ \Rightarrow h &= \frac{5}{8}\sqrt{\frac{L}{6 + LK^2}} \end{aligned} \tag{7.9}$$

Tables 7.2 and 7.3 gives the mesh size limits that must be obeyed in the proximity of the origin, and in the truncated domain, respectively.

This completes the discussion of the essential discrete resolution.

7.3 The Discretization

In this section, we consider the discretization of the model problems. The discussion involves the evaluation of numerical fluxes for a grid cell that is surrounded by cells equal to itself (in area) on all four sides. In this section, we avoid the discussion of flux mismatch at the interface of refinement layers, as that is dealt exactly as described in Chapter 6 and demonstrated in Appendix D. The discretization scheme is the cell-centered FVM as mentioned earlier. We also demonstrate at least one method of discretizing the first order derivative boundary conditions and eliminating them from the operator.

Integrating Equation 7.3 over the domain Ω, gives:

$$\int_\Omega [\, -\nabla \cdot \nabla - \phi(x, y)\,]\, u(x, y) = \int_\Omega f(x, y) \tag{7.10}$$

We cover the domain with a cell centered grid having M square control volumes in all, and apply the Gauss Divergence theorem. For the purpose of this section, let m index an anonymous (interior) control volume. This implies that the nodal location (x_m, y_m) is neighboured by (x_m, y_{m-1}), (x_{m+1}, y_m), (x_m, y_{m+1}), and (x_{m-1}, y_m), to its south, east,

north and west respectively. Then, for this m^{th} cell $\square\overline{ABCD}$:

$$\int_A^B \frac{\partial u_m}{\partial y} dx_m - \int_B^C \frac{\partial u_m}{\partial x} dy_m - \int_C^D \frac{\partial u_m}{\partial y} dx_m + \int_D^A \frac{\partial u_m}{\partial x} dy_m - \int_{\Omega_m} \phi_m u_m d\Omega_m = \int_{\Omega_m} f_m d\Omega_m \tag{7.11}$$

The numerical flux (i.e., the 1^{st} order derivative in each of the first four terms) is approximated by central finite differences, at the center of each edge ($\overline{AB}$, $\overline{BC}$, $\overline{CD}$, and $\overline{DA}$) of the m^{th} control volume. $\phi_m = \phi(x_m, y_m)$, which is the value of the spatial wavenumber ϕ computed at the center of the cell; and likewise f_m. The principle of prime importance here, is the conservation of flux, which is ensured by imposing the condition that the sum of the *net fluxes* (see Remark 6.3.4) to-and-from a cell edge is zero.

Let h represent the mesh size, then for an interior cell we approximate each of the fluxes in Equation 7.11 by $O(h^2)$ central finite differences. E.g. the net flux through the south face of the m^{th} interior cell F_s is:

$$F_s = \int_A^B \frac{\partial u_m}{\partial y} dx_m \approx \frac{u(x_m, y_m) - u(x_m, y_{m-1})}{h} h \tag{7.12}$$

The other terms in Equation 7.11 are approximated w.r.t. their point values, such as:

$$\int_{\Omega_m} \phi_m u_m d\Omega_m \approx h^2 \phi(x_m, y_m) u(x_m, y_m)$$

Now, we describe the flux discretization for control volumes adjacent to a domain boundary. For generality, we implement Robin boundary conditions on all edges of the domain, and realize Dirichlet and 1^{st} order Sommerfeld as its special cases. As an example, consider the south (Dirichlet) edge of the domain, and let the prescribed Robin boundary condition on this edge be given by:

$$c_s^{(1)} \frac{\partial u(x_m, 0)}{\partial y} + c_s^{(0)} u(x_m, 0) = g_s(x_m, 0) \tag{7.13}$$

$c_s^{(1)}, c_s^{(0)} \in \mathbb{C}$. E.g. $c_s^{(1)} = g_s = 0$, and $c_s^{(0)} = 1$, gives us the required homogeneous Dirichlet boundary condition on the south edge. We discretize the derivative in the Robin boundary, Equation 7.13, by the $O(h)$ one sided finite difference, at the midpoint of the appropriate cell interface. We show the evolution for the south and the north edges, which carries over similarly to the west and the east edges, respectively.

$$\begin{aligned} c_s^{(1)} \frac{u(x_m, y_m) - u(x_m, 0)}{h/2} + c_s^{(0)} u(x_m, 0) + O(h) &\approx g_s(x_m, 0) \\ c_n^{(1)} \frac{u(x_m, L) - u(x_m, y_m)}{h/2} + c_n^{(0)} u(x_m, L) + O(h) &\approx g_n(x_m, L) \end{aligned} \tag{7.14}$$

and leads to:

$$\begin{aligned} u(x_m, 0) &= \frac{-2c_s^{(1)}}{-2c_s^{(1)} + hc_s^{(0)}} u(x_m, y_m) + \frac{hg_s(x_m, 0)}{-2c_s^{(1)} + hc_s^{(0)}} \\ u(x_m, L) &= \frac{2c_n^{(1)}}{2c_n^{(1)} + hc_n^{(0)}} u(x_m, y_m) + \frac{hg_n(x_m, L)}{2c_n^{(1)} + hc_n^{(0)}} \end{aligned} \tag{7.15}$$

Next, we discretize the appropriate boundary fluxes of the boundary neighbouring cells with 1-sided $O(h)$ finite differences, and then we substitute in them, the boundary approximations acquired from the discrete boundary conditions, and given by equations, such as Equation 7.15. This gives the following $O(h)$ boundary approximations, (F_{edge}) now represents the flux through a boundary edge:

$$F_s \approx \frac{u(x_m, y_m) - u(x_m, 0)}{h/2} h$$

$$\Rightarrow F_s \approx 2\left[1 + \frac{2c_s^{(1)}}{-2c_s^{(1)} + hc_s^{(0)}}\right] u(x_m, y_m) - \frac{2hg_s(x_m, 0)}{-2c_s^{(1)} + hc_s^{(0)}} \tag{7.16}$$

Similarly, the other boundary fluxes are approximated as:

$$F_e \approx 2\left[1 - \frac{2c_e^{(1)}}{2c_e^{(1)} + hc_e^{(0)}}\right] u(x_m, y_m) - \frac{2hg_e(L, y_m)}{2c_e^{(1)} + hc_e^{(0)}} \tag{7.17}$$

$$F_n \approx 2\left[1 - \frac{2c_n^{(1)}}{2c_n^{(1)} + hc_n^{(0)}}\right] u(x_m, y_m) - \frac{2hg_n(x_m, L)}{2c_n^{(1)} + hc_n^{(0)}} \tag{7.18}$$

$$F_w \approx 2\left[1 + \frac{2c_w^{(1)}}{-2c_w^{(1)} + hc_w^{(0)}}\right] u(x_m, y_m) - \frac{2hg_w(0, y_m)}{-2c_w^{(1)} + hc_w^{(0)}} \tag{7.19}$$

Depending on the location of the m^{th} control volume within the grid, appropriate discrete approximations of the continuous flux are substituted in Equation 7.11. The last terms of Equations 7.16 - 7.19 are substituted into the right hand side of the system. We immediately recognize this as the eliminated boundary scheme for our model problems, and this completes the cell-centered FVM discretization.

Remark 7.3.1. (Alternative $O(h^2)$ discretization of the boundary fluxes) *The discretization of the boundary fluxes that we employ in our work is only first order accurate. A proof is present in [63], which ensures that this scheme is second order accurate and that the accuracy pollution near the domain boundary is strictly local. An alternative $O(h^2)$ discretization is also available [68]. It consists of using the following 1-sided $O(h^2)$ stencils for discretizing the Robin boundary conditions as well as the boundary fluxes.*

$$\text{Forward Difference:} \quad \frac{\partial u(0, y)}{\partial x} = \left(\frac{1}{3h}\right)\left[-8u(0, y) + 9u(h/2, y) - u(3h/2, y)\right]$$

$$\text{Backward Difference:} \quad \frac{\partial u(L, y)}{\partial x} = \left(\frac{1}{3h}\right)\left[8u(L, y) - 9u(L - h/2, y) + u(L - 3h/2, y)\right]$$

From the vantage point of approximating the solution, this $O(h^2)$ discretization near the boundaries is a better approximation when at least two (out of the four) domain edges have derivative boundary conditions prescribed on them. There is no known enhancement in the numerical approximation if all the boundaries are Dirichlet. It is important to point out that this stencil is asymmetric and therefore negative effects of asymmetry degrade iterative solver performance to some extent. Therefore, we stay with the $O(h)$ boundary discretization.

7.4 The Multigrid Preconditioned Krylov Solver

In this section we describe the actual numerical solver that we employ in our experiments. The iterative Krylov method that we use is Bi-CGSTAB [44]. It is also worthwhile to note that and a recent competitor called IDR(s) [69] provides a good alternative to Bi-CGSTAB. We will not go into any length for describing the Krylov component of the solver; rather, our aim is to describe here the preconditioning operator as well as the components of our multigrid method with which the preconditioning operator is approximately inverted.

7.4.1 The Preconditioning Operator

The discrete indefinite Helmholtz operator cannot be handled directly by standard multigrid. A sophisticated multigrid method, [70] by A. Brandt and I. Livshits, exists for the indefinite constant coefficient Helmholtz equation. Efficient implementation of this method makes use of rotated Cartesian grids and resolves the wave and the ray components of the error separately. It is not readily obvious how this method may be extended to incorporate strongly varying coefficients that are possessed (say) by MP1 and MP2. Another approach to solving the indefinite Helmholtz equation, is to use preconditioned Krylov methods, and employ multigrid for approximate preconditioner inversion. This approach can be traced back to the 80s [13, 71]. A good method belonging to this approach was developed at Delft in 2006 by Y. Erlangga et al [14]. This approach employs the following complex shifted Helmholtz operator as a Krylov preconditioner for the indefinite Helmholtz operator.

$$\mathcal{M} = -\Delta - (\beta_1 - \imath\beta_2)k^2(x, y) \tag{7.20}$$

where k is the wavenumber as in Section 7.2. This operator can be inverted approximately by multigrid, and has stood out to be more robust relative to the other preconditioners such as proposed in [13, 71]. The best combination of the real parameters β_1, β_2, reported in [14] is $(1, 0.5)$. Reducing β_2 brings the spectrum of the preconditioner closer to that of the original problem (and makes it a better preconditioning operator), but hampers its approximate multigrid inversion. As a general rule the least value of β_2, for which a good multigrid method might possibly be tuned in, turns out to be the best preconditioner for the model problems considered here.

7.4.2 Smoother

In the context of the indefinite Helmholtz equation, it is known that *fast local relaxation* techniques such as Gauss-Seidel often perform worse than slower counterparts such as ω-Jacobi. The reason [70] is that the frequencies corresponding to the indefinite part of the spectrum are not treated adequately with local relaxation method. The relatively faster methods such as Gauss Seidel thus excite these frequencies more, compared to slower local methods such as ω-Jacobi or the Kacmarcz method [70]. An alternative is to use *global* smoothing methods such as *ILU(0) smoothing* [8]. For the indefinite Helmholtz, this was exploited in [72]. For solution techniques that store the coefficient matrix, this is the smoother of choice for the complex shifted Helmholtz

preconditioner. The following algorithm describes ILU(0) smoothing for n smoothing steps. We'd like to point out that it is possible to implement some ILU(0) versions on the stencil basis.

Let $\mathcal{L}$ and $\mathcal{U}$ represent the lower and the upper matrices (respectively) resulting from the ILU decomposition (with zero fill-in) of the discrete operator (on level l, denoted A_l), so that:

$$A_l = \mathcal{L}\mathcal{U} - \mathcal{R} \tag{7.21}$$

then n smoothing steps are executed as illustrated by Algorithm 3.

Algorithm 3 n smoothing steps based on ILU(0)

INPUTS: A_l, n, v_l (error), rhs (the right hand side)
OUTPUT: Smoothed v_l

1 Evaluate $\mathcal{L}$, $\mathcal{U}$, and $\mathcal{R}$
2 do $i = 1$ to n
  d_l = rhs $+ \mathcal{R}v_l$
  $e_l = \mathcal{L}^{-1}d_l$
  $v_l = \mathcal{U}^{-1}e_l$
  continue

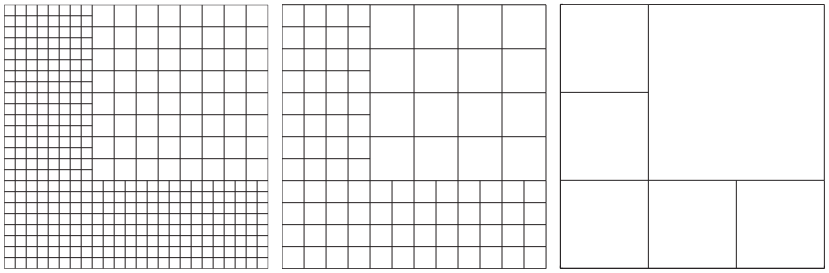
For experiments on locally refined grids we will use the ω-Jacobi point based smoother with different values of the relaxation parameter ω. The parameter values are mentioned with experimental results. In experiments on regular equidistant grids we also use the ILU smoother, which seems to work best in the present context when used in conjunction with an under-relaxation parameter. We mention the value of this parameter with the experiments in Section 7.5.

7.4.3 Transfer Operators

We use the four-point restriction and bi-linear interpolation operators for grid transfer as described in Chapter 2, Section 2.3. Implementationally, we use lexicographic ordering in the regular grids, and L-shaped ordering of the grid unknowns in locally refined grids. If transfer across the layer interface is required (in the case of locally refined grids), then the transfer operators as described in Chapter 6, Section 6.3.3 are employed. It suffices to mention that grid-math across layer interface involves extra work, and can be circumvented by a careful choice of the finest grid.

7.4.4 The Coarse Grid Operators

Both, discretization coarse grid operators as well as the Galerkin coarse grid operators are employed in the numerical experiments. The Galerkin operator is used in regular grids, and the simple DCG operator on locally refined grids. Both work well in their capacities, and have their own advantages and downsides.



(a) Grid 5 (b) Grid 6 (c) Coarsest

Figure 7.3: An assortment of 3 grids, from a sequence of locally refined grids, chosen from one of the experiments, the finest cannot be shown at this scale.

7.4.5 Cycle-type and Complexity

We can choose the finest grid in such a fashion so that standard coarsening is always possible for both grid topologies. The grid size is halved at each coarse level, and therefore, the usual complexity estimates for $2d$ grids (as presented in Chapter 4, Table 4.1) hold true. We experiment both with V as well as the F cycles. It turns out that with ω-Jacobi smoothing, we essentially require F-cycles, whereas, with ILU(0) smoothing V cycles can also sometimes perform the job. The cycle-types are mentioned with the experiments in Section 7.5.

7.5 Numerical Experiments

7.5.1 Visuals of the Locally Refined Grid

Here we present visual examples of a locally refined grid, and some of the subsequent coarse grids, that we use in the experiments for solving MP1 and MP2. Figure 7.3 displays the fifth, the sixth and the eighth (last) grid of a grid sequence which exactly follows the refinement scheme described in Section 7.2.1. These figures result from the actual implementation of the general idea represented by Figure 7.2, and Chapter 6.

7.5.2 Accuracy Vis-a-vis Grid Topology

The mesh size considerations of Section 7.2 ensure ample numerical resolution for the computed solution. Our numerical scheme for the locally refined grids, use simple bilinear interpolation at the layer interface. This prompts us to test and ensure that we have second order accuracy with this scheme over the devised grid topology. For this purpose, we substitute a substantially varying test function, given by:

$$u(x,y) = \left(\sin\frac{20\,\pi\,x}{L} + \sin\frac{20\,\pi\,y}{L}\right) / \left(\frac{x+y}{L} + \frac{1}{5}\right)$$

Table 7.4: MP1 results on equidistant grids with combinations of multigrid components. Precon. shows the performance of the multigrid method on the preconditioner as a stand-alone problem. This value is read as the observed multigrid contraction number against the number of cycles used.

Problem	(β_1, β_2)	Grid	MG-parameters Rel / ω / cycle / Restrict / CG	Precon.	Solver
MP1-pb1	$(1, 0.5)$	384^2	ω-JAC / 0.8 / $F(1,1)$ / FP / DCG	$^{0.38}/_{15}$	100
	$(1, 0.3)$	384^2	ILU(0) / 0.5 / $F(1,1)$ / FP / GCG	$^{0.42}/_{16}$	160
	$(1, 0.2)$	384^2	ILU(0) / 0.2 / $F(1,1)$ / FP / GCG	$^{0.64}/_{31}$	97
MP1-pb2	$(1, 0.5)$	384^2	ω-JAC / 0.8 / $F(1,1)$ / FP / DCG	$^{0.44}/_{17}$	241
	$(1, 0.3)$	384^2	ILU(0) / 0.5 / $F(1,1)$ / FP / GCG	$^{0.48}/_{19}$	117
	$(1, 0.2)$	384^2	ILU(0) / 0.2 / $F(1,1)$ / FP / GCG	$^{0.72}/_{43}$	160

in Equation 7.3 and solve the resulting Helmholtz equation by our solver. The coefficients of the boundary conditions are the same as for Equation 7.3, but their right-hand-sides have been adapted so as to comply with this test function at the boundaries. The numerical solution reproduces the test function accurate to the second order over the grid refinement rendered by dividing every control volume into 4 square control volumes. With this confidence of second order accuracy over the presented local refinement grid topology, we now test the solver through numerical experimentation over regular and refined grids.

7.5.3 Experiments on Regular Equidistant Grids

In this batch of experiments we use regular equidistant grid and check solver performance for MP1 and MP2. We select two problems from each category, i.e., MP1 and MP2, by choosing combinations of particular parameters that define these problems, as well as with combinations of components that define our multigrid method. This forms four (4) problems in all that we test. These are enumerated as follows:

- MP1-pb1: $(\lambda, K) = (7, 2)$
- MP1-pb2: $(\lambda, K) = (1, 4)$
- MP2-pb1: $(L, K) = (50, 1)$
- MP2-pb2: $(L, K) = (125, 2)$

Experimental results are displayed in Tables 7.4 and 7.5.

The results in Table 7.4 present some very interesting features of this solver. A prominent observation is the influence of using the global relaxation method ILU(0). This conforms to the well-known phenomena [70, 73] that local GS-type smoothers do a comparatively poor job for such indefinite problems. They excite the error components that are invisible on the fine grid, thus jeopardizing the smoothing process and making it ineffective. ILU(0) is somewhat more expensive than local relaxation schemes. Nevertheless, it usually allows faster convergence compared to ω-Jacobi for these model problems. For MP1 and MP2, the experiments suggest that the solver

Table 7.5: MP2 results on equidistant grids with combinations of multigrid components

Problem	(β_1, β_2)	Grid	MG-parameters Rel / ω / cycle / Restrict / CG	Precon.	Solver
MP2-pb1	$(1, 0.5)$	768^2	ω-JAC / 0.8 / $F(1,1)$ / FP / DCG	$^{0.33}/_{13}$	118
MP2-pb2	$(1, 0.5)$	1536^2	ω-JAC / 0.8 / $F(1,1)$ / FP / DCG	$^{0.36}/_{13}$	225

is more reliable and stable with simple components, such as the ω-Jacobi smoother (with different relaxation parameters ω), piecewise constant restriction, and the discretization coarse grid operator.

Another observation from the experiments in Table 7.4, is that for MP1, the highest magnitude of the spatial wavenumber is not the only convergence hampering factor. Geometric trait of the solution, also play a role in it, but it is somewhat non-trivial to quantify exactly. It is clear that with $(\lambda, K) = (7, 2)$ (solution with evanescent waves), and with $(\lambda, K) = (1, 4)$ (solution without evanescent waves), the highest wavenumber is the same, and requires the same mesh resolution, however the convergence behaviour is quite different in the two situations.

The ILU(0) smoother usually seems to work in a stable manner with the Galerkin operator on the coarse grids. MP2 is more sensitive with respect to the smoother, and ω-Jacobi seems to be a safe bet. The mesh size requirement for MP2 on regular equidistant grids is given in Table 7.2. For MP2-pb1, this requirement translated into grid size reads, 768^2 (exact solve on grid level 9, i.e. 3^2); while for MP2-pb2, it is 1536^2 (exact solve on level 10 having a 3^2 grid size). The grid size is large owing in turn to the large wavenumbers for MP2. Naturally, the number of iterations with Bi-CGSTAB also grows linearly with the wavenumbers, as experienced by Erlangga et al in [14]. The results are presented in Table 7.5.

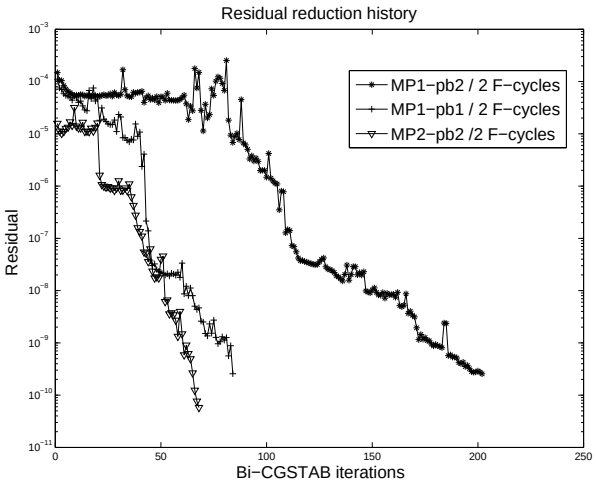


Figure 7.4: Residual reduction histories for some experiments.

Table 7.6: Experiments on grids refined at the south and the west edges

Problem	(β_1, β_2)	Grid	Precond.	Solver (1 precond.)	Solver (2 precond.)
MP1-pb1	(1,0.5)	128^2	$^{0.34}/_{10}$	111	83
MP1-pb2	(1,0.5)	128^2	$^{0.36}/_{12}$	278	201
MP2-pb1	(1,0.5)	256^2	$^{0.33}/_{9}$	78	67
MP2-pb2	(1,0.5)	512^2	$^{0.38}/_{11}$	382	185

7.5.4 Experiments on Locally Refined Grids

In this section we present a few results on the two model problems discretized on locally refined grids, such that the mesh size requirements are fulfilled. In Section 7.5.3, we used the grid size 384^2 for MP1, here in Table 7.6, we quote the locally refined grid size as 128^2. This simply implies that the white part (depicted in Figure 7.2) has the size 128^2, while the mesh size in the adjacent fine layer is just half as large. This scheme gives the required mesh density near steep gradients (and near the singularity) of the solution.

We have not yet implemented an ILU(0) scheme for locally refined grids as currently the project is matrix-free. Although this relieves us of storage considerations, this also implies the unavailability of more involved matrix-dependent smoothing methods, such as ILU(0). We thus test our method only with ω-Jacobi, with varying values of the complex shift parameter β_2, and with FP restriction, bilinear prolongation and the DCG operator. The results of experiments on MP1 and MP2 on grids amply refined near the perturbed regions, are presented in Table 7.6.

We see that MP2-pb1 is converged easier than MP1-pb1, however, this does not scale well with changes in parameters on which the wavenumbers depends, and therefore, solving either of them is very challenging. We tested both problems with many different complex shift values. For example, MP2-pb2 with $\beta_2 = 0.4$ causes the solver to converge in 485 iterations with 1 multigrid preconditioning step and in 286 iterations with 2. Figure 7.4 gives a graphical residual reduction history for MP1 and MP2. The reduction is around 7 orders of magnitude, and experiments with 2 multigrid preconditioning cycles, show a stable reduction history. In cases where (say) 4 orders of magnitude reduction suffices, the algorithm (for MP1-pb2) with 1 mg cycle / iteration, would converge in around 125 iterations.

We see that although the proposed multigrid preconditioners are good, there is still a lot of need for improvement. Moreover, both model problems, form a kind of benchmark for multigrid preconditioned iterative solution techniques. They are essentially non-trivial to solve, and require more research and perseverance.

Chapter 8

Conclusions and Outlook

8.1 Conclusions

This thesis dealt with different problems containing the common issue of efficient coarse grid correction. The two main themes treated here were grid-aligned anisotropy for d-dimensional PDEs, and anisotropies resulting from logarithmic grid stretching. Iterative solution of the indefinite Helmholtz equation (for applications from quantum mechanics) formed the last part. There are therefore a number of conclusions from this work which are summarized below.

The result from Chapter 4 is that efficient multigrid methods for high-dimensional elliptic PDEs, on non-equidistant grids can be built through partial coarsening strategies. Another important message in that chapter, is that point relaxation methods show significantly enhanced smoothing properties in higher dimensions if used with optimal relaxation parameters. To alleviate the implementation issues we showed how difference operator matrices can be put together through tensor methods. This renders testing in abstract higher d dimensions easy. Through numerical results supported by the Local Fourier Smoothing analysis we also showed that partial quadrupling is a strategy of choice in higher dimensions and ensures a computational complexity of $O(M_0)$ (where M_0 are total points on the finest level) even in the worst case, i.e. coarsening in one dimension only. We have also confirmed this through a complexity analysis. Results of the numerical experiments display the excellent multigrid convergence that can be brought about with coarsening strategies proposed, analyzed and experimented in Chapter 4.

A promising technique to handle high-dimensional PDE problems numerically is the sparse grid method; which gives rise to an abundant number of smaller sized problems on equidistant and non-equidistant grids. The non-equidistant grids generate a discrete anisotropy in the system. Although it is well known that multigrid methods are amongst the fastest solvers for elliptic equations, the throughput of a multigrid solver usually depends on how best it could be tuned with optimal attributes which include optimal relaxation and an ideal coarse grid correction. It is often difficult to reach this optimality in a practical situation. Working only with $\omega = 1$, i.e.,

without access to optimal relaxation parameters, this optimality can be mimicked by having multigrid as a preconditioner of a suitable Krylov subspace method, such as Bi-CGSTAB. We showed in Chapter 5, how such a d-multigrid method works when employed as a preconditioner, and supplemented the development with numerical experiments and convergence diagrams. The resulting solver is quite robust and generally applicable to a wide class of discrete parabolic and elliptic problems. We demonstrated the utility by solving the Black-Scholes equation for pricing options dependent on multiple underlying assets. We also indicated that the *grid convergence* of the sparse grid solution is irregular due to a non-differentiable payoff for this application, whereas the convergence of multigrid preconditioned Bi-CGSTAB is excellent.

A geometric multigrid method based on L-shaped coarsening was developed and presented in Chapter 6 for a cell-centered finite volume discretization scheme. This is a hybrid coarsening strategy and results in a combination of full- and semicoarsening in selected regions of the domain, guided by LFA. The method shows near-optimal multigrid convergence with standard components, such as piecewise constant restriction, simple bilinear interpolation and a stationary Gauss-Seidel point-based relaxation scheme. The numerical experiments demonstrated that the method works fine for problems on 2d grids that were logarithmically stretched. Moreover, the method can successfully handle severe stretching, even with V-cycles, without suffering a convergence deterioration, and requires no Krylov acceleration. The complexity of the method is comparable with well known approaches like FAC and $MLAT$; the main difference with these approaches is that the method proposed in Chapter 6, coarsens the mesh in all regions of the computational domain simultaneously, and this leads to a good complexity measure.

The indefinite Helmholtz equation arises in many scientific applications. These include seismics as well as quantum mechanics (QM) amongst others. In Chapter 7, we tackled model problems which arise from simplification of the $6d$ Schrödinger equation in photo-induced Hydrogen ionization experiments. The solutions have evanescent waves with very steep gradients at the south and west edges of the domain, and singularity at the origin. The wavenumber is spatially dependent as well. We solved this problem both on equidistant as well as grids locally refined in the regions of perturbation; by using multigrid for the complex shifted Helmholtz operator as a Krylov preconditioner for the indefinite Helmholtz. The experiments indicated the need of better iterative solvers than the current state-of-the-art. We also observed in that chapter that these spatially dependent Helmholtz equations from this QM application posed a tough benchmark for testing new iterative Helmholtz solvers.

8.2 Outlook

Efficient solution of d-dimensional PDEs still pose a challenge due to the curse of dimensionality, even with the use of sparse grids. The reason is that the sparse grid technique results in sub-problems, and therefore, storage requirements (amongst others) of the largest subproblem is another issue. This can be analyzed in more detail. Further, we also plan to exploit the parallel features of our d-dimensional solver by automating the parallelism, and to work towards realistic higher-dimensional real-life

problems.

Extension of the L-shaped coarsening technique to $3d$ would imply coarsening decisions proceeding along 3d L-shaped geometric structures. Each of these structures would be composed of 3 half (or partial) planes parallel to the coordinate planes. Similar to the 2d case, comparisons for coarsening would be between 2d partial coarsening and 3d full-coarsening for prospective agglomerations. The results are expected to possess the usual scalability for dimensional extensions of the Poisson equation on equidistant grids. However, work still remains to be done in this case.

Helmholtz equations are unique from the perspective that they are just scalar problems, which have continuously defied iterative solution techniques for a very long time now. There are many ideas that float in this context, and new techniques can be researched. Multigrid solution of these equations is one, another is the improvement of the complex-shifted preconditioner. Deflation techniques coupled with this preconditioner have also been developed, and require more outlook especially for 3 and higher-dimensional Helmholtz type problems.

An important area which offers opportunity for development is parallelization of many sequential numerical algorithms. Parallelization of multigrid customized for any application at hand, is both important as well as non-trivial. With the advent of desktop super-computing facilities, such as offered by *Field Programmable Gate Arrays (FPGAs)* and *Graphical Processing Units (GPUs)*, parallel computing is more widely accessible than it was before, and forms a very interesting and potentially rewarding field of research.

Appendix A

The Multigrid Iteration Operator

A.1 Preliminaries

Before introducing the multigrid iteration operator, we would build on a few preliminary concepts. See [8, 2] for more preliminary results and insights.

Any *linear iterative method* for the matrix equation $A_h u_h = b_h$, can be represented in general as:

$$u_h^{i+1} = M u_h^i + s; \qquad i = 1, 2, 3, \cdots \tag{A.1}$$

E.g., the Richardson-type iterative method,

$$\begin{aligned} & u_h^{i+1} = u_h^i + \tau(b_h - A_h u_h^i) \\ \Rightarrow \quad & = (I_h - \tau A_h) u_h^i + \tau b_h \end{aligned} \tag{A.2}$$

and the ω-Jacobi method,

$$u_h^{i+1} = (I_h - \omega D^{-1} A_h) u_h^i + \omega D^{-1} b_h \tag{A.3}$$

can both be realized in the form represented by Equation A.1.

We look into the ω-Jacobi method further. Let $S_h = (I_h - \omega D^{-1} A_h)$, so that with an arbitrary u_h^0, the ${\nu_1}^{th}$ ω-Jacobi iterate is:

$$u_h^{\nu_1} = S_h^{\nu_1} u_h^i + \left(\sum_{k=0}^{\nu_1 - 1} S_h^k \right) \omega D^{-1} b_h$$

which may be written as:

$$u_h^{\nu_1} = S_h^{\nu_1} u_h^i + s_h \tag{A.4}$$

(s_h represents the last term of the equation).

We now show that any error correction iterative scheme (e.g. multigrid) can be written as a particular Richardson iteration. Assume that u_h^i is any approximation of u_h. Then:

$$r_h^i = b_h - A_h u_h^i = A_h e_h^i \tag{A.5}$$

If $\hat{A}_h$ is any easily invertible approximation of A_h, then approximate error $\hat{e}_h^i$ can be written as:

$$\hat{e}_h^i = \hat{A}_h^{-1} r_h^i \tag{A.6}$$

and leads us to the iteration given by Equation A.7:

$$u_h^{i+1} = u_h^i + \hat{e}_h^i \tag{A.7}$$

substituting the expressions for $\hat{e}_h^i$ and subsequently r_h^i in Equation A.7, from Equations A.6 and A.5 respectively; we have:

$$u_h^{i+1} = (I_h - \hat{A}_h^{-1} A_h) u_h^i + \hat{A}^{-1} b_h \tag{A.8}$$

which we recognize as a particular Richardson iteration where the constant τ has been replaced by a more general approximation operator $\hat{A}_h^{-1}$.

We would like to point out that in any iterative method, which can be described in the form of Equation A.8, if the initial guess is an all zero vector, then the γ^{th} iterate evolves as follows:

$$\begin{aligned}
& u_h^0 = \mathbf{0} \\
& u_h^1 = \hat{A}_h^{-1} b_h \\
& u_h^2 = (I_h - \hat{A}_h^{-1} A_h)\hat{A}_h^{-1} b_h + \hat{A}_h^{-1} b_h \\
& \quad = \left(I_h + (I_h - \hat{A}_h^{-1} A_h)\right) \hat{A}_h^{-1} b_h \\
& u_h^3 = \left(I_h + (I_h - \hat{A}_h^{-1} A_h) + (I_h - \hat{A}_h^{-1} A_h)^2\right) \hat{A}_h^{-1} b_h \\
\Rightarrow \quad & u_h^\gamma = \left(I_h + (I_h - \hat{A}_h^{-1} A_h) + (I_h - \hat{A}_h^{-1} A_h)^2 + \cdots + (I_h - \hat{A}_h^{-1} A_h)^{\gamma-1}\right) \hat{A}_h^{-1} b_h \\
& \quad = \left(I_h - (I_h - \hat{A}_h^{-1} A_h)^\gamma\right) \left(I_h - (I_h - \hat{A}_h^{-1} A_h)\right)^{-1} \hat{A}_h^{-1} b_h \\
& \quad = \left(I_h - (I_h - \hat{A}_h^{-1} A_h)^\gamma\right) A_h^{-1} b_h
\end{aligned} \tag{A.9}$$

A.2 The 2-grid Operator

In this section, we follow the same notation (and context) as in Section 2.2.3 (Chapter 2). We now move on to the *two-grid operator* that results from Algorithm 1. For k and $k+1$ representing the fine and the coarse levels respectively, we have:

$$\begin{aligned}
u_k^{i+1} &= S_k^{\nu_2} \left(S_k^{\nu_1} u_k^i + s_k + I_{k+1}^k e_{k+1}\right) + s_k && \text{(A.10)} \\
&= S_k^{\nu_2} \left(S_k^{\nu_1} u_k^i + s_k + I_{k+1}^k A_{k+1}^{-1} I_k^{k+1} \left\{b_k - A_k (S_k^{\nu_1} u_k^i + s_k)\right\}\right) + s_k \\
&= S_k^{\nu_2} \left(I_k - I_{k+1}^k A_{k+1}^{-1} I_k^{k+1} A_k\right) S_k^{\nu_1} u_k^i + \hat{s}_k && \text{(A.11)}
\end{aligned}$$

The last term $\hat{s}_k$ is independent of u_k^i therefore we get the 2-grid error iteration operator as:

$$M_k^{k+1} = S_k^{\nu_2} \left(I_k - I_{k+1}^k A_{k+1}^{-1} I_k^{k+1} A_k\right) S_k^{\nu_1} \tag{A.12}$$

In Equation A.10, the error e_{k+1}, can be replaced by a suitable approximation $\hat{e}_{k+1}$, which can be obtained by recursively using a yet coarser grid in the 2-grid sense.

Starting with an all zero initial guess, if γ cycles are employed for this approximation, then we have:

$$\begin{aligned}\hat{e}_{k+1}^{(\gamma)} &= \left(M_{k+1}^{k+2}\right)^{\gamma} \hat{e}_{k+1}^{(0)} + \tilde{s}_k \\ &= \left(I_{k+1} - \left(M_{k+1}^{k+2}\right)^{\gamma}\right) A_{k+1}^{-1} r_{k+1}\end{aligned} \tag{A.13}$$

according to the result obtained in Equation A.9. Replacing e_{k+1} in Equation A.10 by $\hat{e}_{k+1}^{(\gamma)}$ from Equation A.13, we get:

$$u_k^{i+1} = S_k^{\nu_2}\left[I_k - I_{k+1}^k\left\{I_{k+1} - \left(M_{k+1}^{k+2}\right)^{\gamma}\right\} A_{k+1}^{-1} I_k^{k+1} A_k\right] S_k^{\nu_1} u_k^i + \bar{s}_k$$

where the term $\bar{s}_k$ is independent of u_k^i. Comparing this 3-grid equation with Equation A.11, we immediately recognize the *multigrid error iteration operator* as:

$$M_k = S_k^{\nu_2}\left[I_k - I_{k+1}^k\left\{I_{k+1} - (M_{k+1})^{\gamma}\right\} A_{k+1}^{-1} I_k^{k+1} A_k\right] S_k^{\nu_1} \tag{A.14}$$

such that, $k = 0, 1, 2, \cdots, l$ (0 represents finest, l coarsest grid), and $M_{l+1} = 0$.

Appendix B

On Eigenfunctions and Eigensymbols of Operators

In this appendix, we derive the *eigenfunctions and eigensymbols*[1] of the continuous and the discrete d-dimensional anisotropic stationary diffusion operator, with homogeneous Dirichlet boundary conditions. This is an important result and is used in quite a few places in the thesis.

B.1 Modes and Symbols (Continuous Operator)

$\boldsymbol{x}$ is a d-tuple representing a Cartesian point in $\mathbb{R}^d$, i.e., $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$. The continuous anisotropic stationary diffusion -eigenvalue problem is then given as:

$$L\varphi(\boldsymbol{x}) = \widetilde{L}\varphi(\boldsymbol{x}); \qquad \boldsymbol{x} \in \Omega = (0,1)^d \tag{B.1}$$

$$\varphi(\boldsymbol{x}) = 0; \qquad x_i \in \{0,1\}, (i = 1, 2, \cdots, d) \tag{B.2}$$

$$L = = -\sum_{i=1}^{d} \epsilon_i \frac{\partial^2}{\partial x_i^2}; \qquad \epsilon_i \in (0,1) \ni \sum_{i=1}^{d} \epsilon_i = 1.0$$

where $\varphi(\boldsymbol{x})$ represents the eigenfunctions, and $\widetilde{L}$ represents the associated eigensymbols of the continuous operator L.

We employ the *variable separable method* to solve this eigenvalue problem. Let

$$\varphi(\boldsymbol{x}) = \prod_{i=1}^{d} X_i(x_i) \tag{B.3}$$

Under this assumption φ is a product of d functions X_i, each depending on only a single variable x_i. When this is substituted in Equation B.1 and the result simplified,

[1] Eigenfunctions and eigensymbols are to continuous and discrete operators, what eigenvectors and eigenvalues (respectively) are to matrices.

we get:

$$-\sum_{i=1}^{d} \frac{\epsilon_i}{X_i} \frac{d^2}{dx_i^2} X_i = \widetilde{L} \tag{B.4}$$

As each term inside the summation depends on only a single variable, it is justified to break this equation into d linear ordinary differential equations, which gives:

$$-\frac{\epsilon_i}{X_i} \frac{d^2}{dx_i^2} X_i = \nu_i \quad (i = 1, 2, \cdots, d) \tag{B.5}$$

where ν_i are constants with the condition that $\sum_{i=1}^{d} \nu_i = \widetilde{L}$. Solving these ODEs give d solutions as follows:

$$X_i = C_1 e^{\sqrt{-(\nu_i/\epsilon_i)}\, x_i} + C_2 e^{-\sqrt{-(\nu_i/\epsilon_i)}\, x_i} \quad (i = 1, 2, \cdots, d) \tag{B.6}$$

Depending on $c_i = \nu_i/\epsilon_i$ this results in:

$$X_i = A_i \sinh(\sqrt{-c_i}\, x_i) + B_i \cosh(\sqrt{-c_i}\, x_i) \qquad c_i < 0 \tag{B.7}$$

$$X_i = A_i \sin(\sqrt{c_i}\, x_i) + B_i \cos(\sqrt{c_i}\, x_i) \qquad c_i > 0 \tag{B.8}$$

Homogeneous boundary conditions rule out the solution represented by Equation B.7. The boundary conditions take the form:

$$X_i(0) = 0, \;\; X_i(1) = 0$$

Substitution in the solution given by Equation B.8, gives:

$$B_i = 0, \; A_i \sin(\sqrt{c_i}) = 0, \Rightarrow \sqrt{c_i} = l_i \pi \quad \text{where } l_i \in \mathbb{Z}^+$$

Thus the solution represented by Equation B.8 takes the form:

$$X_i(x_i) = A_i \sin(l_i \pi x_i)$$

Substituting these d solutions in Equation B.3, we have:

$$\varphi(\boldsymbol{x}) \equiv \prod_{i=1}^{d} \sin(l_i \pi x_i) \tag{B.9}$$

Note that the constant multiple $\prod_{i=1}^{d} A_i$ has been conveniently removed from the above representation. The legitimacy of the step stems from the result that all constant multiples of eigenfunctions are eigenfunctions of the original operator, with associated eigensymbols scaled by their constant factors. We recognize that $\varphi(\boldsymbol{x})$ in Equation B.9 are the eigenfunctions of the continuous Laplace operator as well as the anisotropic operator. Substituting $\varphi(\boldsymbol{x})$ in Equation B.1, results in the eigensymbols $\widetilde{L}$:

$$\widetilde{L} = \pi^2 \sum_{i=1}^{d} \epsilon_i \, l_i^2 \tag{B.10}$$

B.2 Modes and Symbols (Discrete Operators)

Let L_h^{2o} and L_h^{4o} represent the $O(h^2)$ and the $O(h^4)$-long-stencil FDM discretizations of L, the continuous d-dimensional Laplacian. Along each dimension, the unit square domain is discretized into N divisions, so that $h = \frac{1}{N}$ gives the mesh size. The eigenfunctions of the discrete operators are therefore approximated by the eigenfunctions of the continuous operator, given by Equation B.9. This approximation is denoted by $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x})$ and is given by:

$$\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) = \prod_{i=1}^{d} \sin\left(\frac{\theta_i\, x_i}{h}\right); \quad \text{where } \theta_i = l_i h\, \pi, \quad l_i = 1, 2, \cdots, N-1 \tag{B.11}$$

B.2.1 Eigensymbols of L_h^{2o} ($O(h^2)$ Central FDM)

The complete d-dimensional stencil ($(2d+1)$ star) is a sum of the individual $1d$ stencils. For this reason, we will apply the k^{th} $1d$ stencil to the representation in Equation B.11, and sum the result over k. (Overline represents precedence of operation).

$$\begin{aligned}
L_h^{2o}\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) &= \sum_{k=1}^{d} \frac{\epsilon_k}{h^2} [-1 \quad \underline{2} \quad -1] \left\{ \prod_{i=1}^{d} \sin\left(\frac{\theta_i\, jh}{h}\right) \right\}; \quad j = 0, 1, \cdots, N \\
&= \sum_{k=1}^{d} \frac{\epsilon_k}{h^2} \left[-\sin(\theta_k\, \overline{j-1}) \prod_{\substack{i=1 \\ i\neq k}}^{d} \sin(\theta_i\, j) + 2 \prod_{i=1}^{d} \sin(\theta_i\, j) \right. \\
&\qquad \left. - \sin(\theta_k\, \overline{j+1}) \prod_{\substack{i=1 \\ i\neq k}}^{d} \sin(\theta_i\, j) \right] \\
&= \sum_{k=1}^{d} \frac{2\epsilon_k}{h^2} [1 - \cos(\theta_k)] \prod_{i=1}^{d} \sin(\theta_i\, j) \\
&= \widetilde{L}_h^{2o}(\boldsymbol{\theta}) \prod_{i=1}^{d} \sin(\theta_i\, j)
\end{aligned}$$

Therefore the eigensymbols of the discrete Laplacian in this case are:

$$\widetilde{L}_h^{2o}(\boldsymbol{\theta}) = \frac{2}{h^2} [1 - \sum_{k=1}^{d} \epsilon_k \cos(\theta_k)] \tag{B.12}$$

or alternately:

$$\widetilde{L}_h^{2o}(\boldsymbol{\theta}) = \frac{4}{h^2} \sum_{k=1}^{d} \epsilon_k \sin^2\left(\frac{\theta_k}{2}\right) \tag{B.13}$$

B.2.2 Eigensymbols of L_h^{4o} ($O(h^4)$ FDM Long-stencil)

The $1d$ $O(h^4)$ FDM long-stencil is considered in this part. This leads to:

$$L_h^k \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) = \sum_{k=1}^{d} \frac{1}{12h_k^2} \begin{bmatrix} 1 & -16 & \underline{30} & -16 & 1 \end{bmatrix} \left\{ \prod_{i=1}^{d} \sin\left(\frac{\theta_i \; j_i h_i}{h_i} \right) \right\}.$$

Trigonometric considerations very similar to the last part, lead to the eigensymbols:

$$\widetilde{L}_h^{4o}(\boldsymbol{\theta}) = \frac{1}{6h^2} \left[15 - \sum_{k=1}^{d} \epsilon_k (16 \cos\theta_k - \cos 2\theta_k) \right], \tag{B.14}$$

or alternately

$$\widetilde{L}_h^{4o}(\boldsymbol{\theta}) = \frac{1}{3h^2} \sum_{k=1}^{d} \epsilon_k \left[16 \sin^2 \frac{\theta_k}{2} - \sin^2 \theta_k \right]. \tag{B.15}$$

Appendix C

Matrix-Techniques for ω-RB Jacobi

ω-RB Jacobi can be implemented in several ways depending on the test platform. We have here a method that is suitable for iterations over a solution vector as opposed to explicit update in a loop. The scheme that we present in this appendix is developed to bring about the odd-even (red-black) partitioning with minimal manipulation of the grid-points.

We store three vectors in this scheme (two of which have just half the storage requirement as the first one). The first vector is the unmanipulated vector (henceforth called the *main-vector*) containing values for all grid points (red as well as black), the second vector has the values at the black points ejected out and replaced with zeros. Symmetrically, the third has the values at the red positions ejected out and replaced by zeros. This ejection-process is actually where our injection operators (henceforth called *ejectors*) fit in.

First we construct the partition of the main vector storing the red and the black parts, then we carry out the first partial ω-Jacobi sweep by updating only the red part. This new red part along with the previously stored black part represents the main vector after the first partial sweep. Carrying out the second partial sweep in exactly the same manner, now for the black part instead, gives one ω-RB Jacobi iteration.

We present two *injection* operators, one for points of each color (even/odd category). We denote these *ejectors* by $\mathcal{E}_R$, and $\mathcal{E}_B$, with

$$\mathcal{E}_R = \left(\bigoplus_{i=1}^{d} \eta_{(d-i+1)} \right) \ mod \ 2,$$

$$\mathcal{E}_B = (\mathcal{E}_R + 1) \ mod \ 2.$$

$\bigoplus$ is the cumulative tensor sum of η_i, which counts the interior points along the i^{th} dimension, i has the reverse order (from d to 1) to match the lexicographic layout of the grid points. Due to space limitations we can only provide a $2d$ example, although this formulation is true in general for an abstract higher dimension d.

Example 2: Consider a $2d$ grid $\mathbf{G} = [4, 5]$. In all we have 12 interior points which when counted in the lexicographic order, appear as follows:

$$\mathbf{u} = [u_{11}\ \ u_{12}\ \ u_{13}\ \ u_{14}\ \ u_{21}\ \ u_{22}\ \ u_{23}\ \ u_{24}\ \ u_{31}\ \ u_{32}\ \ u_{33}\ \ u_{34}]^T.$$

Evidently, $\boldsymbol{G}$, the collection of all the points, has the following partition:

$$\begin{aligned} \boldsymbol{G}_R &= \{u_{11}, u_{13}, u_{22}, u_{24}, u_{31}, u_{33}\}, \\ \boldsymbol{G}_B &= \{u_{12}, u_{14}, u_{21}, u_{23}, u_{32}, u_{34}\}. \end{aligned}$$

Therefore according to our scheme:

$$\begin{aligned} \mathbf{u}_R &= [\ \ u_{11}\ \ 0\ \ u_{13}\ \ 0\ \ 0\ \ u_{22}\ \ 0\ \ u_{24}\ \ u_{31}\ \ 0\ \ u_{33}\ \ 0\ \ ]^T, \\ \mathbf{u}_B &= [\ \ 0\ \ u_{12}\ \ 0\ \ u_{14}\ \ u_{21}\ \ 0\ \ u_{23}\ \ 0\ \ 0\ \ u_{32}\ \ 0\ \ u_{34}\ \ ]^T \end{aligned}$$

and η_i in this case would be:

$$\eta_1 = [1\ \ 2\ \ 3\ \ 4]^T \quad \& \quad \eta_2 = [1\ \ 2\ \ 3]^T$$

which leads to:

$$\begin{aligned} \mathcal{E}_R &= (\eta_2 \oplus \eta_1)\ \ mod\ \ 2 \\ &= [2\ \ 3\ \ 4\ \ 5\ \ 3\ \ 4\ \ 5\ \ 6\ \ 4\ \ 5\ \ 6\ \ 7]^T\ \ mod\ \ 2 \\ &= [0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1]^T \\ \therefore \quad \mathcal{E}_B &= [1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0]^T. \end{aligned}$$

These red and black point ejectors now can be used to partition the grid as described. Once this partition is obtained, ω-RB Jacobi relaxation sweeps are trivial to perform, in that, they are no different than their $2d$ counterparts.

Appendix D

Derivation of the Discrete Flux Balance Equations

To illustrate completely how the discrete coarse grid operator is built, we derive, in this appendix, the discrete flux-balance equations for Nodes 2 and 12 of Chapter 6, Figure 6.3. In the following, k indicates a face of the control volume; specifically, the values 0, 1, 2, 3 point to the south, east, north, and west faces, respectively. F_m^k denotes the outward flux through the k^{th} face of the m^{th} control volume. f_m^Ω is the source function, f^Ω, computed at (x_m, y_m), which is the Cartesian position of node m. h_m^x and h_m^y are the width and the height (i.e. mesh-sizes along x and y) of the m^{th} control volume, respectively.

For Node $m = 2$ of Figure 6.3(b), Equation (6.4) yields:

$$F_2^0 + F_2^1 + F_2^2 + F_2^3 = f_2^\Omega \, (h_2^x \, h_2^y)$$

or

$$-F_1^2 + F_2^1 + F_2^2 + F_2^3 = f_2^\Omega \, (h_2^x \, h_2^y) \tag{D.1}$$

where

$$\begin{aligned}
F_1^2 &= a_2\Big(x_1,\ y_1 + \tfrac{h_1^y}{2}\Big)\,(u_1 - u_2)\,\Big(\tfrac{2h_1^x}{h_1^y + h_2^y}\Big)\\
F_2^1 &= a_1\Big(x_2 + \tfrac{h_2^x}{2},\ y_2\Big)\,\{u_2 - (c_1 u_{11} + c_2 u_{12})\}\,\Big(\tfrac{2h_2^y}{h_2^x + h_{12}^x}\Big)\\
F_2^2 &= a_2\Big(x_2,\ y_2 + \tfrac{h_2^y}{2}\Big)\,(u_2 - u_3)\,\Big(\tfrac{2h_2^x}{h_2^y + h_3^y}\Big)\\
F_2^3 &= a_1\Big(x_2 - \tfrac{h_2^x}{2},\ y_2\Big)\,\Big\{u_2 - f^\Gamma\Big(x_2 - \tfrac{h_2^x}{2},\ y_2\Big)\Big\}\,\Big(\tfrac{2h_2^y}{h_2^x}\Big)
\end{aligned}$$

Let

$$\begin{aligned}
\beta_1^2 &= a_2\Big(x_1,\ y_1 + \tfrac{h_1^y}{2}\Big)\Big(\tfrac{2h_1^x}{h_1^y + h_2^y}\Big), \qquad \beta_2^1 = a_1\Big(x_2 + \tfrac{h_2^x}{2},\ y_2\Big)\Big(\tfrac{2h_2^y}{h_2^x + h_{12}^x}\Big),\\
\beta_2^2 &= a_2\Big(x_2,\ y_2 + \tfrac{h_2^y}{2}\Big)\Big(\tfrac{2h_2^x}{h_2^y + h_3^y}\Big), \qquad \beta_2^3 = a_1\Big(x_2 - \tfrac{h_2^x}{2},\ y_2\Big)\Big(\tfrac{2h_2^y}{h_2^x}\Big)
\end{aligned}$$

and

$$\beta_2 = \beta_1^2 + \beta_2^1 + \beta_2^2 + \beta_2^3$$

Then, from the flux balance equation (D.1), we get:

$$\beta_2\, u_2 = f_2^{\Omega}\,(h_2^x\, h_2^y) + \beta_1^2\, u_1 + \beta_2^1\,(c_1 u_{11} + c_2 u_{12}) + \beta_2^2\, u_3 + \beta_2^3\, f^{\Gamma}\left(x_2 - \frac{h_2^x}{2},\ y_2\right) \quad \text{(D.2)}$$

Similarly, for node $m = 12$, we get from (6.4):

$$F_{12}^0 + F_{12}^1 + F_{12}^2 + F_{12}^3 = f_{12}^{\Omega}\,(h_{12}^x\, h_{12}^y)$$

or

$$-F_{11}^2 + F_{12}^1 + F_{12}^2 - (F_2^1 + F_3^1) = f_{12}^{\Omega}\,(h_{12}^x\, h_{12}^y) \quad \text{(D.3)}$$

where

$$\begin{aligned}
F_{11}^2 &= a_2\Big(x_{11},\ y_{11} + \tfrac{h_{11}^y}{2}\Big)\,(u_{11} - u_{12})\,\Big(\tfrac{2h_{11}^x}{h_{11}^y + h_{12}^y}\Big)\\
F_{12}^1 &= a_1\Big(x_{12} + \tfrac{h_{12}^x}{2},\ y_{12}\Big)\,\{u_{12} - u_{18})\}\,\Big(\tfrac{2h_{12}^y}{h_{12}^x + h_{12}^x}\Big)\\
F_{12}^2 &= a_2\Big(x_{12},\ y_{12} + \tfrac{h_{12}^y}{2}\Big)\,(u_{12} - u_{13})\,\Big(\tfrac{2h_{12}^x}{h_{12}^y + h_{13}^y}\Big)\\
F_3^1 &= a_1\Big(x_3 + \tfrac{h_3^x}{2},\ y_3\Big)\,\{u_3 - (c_3 u_{12} + c_4 u_{13})\}\,\Big(\tfrac{2h_3^y}{h_3^x + h_{12}^x}\Big)
\end{aligned}$$

Let

$$\begin{aligned}
\beta_{11}^2 &= a_2\Big(x_{11},\ y_{11} + \tfrac{h_{11}^y}{2}\Big)\Big(\tfrac{2h_{11}^x}{h_{11}^y + h_{12}^y}\Big), \qquad \beta_{12}^1 &= a_1\Big(x_{12} + \tfrac{h_{12}^x}{2},\ y_{12}\Big)\Big(\tfrac{2h_{12}^y}{h_{12}^x + h_{18}^x}\Big),\\
\beta_{12}^2 &= a_2\Big(x_{12},\ y_{12} + \tfrac{h_{12}^y}{2}\Big)\Big(\tfrac{2h_{12}^x}{h_{12}^y + h_{13}^y}\Big), \qquad \beta_3^1 &= a_1\Big(x_3 + \tfrac{h_3^x}{2},\ y_3\Big)\Big(\tfrac{2h_3^y}{h_3^x + h_{12}^x}\Big)
\end{aligned}$$

and

$$\beta_{12} = \beta_{11}^2 + \beta_{12}^1 + \beta_{12}^2 + c_2\,\beta_2^1 + c_3\,\beta_3^1$$

Then from the flux balance equation (D.3), we get:

$$\beta_{12}\, u_{12} = f_{12}^{\Omega}\,(h_{12}^x\, h_{12}^y) + \beta_{11}^2\, u_{11} + \beta_{12}^1\, u_{18} + \beta_{12}^2\, u_{13} + \beta_2^1\, c_1\, u_{11} + \beta_3^1\, c_4\, u_{13} \quad \text{(D.4)}$$

Equations (D.2) and (D.4) lay emphasis on the way we ensure flux balance in the system. The flux through the south face of Cell 2 is the same quantity computed as Cell 1's north-face flux. Similarly, the flux through the west face of Cell 12 is the cumulative flux flowing in from its west neighbours, i.e. Cells 2 and 3. We reiterate that traversal of the L-strips is in such a fashion that the required fluxes are already available if they have been computed previously. Particularly in connection with this example, note that Cell 1 would be treated earlier than Cell 2, hence its north-face flux is available for use as Cell 2's south-face flux. Similarly Cell 2 and Cell 3 (due to the enumeration scheme) would be treated earlier than Cell 12 and, therefore, their east-face fluxes are already available to add up into the west-face flux of Cell 12.

Bibliography

[1] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31: 333–390, 1977.

[2] K. Stüben and U. Trottenberg. *Multigrid Methods: fundamental algorithms, model problem analysis and applications*. Springer Berlin, 1982.

[3] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*. Cambridge University Press, Cambridge, 1984.

[4] W. Hackbusch. *Multi-grid methods and applications*. Springer series in computational mathematics. Springer-Verlag Berlin, 1985.

[5] Jr. J. E. Dendy. Black box multigrid for systems. *Applied Mathematics and Computation*, 19: 57–74, 1986.

[6] P. Wesseling. *An Introduction to Multigrid Methods*. Pure and Applied Mathematics. John Wiley & Sons, 1992.

[7] I. Yavneh. Multigrid smoothing factors for red-black gauss–seidel relaxation applied to a class of elliptic operators. *SIAM Journal on Numerical Analysis*, 32: 1126–1138, 1995.

[8] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.

[9] R. Wienands and W. Joppich. *Practical Fourier Analysis for Multigrid Methods*. Numerical Insights. Chapman & Hall/CRC, Boca Raton, Florida, USA, 2005.

[10] R. U. Seydel. *Tools for computational finance*. Springer-Verlag Berlin Heidelberg, 2006.

[11] W. A. Mulder. Geophysical modeling of 3d electromagnetic diffusion with multigrid. *Computing and Visualization in Science*, 11: 129–138, 2008.

[12] T. N. Rescigno, M. Baertschy, W. A. Isaacs, and C. W. McCurdy. Collisional breakup in a quantum system of three charged particles. *Science*, 286: 2474–2479, 1999.

[13] A. Bayliss, C. I. Goldstein, and E. Turkel. On accuracy conditions for the numerical computation of waves. *Journal of Computational Physics*, 59: 396–404, 1985.

[14] Y.A. Erlangga, C.W. Oosterlee, and C. Vuik. A novel multigrid based preconditioner for heterogeneous helmholtz problems. *SIAM Journal on Scientific Computing*, 27: 1471–1492, 2006.

[15] J. Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114: 185–200, 1994.

[16] S.D. Gedney. An anisotropic perfectly matched layer absorbing media for the truncation of FDTD latices. *IEEE Transactions on Antennas and Propagation*, 44: 1630–1639, 1996.

[17] R. Wienands and C. W. Oosterlee. On three grid Fourier analysis for multigrid. *SIAM Journal on Scientific Computing*, 23(2): 651–671, 2001.

[18] R. Wienands. *Extended local Fourier analysis for multigrid: Optimal smoothing, coarse grid correction, and preconditioning*. PhD thesis, University of Cologne, June 2001.

[19] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[20] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[21] P. Wesseling. Theoretical and practical aspects of a multigrid method. Technical report, Report NA-37, Department of Mathematics, Delft University of Technology, Delft, The Netherlands, 1980.

[22] P.W. Hemker. On the order of prolongations and restrictions in multigrid procedures. *Journal of Computational and Applied Mathematics*, 32: 423–429, 1990.

[23] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34: 581–613, 1992.

[24] J. H. Bramble. *Multigrid methods*. Pitman Research Notes in Mathematics, Series 294. Longman, Harlow, 1993.

[25] W. Mulder. A new multigrid approach to convection problems. Technical report, CAM Report 88-04, 1988.

[26] N. H. Naik and J. Van Rosendale. The improved robustness of multigrid elliptic solvers based on multiple semicoarsened grids. *SIAM Journal on Numerical Analysis*, 30(1):215–229, 1993.

[27] E. Morano, D. J. Mavriplis, and V. Venkatakrishnan. Coarsening strategies for unstructured multigrid techniques with application to anisotropic problems. In N. D. Melson, T. A. Manteuffel, S. F. McCormick, and C. C. Douglas, editors, *Seventh Copper Mountain Conference on Multigrid Methods*, volume CP 3339, pages 591–606, Hampton, VA, 1996. NASA.

[28] J. Larsson, F.S. Lien, and E. Yee. Conditional semicoarsening multigrid algorithm for the Poisson equation on anisotropic grids. *Journal of Computational Physics*, 208: 368–383, 2005.

[29] H. bin Zubair, C.W. Oosterlee, and R. Wienands. Multigrid for high-dimensional elliptic partial differential equations on non-equidistant grids. *SIAM Journal on Scientific Computing*, 29(4): 1613–1636, 2007.

[30] I. Yavneh. On red-black sor smoothing in multigrid. *SIAM Journal on Scientific Computing*, 17: 180–192, 1996.

[31] C. Reisinger and G. Wittum. On multigrid for anisotropic equations and variational inequalities. *Computing and Visualization in Science*, 2004.

[32] J. Elf, P. Lötstedt, and P. Sjöberg. Problems of high dimension in molecular biology. In *Proceedings of the 19th GAMM-Seminar Leipzig*, pages 21–30, 2003.

[33] G. Beylkin and J.M. Martin. Algorithms for numerical analysis in high dimensions. *SIAM Journal on Scientific Computing*, 26(6): 2133–2159, 2005.

[34] H. Yserentant. Sparse grid spaces for the numerical solution of the electronic schrödinger equation. *Numerische Mathematik*, 101: 381–389, 2005.

[35] S.R. Elias, G.D. Stubley, and G.D. Raithby. An adaptive agglomeration method for additive correction multigrid. *International Journal for Numerical Methods in Engineering*, 40: 887–903, 1997.

[36] G. Horton and S. Vandewalle. A space-time multigrid method for parabolic partial differential equations. *SIAM Journal on Scientific Computing*, 16(4):848–864, 1995.

[37] H.W. Steeb. *Kronecker Product of Matrices and Applications*. Wissenschaftsverlag, 1991.

[38] K. Stüben and U. Trottenberg. On the construction of fast solvers for elliptic equations. Technical report, von Karman Institute for Fluid Dynamics, Rhode-Saint-Genese, Belgium, 1982.

[39] C.A. Thole and U. Trottenberg. Basic smoothing procedures for the multigrid treatment of elliptic 3-d operators. *Applied Mathematics and Computation*, 19: 333–345, 1986.

[40] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, New Jersey, 1961.

[41] H.J. Bungartz and M. Griebel. *Sparse Grids*. *Acta Numerica*, pages 147–269, May 2004.

[42] M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In *Proceedings of the IMACS International Symposium on Iterative Methods in Linear Algebra*, pages 263–281. Elsevier, Amsterdam, 1992.

[43] C. Zenger. Sparse grids. In *Proceedings of the 6th GAMM seminar, Notes on numerical fluid mechanics*, volume 31, 1990.

[44] H. A. Van Der Vorst. A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13: 631–644, 1992.

[45] C.W. Oosterlee and T. Washio. An evaluation of parallel multigrid as a solver and as a preconditioner for singularly perturbed problems. *SIAM Journal on Scientific Computing*, 19: 87–110, 1998.

[46] R. Wienands, C.W. Oosterlee, and T. Washio. Fourier analysis of gmres(m) preconditioned by multigrid. *SIAM Journal on Scientific Computing*, 22: 582–603, 2000.

[47] Y. K. Kwok. *Mathematical models of financial derivatives*. Springer Finance. Springer-Verlag, Singapore, second edition, 1998.

[48] Z. You-lan, W. Xiaonan, and C. I-Liang. *Derivative Securities and Difference Methods*. Springer Finance. Springer Science+Business Media Inc. USA, 2004.

[49] P. Hofmann. Asymptotic expansions of the discretization error of boundary value problems of the laplace equation in rectangular domains. *Numerische Mathematik*, 9:302–322, 1967.

[50] H.J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Pointwise convergence of the combination technique for the Laplace equation. *East-West Journal of Numerical Mathematics.*, 2:21–45, 1994.

[51] C. C. W. Leentvaar and C. W. Oosterlee. On coordinate transformation and grid stretching for sparse grid pricing of basket options. *Journal of Computational and Applied Mathematics*, 222:193–209, 2008.

[52] W.L. Briggs, V.E. Henson, and S.F. McCormick. *A Multigrid Tutorial*. SIAM, 2000.

[53] W. A. Mulder. A high-resolution Euler solver based on multigrid, semi-coarsening and defect-correction. *Journal of Computational Physics*, 100: 91–104, 1992.

[54] B. Lee, S. F. McCormick, B. Philip, and D. J. Quinlan. Asynchronous fast adaptive composite-grid methods: numerical results. *SIAM Journal on Scientific Computing*, 25(2):682–700 (electronic), 2003.

[55] M. J. Berger and R. J. Leveque. Adaptive mesh refinement using wave propagation algorithms for hyperbolic systems. *SIAM Journal on Numerical Analysis*, 35(6):2298–2316, 1998.

[56] M. Barad and P. Colella. A fourth-order accurate local refinement method for Poisson's equation. *Journal of Computational Physics*, 209(1):1–18, 2005.

[57] D. F. Martin, P. Colella, M. Anghel, and F. J. Alexander. Adaptive mesh refinement for multiscale nonequilibrium physics. *Computing in Science and Engineering*, 7(3):24–31, 2005.

[58] E. Haber and S. Heldmann. An octree multigrid method for quasi-static Maxwell's equations with highly discontinuous coefficients. *Journal of Computational Physics*, 223(2):783–796, 2007.

[59] E. Haber, S. Heldmann, and J. Modersitzki. An octree method for parametric image registration. *SIAM Journal on Scientific Computing*, 29(5):2008–2023, 2007.

[60] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods, Frontiers in Applied Mathematics*, volume 3, pages 73–130. SIAM, Philadelphia, 1987.

[61] T.F. Chan, J. Xu, and L. Zikatanov. An agglomeration multigrid method for unstructured grids. *Contemporary Mathematics, AMS.*, 218: 67–81, 1998.

[62] A. Janka. Smoothed aggregation multigrid for a Stokes problem. *Computing and Visualization in Science*, 11:169–180, 2008.

[63] P. Wesseling. *Principles of Computational Fluid Dynamics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[64] L. P. Eisenhart. Enumeration of potentials for which one-particle schroedinger equations are separable. *Physical Review*, 74(1):87–89, Jul 1948.

[65] G. E. Bowman. *Essential quantum mechanics*. Oxford University Press, 2008.

[66] W. Vanroose, F. Martin, T. N. Rescigno, and C. W. McCurdy. Complete photo-induced breakup of the h_2 molecule as a probe of molecular electron correlation. *Science*, 310: 1787–1789, 2005.

[67] F. Ihlenburg and I. Babuska. Finite element solution to the helmholtz equation with high wave numbers. *Computers and Mathematics with Applications*, 30:9–37, 1995.

[68] J. M. G-Jordan, S. Rojas, M. F-Villegas, and J. E. Castillo. A new second order finite difference conservative scheme. *Divulgaciones Mathemáticas*, 13(1):107–122, 2005.

[69] P. Sonneveld and M. B. van Gijzen. IDR(s): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM Journal on Scientific and Statistical Computing*, 31(2): 1035–1062, 2008.

[70] A. Brandt and I. Livshits. Wave-ray multigrid method for standing wave equations. *ETNA*, 6: 162–181, 1997.

[71] A. L. Laird and M. B. Giles. Preconditioned iterative solution of the 2d Helmholtz equation. Technical report, Report 02/12, Oxford Computer Laboratory, Oxford, UK, 2002.

[72] N. Umetani, S.P. MacLachlan, and C. W. Oosterlee. A multigrid-based shifted-laplacian preconditioner for a fourth order helmholtz discretization. *Numerical Linear Algebra with Applications*, 16:603–626, 2009.

[73] I. Livshits and A. Brandt. Accuracy properties of the wave-ray multigrid algorithm for Helmholtz equations. *SIAM Journal on Scientific Computing*, 28(4): 1228–1251, 2006.

Publications

H. bin Zubair, C.W. Oosterlee and R. Wienands.
Multigrid for High-dimensional Elliptic Partial Differential Equations on Non-Equidistant Grids, SIAM Journal of Scientific Computing. Vol. 29, No. 4, March 2007, pp 1613-1636.

H. bin Zubair, S. P. MacLachlan and C. W. Oosterlee.
A geometric multigrid method based on L-shaped coarsening for PDEs on stretched grids, (accepted for publication in Numerical Linear Algebra with Applications, Wiley Interscience.)
Preprint: http://ta.twi.tudelft.nl/TWA_Reports/08-15.pdf

H. bin Zubair, C.C.W. Leentvaar and C.W. Oosterlee.
Efficient d-Multigrid Preconditioners for Sparse-Grid Solution of High-Dimensional Partial Differential Equations, International Journal of Computer Mathematics. IJCM. Vol 84, No. 8, August 2007, 1129-1147.

H. bin Zubair.
On Multigrid for High-Dimensional Anisotropic Partial Differential Equations. One of the top three winners of the Student Paper Competetion held at the 13th Copper Mountain Conference on Multigrid Methods, Copper Mountain, Colorado, USA, March 2007.
http://amath.colorado.edu/faculty/copper/2007/program.html

H. bin Zubair and C.W. Oosterlee.
Multigrid Preconditioners for Bi-CGSTAB for the Sparse-Grid Solution of High-Dimensional Anisotropic Diffusion Equation. 3rd International Conference on 21st Century Mathematics, March 2007, School of Mathematical Sciences, GC University, Lahore, Pakistan. Paper selected to appear in The Journal of Prime Research in Mathematics, Vol 3, November 2007, ISSN 1817-2725, Abdus Salam School of Mathematical Sciences, GCU, Lahore.

Curriculum Vitae

Hisham bin Zubair was born on the 24^{th} of September, 1974 in Karachi, Pakistan. For his primary and secondary schooling he attended the St. Patrick's High School 1990, and the D. J. Science College 1992, Karachi, Pakistan. He did his B.Sc. 1996 and subsequently M.Sc. 1997 in Applied Mathematics, from the University of Karachi. Since his M.Sc. in 1997, he has basically been an educator at various institutions in Karachi. They include Sir Syed University of Engineering and Technology SSUET 1998 and the Aga Khan Higher Secondary School 1999. Over a period of two years 2001-2003 he acquired an M.Sc. in Computer Sciences from the University of Karachi while teaching undergrad math at the Textile Institute of Pakistan 2005. Since 2005, he has been in the Netherlands, pursuing his PhD in Scientific Computing in the Numerical Analysis group, at the Delft University of Technology.

He has presented various papers in seminars and conferences related to applied scientific computing. One such venture at the 13^{th} Copper Mountain Multigrid conference resulted in a shield for ranking among the top three PhD student papers. Other prominent activities include his participation and presentation in ECCOMAS 2006, EMG 2008 and the 14^{th} Copper Mountain MG conference. Currently, he is actively involved in researching iterative solution methods for the indefinite Helmholtz equation that arises in many applied problems.

Debating and poetry formed Hisham's extra-academic activities during his student life. At the department of Mathematics (University of Karachi), he won the first prize in an Urdu poetry contest. He has a very unusual combination of interests which include poetry, scientific computing and pigeon fancy. As a hobby he keeps and breeds Indian Mookee, and Racing Homer pigeons in his Karachi home.