Institut für Informatik,
Technische Universität München

Multigrid methods for matrices with structure and applications in image processing

Jochen Staudacher

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Rudolf Bayer, Ph.D.
Prüfer der Dissertation:	
	1. Univ.-Prof. Dr. Thomas Huckle
	2. Univ.-Prof. Dr. Heike Faßbender

Die Dissertation wurde am 22.04.2002 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 03.07.2002 angenommen.

Abstract

Multigrid methods are among the fastest algorithms for the solution of linear systems of equations $Ax = b$. For many problems the computational efforts for the multigrid solution of the linear system are of the same complexity as the multiplication of a vector with the matrix A. This thesis deals with multigrid algorithms for structured linear systems. Particular focus is put on Toeplitz matrices, i.e. matrices with entries constant along diagonals. Via the FFT a dense Toeplitz matrix $A \in \mathbb{R}^{n \times n}$ can be multiplied with a vector $x \in \mathbb{R}^n$ in $O(n \log n)$ operations.

This work presents new efficient multigrid algorithms of complexity $O(n \log n)$ for solving dense Toeplitz systems corresponding to nonnegative generating functions with isolated zeros of finite order. For the first time natural coarse grid operators are employed in the context of multigrid for Toeplitz systems. That way Toeplitz structure can be preserved on the coarse grids.

Afterwards ideas for sparse Toeplitz matrices are carried over to general banded linear systems. The so called 'Matrix Multilevel Method' (MML) uses matrix-dependent prolongation and restriction operators based on a good upper bound for the maximum eigenvalue of the system matrix A. Unlike most multigrid approaches, the MML is not restricted to M-matrices. Plenty of numerical examples for elliptic problems with oscillatory or discontinuous coefficients are given for which standard multigrid deteriorates whereas the MML leads to optimal computational performance.

Then I take a look at Fredholm integral equations of the first kind as they arise e.g. from image deblurring. I extend an idea for a multigrid algorithm by R. Chan, T. Chan and W. Wan who proposed to use preconditioned conjugate gradients as a smoother. Again, an efficient $O(n \log n)$ multigrid algorithm for the dense structured linear systems from image deblurring can only be obtained if we get the coarse grid operators via rediscretization.

Finally, a new preconditioner for sparse structured matrices arising from the problem of high-resolution image reconstruction with multisensors is proposed, analyzed and integrated into a powerful software package. The numerical experiments show that the new method is efficient and leads to the desired $O(n)$ speed of convergence.

Zusammenfassung

Mehrgittermethoden gehören zu den schnellsten Verfahren für die Lösung eines linearen Gleichungssystems $Ax = b$. Für viele Probleme ist der Aufwand für eine Mehrgitterlösung von der gleichen Komplexität wie die Multiplikation eines Vektors mit der Matrix A. Diese Arbeit handelt von Mehrgitteralgorithmen für struktierte Gleichungssysteme. Ein besonderer Schwerpunkt liegt dabei auf Toeplitz-Matrizen, das sind Matrizen mit konstanten Einträgen entlang allen Diagonalen. Via FFT kann eine vollbesetzte Toeplitz-Matrix $A \in \mathbb{R}^{n\times n}$ mit einem Vektor $x \in \mathbb{R}^n$ in $O(n \log n)$ Operationen multipliziert werden.
Diese Arbeit präsentiert neue effiziente Mehrgitteralgorithmen mit Komplexität $O(n \log n)$ für vollbesetzte Toeplitz-Systeme, die zu nichtnegativen generierenden Funktionen mit isolierten Nullstellen endlicher Ordnung gehören. Dabei werden zum ersten Mal im Kontext von Multigrid für Toeplitz-Systeme natürliche Grobgitteroperatoren eingesetzt. Auf diese Weise bleibt die Toeplitz-Struktur auf den Grobgittern erhalten.
Anschließend werden Ideen für dünnbesetzte Toeplitz-Matrizen auf allgemeine bandbeschränkte Systeme übertragen. Die sogenannte 'Matrix Multilevel Methode' (MML) verwendet matrix-abhängige Prolongations- und Restriktionsoperatoren basierend auf einer guten oberen Schranke für den größten Eigenwert der Matrix A. Anders als die meisten Mehrgitteransätze ist die MML nicht auf M-Matrizen beschränkt. Es wird eine Vielzahl numerischer Beispiele für elliptische Probleme mit oszillierenden oder unstetigen Koeffizienten präsentiert, für die Standard-Multigrid versagt, während die MML optimales Laufzeitverhalten zeigt.
Dann widme ich mich Fredholmschen Integralgleichungen 1.Art, wie sie etwa bei der Bildentzerrung auftreten. Ich erweitere eine Idee für einen Mehrgitteralgorithmus von R. Chan, T. Chan und W. Wan, die vorschlugen, ein vorkonditioniertes CG-Verfahren als Glätter zu verwenden. Erneut kann ein effizienter $O(n \log n)$ Multigridalgorithmus für die vollbesetzten strukturierten Systeme aus der Bildentzerrung nur erreicht werden, falls man die Grobgitteroperatoren mittels Rediskretisierung bestimmt.
Schließlich wird ein neuer Vorkonditionierer für dünnbesetzte strukturierte Matrizen aus dem Problem der hochauflösenden Bildrekonstruktion mit Multisensoren vorgeschlagen, analysiert und in ein leistungsfähiges Softwarepaket integriert. Die numerischen Resultate zeigen, dass das neue Verfahren effizient arbeitet und auf die gewünschte $O(n)$ Konvergenzgeschwindigkeit führt.

Acknowledgements

Above all, I would like to express my deep gratitude to my advisor Prof. Thomas Huckle for his supervision on this research project which started in February 1999. His insightful comments, thorough approach and patience helped make the whole effort worthwhile. I shall never forget all his help and friendliness over the last years – and I would like to emphasize how much I appreciated his way of acting simultaneously as both a supervisor and a friend. Finally, I would like to thank him for plenty of humourous conversations we had over the last years.

I would also like to thank Prof. Huckle and his colleague Prof. Heike Faßbender for having turned Technical University Munich into a lively place for research in numerical linear algebra, e.g. through inviting other scientists in our field to give talks in the Numerical Analysis seminar on Monday afternoon. I am using this chance to thank Prof. Faßbender for the good contact we had since she joined TU Munich in spring 2000 – and also for her willingness to referee this Ph.D. thesis.
Finally, I am grateful to Prof. Folkmar Bornemann for having established the Numerical Analysis seminar in autumn 1998.

Part of this work was written during a research visit to UCLA from September 2000 to February 2001: In particular, I would like to thank Prof. Tony Chan for having invited me to join his research group for half a year. From him I learned what hospitality is all about – and I wish to express my gratitude to him for plenty of fruitful discussions on both image processing and multilevel methods and for having made my stay so profitable. With his Ph.D. student Andy Yip I made a close friend together with whom I enjoyed plenty of dinners on or near the UCLA campus. Andy helped me a lot to familiarize with techniques in image processing and got me started working on the problem of high-resolution image reconstruction with multisensors by handing me a MATLAB software package. I would also like to thank Prof. Björn Engquist for having introduced me to the theory of homogenization and for his encouragement to test the "Matrix Multilevel Method" for

problems with highly oscillatory coefficients. Clearly, the case study in section 5.7 would not exist without his motivation. Finally, I thank the DAAD for having made my stay in Los Angeles possible via their grant D/00/20283.

Over the last years I enjoyed plenty of discussions on my work with academic colleagues either at conferences or in connection with the Numerical Analysis seminar mentioned previously. There are certainly far too many to name all of them here. However, I would like to emphasize on the role of Prof. Raymond Chan from the Chinese University of Hongkong whom I met first during one of his visits to Germany in July 1999 and another time at a conference in Rousse (Bulgaria) in June 2000. I very much appreciate in how much detail he explained me his results and experience on multigrid for image deblurring summarized in the paper [22]. Our conversations in Rousse helped me to get started on the research which now forms Chapter 6 of this thesis.

This is also the place to thank three former academic teachers who started my fascination for the fields of numerical linear algebra and multilevel algorithms when I was an M.Sc. student at Oxford University in the academic year 1997/1998, i.e. my former supervisor Dr. David Handscomb and Dr. Andy Wathen and Prof. Nick Trefethen. I am very glad still to be in close contact will all of them. Finally, I also wish to mention Priv.-Doz. Dr. Eugen Schäfer whose lectures at Munich University aroused my interest for the fields of Numerical Analysis and Scientific Computing in the first place.

I have very much enjoyed my Ph.D. studies at Technical University Munich and I wish to acknowledge the excellent environment in terms of computer equipment and libraries. Furthermore, I am very grateful to Technical University Munich for having funded my research with a doctoral scholarship from April 2001 onwards.
Finally, I would also like to thank all the colleagues of the "Chair for Informatics V" for the pleasant atmosphere over the last three years; in particular, this refers to my former room-mate Stefan Achatz and to Michael "Michi" Riss with whom I am currently sharing an office. Last but not least, I would like to thank Prof. Christoph Zenger for having invited me to join his group, for plenty of interesting discussions on multigrid methods – and for providing such a rewarding environment in which to study.

Munich, April 2002 Jochen Staudacher

Contents

Chapter 1

Outline of this thesis

This thesis is devoted to studying multigrid algorithms for different types of structured matrices. Mainly, our notion of "structured matrix" is twofold, i.e. we will investigate on Toeplitz matrices as well as on sparse banded matrices. These two concepts are also reflected in the structure of the thesis itself: Chapter 4 presents new multigrid algorithms for dense Toeplitz matrices. Chapter 5 discusses the "Matrix Multilevel Method" which can be interpreted as a generalization of ideas from the previous chapter for sparse Toeplitz matrices to general sparse banded linear systems. Chapter 6 deals with multigrid methods for dense Toeplitz systems from image deblurring. Finally, Chapter 7 presents a new optimal order method for sparse banded linear systems from the problem of high-resolution image reconstruction with multisensors – and these banded matrices can be interpreted as perturbed Toeplitz systems. The major goal is to point out how different multigrid components have to be adjusted for the different types of structured linear systems in order to obtain a feasible multigrid algorithm.

In the following few pages we would like to give the reader an overview of the contents and ideas of this Ph.D. thesis.

Chapters 2 and 3 introduce basic concepts concerning multigrid algorithms and Toeplitz matrices:
In Chapter 2 we will first explain the celebrated V- and W-cycle algorithms for the numerical solution of a linear system $Ax = b$. The role of smoothers and grid transfer operators (for prolongation and restriction) will be intro-

duced and the distinction between natural and Galerkin coarse grid operators will be discussed. It will be pointed out how multigrid cycles can also be employed as preconditioners for Krylov subspace methods, like e.g. the Conjugate Gradient algorithm. Furthermore, additive preconditioners like BPX [9] and MDS [120] will be introduced very briefly. This is also the place for a few words about multigrid convergence theory, although we would like to emphasize that the focus of this thesis lies on the development of efficient algorithms and on the presentation of numerical results showing their effectivity – and not on the underlying mathematical theory. In that sense, our way of discussing multigrid ideas is strongly influenced by the recent 600-page monograph by Trottenberg, Oosterlee and Schüller [108].
In Chapter 3 we then present some important facts about Toeplitz matrices, i.e. matrices with entries constant along diagonals. The relation of these matrices and their corresponding generating functions will be discussed as well as equidistribution results concerning their spectrum. It will also be pointed out how a dense Toeplitz matrix $A \in \mathbb{R}^{n \times n}$ can be multiplied with a vector in $O(n \log n)$ time using the idea of circulant embedding and the Fast Fourier Transform (FFT). Hence we use the chance to say a little more about circulant preconditioners, emphasizing on the optimal circulant preconditioner [28] by T. Chan. Analogously, the previous ideas will be carried over to Block Toeplitz matrices with Toeplitz blocks (BTTB matrices), i.e. the bivariate counterpart of Toeplitz systems, and to Block Circulant preconditioners with Circulant blocks (BCCB preconditioners).

Chapter 4 discusses multigrid algorithms for Toeplitz matrices belonging to nonnegative generating functions. First, we repeat a few results from existing work by R. Chan and collaborators [104], [25] as well as by Serra and Fiorentino [47], [48]. We then focus on multigrid algorithms for Toeplitz matrices corresponding to nonnegative 2π-periodic generating functions with a single zero $x_0 \in]-\pi, \pi]$ of finite order. In this case we can scale our Toeplitz systems via diagonal matrices such that the zero is shifted to the origin. This strategy provides two major advantages: Firstly, we can use the same operators for prolongation and restriction on every level. Secondly, we can employ a natural coarse grid operator, i.e. our coarse grid representation A^{coarse} is nothing but a Toeplitz matrix of half size corresponding to the original generating function. Thus Toeplitz structure is preserved on the coarse levels. All previous approaches on multigrid for Toeplitz systems have been employing

a Galerkin operator

$$A^{coarse} = R * A * P$$

with P and R denoting the operators for prolongation and restriction, respectively. In that case A^{coarse} is not in general guaranteed to be Toeplitz. The numerical experiments show that our multigrid algorithms with natural coarse grid operators work out very well and hence can be regarded as an algorithmic improvement.
This approach carries over very easily to nonnegative 2π-periodic generating functions with a finite number of equidistant zeros of finite order in $]-\pi,\pi]$. Again, we first perform a diagonal scaling and then we may use a natural coarse grid operator. It is remarkable to see how natural coarse grid operators can be constructed taking carefully into account the orders of the zeros of the generating function – and how the resulting multigrid cycles lead to an efficient computational performance.
Natural coarse grid operators can also be successfully employed within multigrid algorithms for Block Toeplitz matrices with Toeplitz Blocks (BTTB matrices) belonging to a nonnegative 2π-periodic generating function (in two variables) with a single zero $x_0 \in]-\pi,\pi]^2$. In the context of BTTB matrices the impact of using a natural coarse grid operator is still more eminent than in the univariate case, because the perturbations of the Toeplitz structure in Galerkin coarse grid operators are in general much more severe. Finally, we also give a new phenomenological characterization of the well-known difficulties encountered in multigrid approaches for indefinite BTTB matrices.

Chapter 5 introduces the "Matrix Multilevel Method" (MML) which is based on a purely matrix dependent description of multigrid methods. This approach has been published by Huckle and the author in the paper [72]. However, we will in this chapter significantly extend the results from [72] and discuss the properties of the MML in much more detail.
The "Matrix Multilevel Method" is motivated by observations on the choice of prolongation and restriction operators in multigrid algorithms for sparse Toeplitz systems and tries to carry over these ideas to general sparse banded problems: The formulation of multilevel methods as singular matrix extensions via generating systems leads to the description of the method as a preconditioned iterative scheme, and illuminates the importance of the used prolongation and restriction operator for the related preconditioner. We define the matrix dependent black box restriction C by shifting the original

matrix A in the form $B = \alpha I - A$ and picking out every second column to $C = B(:, 2 : 2 : n)$. Here, α has to be chosen as a good upper bound of the largest eigenvalue of A. By this mapping the related preconditioner enlarges the small eigenvalues while the maximum eigenvalue remains nearly unchanged. Although we derive our method in an additive setting, we can most certainly also use the new prolongations/restrictions in multiplicative algorithms. Furthermore, we observe that the MML usually works best if we employ diagonally scaled coarse grid matrices. Our test results for one-dimensional problems are very promising: We give various numerical examples where multigrid with standard prolongation/restriction deteriorates whereas our method shows optimal behaviour. Finally, we explain why for M-matrices using $B = abs(A)$ instead of $B = \alpha I - A$ within the MML we obtain equally good results.
Then the MML algorithms are carried over to separable and to general problems in higher dimensions: We present examples of elliptic problems with highly oscillatory or discontinuous coefficients for which the MML clearly outperforms standard multigrid. We also point out why it might be favourable to employ the MML without diagonal scalings for general elliptic problems. The chapter ends with a case study for elliptic problems with highly oscillatory problems: We compare our methods to work by Engquist and Luo [82], [45] who proposed to use a discretization of the corresponding homogenized equation to obtain the coarse grid representation. The basic result of our experiments is that proper matrix-dependent transfer operators and Galerkin coarsening gives better results than the homogenized coarse grid operators. In other words: Variational coarsening gives good homogenization. And, other than in the Toeplitz context, the variational coarse grid operators come out to be the clear winners for this type of problems.

Chapter 6 studies integral equations of the first kind as they arise from image deblurring. We start with a very little introduction to inverse problems and Tikhonov regularization and first take a look at Toeplitz matrices from one-dimensional deblurring. It is plain that the strategies from Chapter 4 can not be carried over, because we deal with a "zero of infinite order".
Instead, we extend an algorithm by R. Chan, T. Chan and J. Wan [22] who proposed to use preconditioned conjugate gradients for smoothing. Again, the algorithms can significantly be improved by introducing a natural coarse grid operator, i.e. we recommend to get the coarse grid matrices by discretizing the original integral equation on a coarser mesh. We illustrate why

standard linear interpolation should be used for prolongation and why trivial injection is the most evident choice for restriction. Our technique carries over in a straightforward manner to BTTB matrices from multivariate integral equations and thus leads to a practical algorithm for deblurring images. We test our solvers on images subject to atmospheric turbulence blur.
To us it is very important to establish our algorithms in the context of the so called "multigrid method of the second kind" proposed by Hackbusch [62], ch. 16, in the eighties. The multigrid method of the second kind is a very fast scheme for the solution of linear systems of the form

$$(\lambda I - K)x = b$$

arising from discrete Fredholm integral equations of the second kind and it uses Richardson smoothing. In the above equation, K is a discretized integral operator, I denotes the identity and $\lambda \in \mathbb{R}$ is a positive parameter. We point out how our deblurring algorithms can be seen as a generalization of the multigrid method of the second kind for the case of small λ by replacing Richardson relaxation by a semi-iterative smoother.

Chapter 7 is devoted to the problem of high-resolution image reconstruction with multisensors [24]: There a high-resolution image is reconstructed from four undersampled, shifted, degraded and noisy low-resolution images. In a mathematical model the boundary conditions play a vital role: If we assign Dirichlet boundary conditions, i.e. we assume a dark background around our reconstructed image, we need to solve a (regularized) system of normal equations $A^T A$ with a sparse BTTB matrix A. However, this leads to boundary artifacts in the reconstructed image which can be overcome by using reflecting (i.e. Neumann) boundary conditions. Then A becomes a sparse Block Toeplitz-plus-Hankel matrix with Toeplitz-plus-Hankel blocks. In practice, there will always be calibration errors within our four multisensors which can be measured by the camera manufacturer and lead to small perturbations in the entries of A.
Previously R. Chan, T. Chan, M. Ng and collaborators had been proposing very successful fast cosine transform based preconditioners for the problem in question (see e.g. [24], [27]). On the other hand, no $O(n)$ preconditioners for these sparse problems had been developed. However, our multigrid algorithms from Chapter 4 can again not be carried over: In the simple case of Dirichlet boundary conditions and no calibration errors we would be confronted with a BTTB system with an infinite number of zeros of order 2.

Anyway, we are able to present a simple and effective $O(n)$ preconditioner based on the structure of the linear systems: The idea is that the matrix structure allows for a helpful "analytic factorization". Various numerical examples underline that our preconditioner leads to an efficient optimal order performance.

Our observations on the choice of different multigrid components could be summarized as follows:
From Chapter 5 we learn that for elliptic problems with discontinuous or highly oscillatory coefficients matrix-dependent transfer operators and variational coarsening are the clear winners. However, for Toeplitz matrices natural coarse grid representations are a viable simpler alternative. As long as the ill-conditioning is "moderate" – in the sense of second-order discretizations of elliptic problems or Toeplitz systems corresponding to generating functions with a finite number of zeros of finite order – then standard stationary iterative methods (like e.g. Richardson, Jacobi or Gauss-Seidel) are the smoothers of choice. In terms of Toeplitz matrices ill-posed problems can be viewed as connected to a "zero of infinite order" (first order integral equations from image deblurring problems) or to an infinite number of zeros of finite order (high-resolution image reconstruction). In both cases, the "classical" multigrid methodology no longer works out and has to be adjusted. For the image deblurring problem a non-stationary smoother, i.e. preconditioned conjugate gradients, does the job; for the sparse linear systems from the problem of high-resolution image reconstruction there is just a simpler way than multigrid to derive an $O(n)$ preconditioner.

This thesis has lead to two publications which both summarize joint research efforts of the author and his supervisor:

[P1] T. Huckle, J. Staudacher: Matrix Multilevel Methods and Preconditioning, preprint, 20 pages, to appear in BIT 42(4) in December 2002.

introduces the "Matrix Multilevel Method" and forms the basis for our investigations in Chapter 5.

[P2] T. Huckle, J. Staudacher: Multigrid Preconditioning and Toeplitz matrices, preprint, 23 pages, submitted to ETNA, February 2002. Recommended for publication after slight modification by the referees.

summarizes the major results of Chapter 4 and Chapter 6.

Chapter 2

The basic principles of multigrid algorithms

Multigrid methods have first been systematically analyzed in the seventies, in particular through work by Brandt [10] and Hackbusch [61]. Since then multigrid algorithms have become a popular tool for the fast iterative solution of large linear systems of equations $Ax = b$ – and we may well regard multigrid as the first computationally successful example of modern hierarchical methods in the field of Scientific Computing.
The following subsections give a brief introduction to multigrid in order to equip the reader with some background information needed to understand chapters 4, 5 and 6. Our overview is strongly influenced by the recent book by Trottenberg, Oosterlee and Schüller [108].

2.1 Model problems

Let us first introduce two simple model problems. They will not only guide us through this introduction to multigrid, but also appear frequently in other parts of this work.

2.1.1 One-dimensional Laplacian

An important model problem throughout this chapter will be the uniform finite difference discretization of Poisson's equation with homogeneous Dirichlet boundary conditions on the unit line.

Example 1 (Discrete 1D-Laplacian with Dirichlet boundary)
We want to solve the two-point boundary value problem

$$\begin{array}{lll} -u_{xx} = f & in & \Omega =]0,1[, \\ u(x) = 0 & on & \partial\Omega \end{array}$$

with $f : \Omega \to \mathbb{R}$ *given.*
For the discretisation of the problem let us consider a uniform grid with mesh size $h = \frac{1}{n}$ *and let*

$$x_j = j \cdot h, \qquad j = 0, \ldots n$$

Now let u_j *denote an approximation to the solution* u *at the point* $P = (x_i)$*. Then our unknowns are the interior grid points and they form a vector* v *given by*

$$v = (u_1, u_2, \ldots, u_{n-1})^T \in \mathbb{R}^{n-1}$$

The finite difference approximation

$$u_{xx}(x_i) = \frac{u(x_{i-1}) + u(x_{i+1}) - 2u(x_i)}{h^2}$$

is second order accurate and leads to a linear system

$$A_{n-1} * v = g$$

There the matrix $A_{n-1} \in \mathbb{R}^{(n-1)\times(n-1)}$ *is a tridiagonal matrix of the form*

$$\mathbf{A}_{n-1} = \begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 2 \end{pmatrix} \tag{2.1}$$

From now on we will abbreviate: $A_{n-1} = tridiag(-1, 2, -1)$*.*
The right hand side g *is the vector*

$$g = (g_1, g_2, \ldots, g_{n-1}) \in \mathbb{R}^{n-1} \, with \qquad g_j = h^2 \cdot f(x_j)$$

It is well known that the matrix (2.1) has the convenient property that there are closed expressions for its eigenvalues and eigenvectors (see Demmel [33], p. 268):

Lemma 1 *The eigenvalues of the matrix* $A_{n-1} = tridiag(-1, 2, -1) \in \mathbb{R}^{(n-1)\times(n-1)}$ *are given by*

$$\lambda_j = 2(1 - \cos(\frac{\pi j}{n})) \tag{2.2}$$

for $j = 1, \ldots, n-1$. *The corresponding eigenvectors are* z_j *with components*

$$z_j(k) = \sqrt{\frac{2}{n}} \sin(\frac{jk\pi}{n}) \tag{2.3}$$

for $j, k = 1, \ldots, n-1$. z_j *has unit 2-norm. Let* $Z = [z_1, \ldots, z_{n-1}]$ *be the orthogonal matrix formed with the eigenvectors and* $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$ *the diagonal matrix of eigenvalues, then we can write* $A_{n-1} = Z\Lambda Z^T$. *In other words: The matrix* A_{n-1} *can be diagonalized in* $O(n \log n)$ *time via the fast sine transform.*

2.1.2 Two-dimensional Laplacian

Plenty of the reasoning in this subsection will be based on the above model problem Example 1; due to its simple structure it quickly gives valuable insight. However, tridiagonal matrices can well be solved with $O(n)$ requirements for computing time and storage simply by direct methods like e.g. the so-called "Thomas algorithm" (cf. [83], pp. 23) and there is no need for multigrid. In the context of PDEs multigrid is hence intrinsically seen as a technique for higher dimensional problems. Here is the corresponding two-dimensional model problem, i.e. a uniform finite difference discretization of Poisson's equation with Dirichlet boundary conditions on the unit square.

Example 2 (Discrete 2D-Laplacian with Dirichlet boundary)
The simplest example of a boundary value problem for a linear partial differential equation might be Poisson's equation with Dirichlet boundary conditions on the unit square, i.e. we are looking at

$$\begin{array}{lll} -\triangle u(x,y) = f(x,y) & in & \Omega = \{(x,y) \in \mathbb{R}^2 : 0 < x, y < 1\}, \\ u(x,y) = 0 & on & \partial\Omega \end{array}$$

The Laplace operator $\triangle$ *denotes*

$$\triangle u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

and $f : \Omega \to \mathbb{R}$ is given.
For the discretisation of the boundary value problem let us consider a uniform grid with mesh size $h = \frac{1}{n}$ and let

$$x_i = i \cdot h, \qquad y_j = j \cdot h, \qquad i, j = 0, \ldots n$$

Let now $u_{i,j}$ denote an approximation to the solution u at the interior grid point $P = (x_i, y_j), 1 \leq i, j \leq n-1$, and let us order the unknowns lexicographically, i.e. row-wise. Then our unknowns form a vector v given by

$$v = (u_{1,1}, \ldots, u_{n,1}, u_{1,2}, \ldots, u_{n,2}, \ldots, u_{1,n}, \ldots, u_{n,n})^T \in \mathbb{R}^{(n-1)^2}$$

The finite difference approximation

$$\triangle u(x_i, y_j) = \frac{u(x_{i-1}, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 4u(x_i, y_j)}{h^2}$$

is second order accurate and leads to a linear system

$$A_{(n-1)^2} * v = g$$

The matrix $A_{(n-1)^2}$ is given by

$$\mathbf{A_{(n-1)^2}} = \begin{pmatrix} D_{n-1} & -I_{n-1} & & & \\ -I_{n-1} & D_{n-1} & -I_{n-1} & & \\ & \ddots & \ddots & \ddots & \\ & & -I_{n-1} & D_{n-1} & -I_{n-1} \\ & & & -I_{n-1} & D_{n-1} \end{pmatrix} \tag{2.4}$$

Here $I_{n-1} \in \mathbb{R}^{(n-1)\times(n-1)}$ denotes the identity and D_{n-1} is given by

$$\mathbf{D}_{n-1} = \begin{pmatrix} 4 & -1 & & & 0 \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ 0 & & & -1 & 4 \end{pmatrix} \tag{2.5}$$

In order words: $A_{(n-1)^2}$ is a block tridiagonal sparse matrix. The right hand side g is the vector

$$g = (g_{1,1}, \ldots, g_{n,1}, g_{1,2}, \ldots, g_{n,2}, \ldots, g_{1,n}, \ldots, g_{n,n})^T \in \mathbb{R}^{(n-1)^2}$$
$$\text{with} \qquad g_{i,j} = h^2 \cdot f(x_i, y_j)$$

In order to connect the matrix (2.4) from Example 2 to the 1D Laplacian (2.1) we would like to introduce Kronecker products (see e.g. [33], p. 274):

Definition 1 (Kronecker product of matrices)
Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$ be two matrices.
Then $A \otimes B$, the Kronecker product of A and B, is the $(m\cdot p)$-by-$(n\cdot q)$ matrix

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{1,1} \cdot B & \dots & a_{1,n} \cdot B \\ \vdots & & \vdots \\ a_{m,1} \cdot B & \dots & a_{m,n} \cdot B \end{pmatrix}$$

Lemma 2 *(see [33], pp. 275) The matrix $A_{(n-1)^2}$ given by (2.4) is related to the one-dimensional Laplacian A_{n-1} from (2.1) via*

$$A_{(n-1)^2} = I_{n-1} \otimes A_{n-1} + A_{n-1} \otimes I_{n-1}$$

Using the diagonal matrix $\Lambda = diag(\lambda_1, \dots, \lambda_n)$ of the eigenvalues λ_j given by (2.2) and the matrix $Z = [z_1, \dots, z_n]$ of the eigenvectors z_j given by (2.3) from Lemma 1 we see

$$I_{n-1} \otimes A_{n-1} + A_{n-1} \otimes I_{n-1} = (Z \otimes Z) \cdot (I_{n-1} \otimes \Lambda + \Lambda \otimes I_{n-1}) \cdot (Z \otimes Z)^T$$

Hence (2.4) can be diagonalized by the two-dimensional fast sine transform and its (i,j)th eigenvalue is the $(i*(n-1)+j)$th entry of the diagonal matrix $I_{n-1} \otimes \Lambda + \Lambda \otimes I_{n-1}$, i.e.

$$\lambda_{i,j} = \lambda_i + \lambda_j \tag{2.6}$$

$Z \otimes Z$ is an orthogonal matrix whose $(i(n-1)+j)$th column, the corresponding eigenvector, is

$$z_{i,j} = z_i \otimes z_j. \tag{2.7}$$

2.2 The multigrid idea

2.2.1 The smoothing principle

In order to understand the basic principles of the multigrid method let us first take a look at simple **stationary iterative methods** for the solution of $Ax = b$ of the form

$$x^{(m+1)} = x^{(m)} - M * (A * x^{(m)} - b) \tag{2.8}$$

From (2.8) we recognize a number of well-known relaxation methods:

- Setting M equal to the identity leads to the **Richardson iteration**.
- Setting M equal to the inverse of the diagonal part of A leads to the **Jacobi iteration**.
- Setting M equal to the lower triangular part of A we recognize the **Gauss-Seidel iteration**.

Certainly, we can damp Richardson, Jacobi or Gauss-Seidel iterations by multiplying M with a parameter $\omega \in]0,1[$ – or even use a parameter $\omega \in]1,2[$ to produce an overrelaxation method like the celebrated SOR [111], [118].
Rewriting (2.8) in the form

$$x^{(m+1)} = G * x^{(m)} + Mb \qquad \text{with} \quad G = I - MA \tag{2.9}$$

we understand that the iterate $x^{(m)}$ is related to the initial guess $x^{(0)}$ via

$$x^{(m)} = G^m * x^{(0)} + N^{(m)}b \qquad \text{with} \quad N^{(m)} = \sum_{k=0}^{m-1} G^k M \tag{2.10}$$

As the solution of $Ax = b$ is a fixed point of the iteration we learn from (2.9) and (2.10) that the error $e^{(m)} = x - x^{(m)}$ after m iteration steps can be written in terms of the initial error $e^{(0)} = x - x^{(0)}$ as

$$e^{(m)} = G^m * e^{(0)}, \tag{2.11}$$

i.e. the matrix $G = I - MA$ governs the behaviour of the iteration (2.8) and is therefore usually referred to as its **iteration matrix**.

Now imagine that our matrix A is a sparse matrix resulting from an elliptic problem – let us say, it is the uniform discretization of the two-dimensional Laplacian (2.4) from Example 2.
Then running a few steps of a standard stationary iterative method on the linear system we observe that the oscillatory components of the error (i.e. the error components with a wavelength of the same order as the mesh size) are reduced very quickly. However, the method has severe problems in damping the slowly varying error components and this causes an overall slow convergence of the basic iterative method. In other words: The error becomes smooth, although it need not necessarily become small. We visualize this property of smoothing the error for the Gauss-Seidel method applied to the

two-dimensional Laplacian (2.4) from Example 2 in Figure 2.1.
For our model problem (2.4) we would like to recapitulate that in terms of its basis of eigenvectors $z_{i,j}$ given by (2.7) the problematic error components are associated with the smooth eigenvectors, i.e. both i and j are small. We can summarize our reasoning in form of the

Smoothing principle (see [108], sec. 1.4)
If standard stationary iterative methods (like Richardson, Jacobi or Gauss-Seidel) are appropriately applied to discrete elliptic problems they have a strong smoothing effect on the error of any approximation.

2.2.2 The coarse grid principle

The observations of the previous subsection about problematic smooth error components lead to the idea of employing coarser grids: A slowly varying error component certainly looks more oscillatory on a coarse grid – and thus we can expect our basic iterative method to do a much better job on a coarser grid. Furthermore, if we are able to approximate a smooth error component well on a coarse grid – let us say with the double mesh size – then dealing with it on the coarse grid will also be less expensive computationally. This idea can also be formulated as the

Coarse grid principle (see [108], sec. 1.4)
A smooth error term can be well approximated on a coarse grid. As there are substantially fewer grid points on the coarse grid it is plain that performing a number of relaxation steps on a coarse grid is much cheaper computationally than the corresponding fine grid procedure.

2.3 Twogrid and multigrid cycles

2.3.1 From concepts to algorithms

So far, we have introduced the two basic ideas of multigrid, i.e. the smoothing principle and the coarse grid principle. However, the reader might still not be happy with what we mean by "reducing the error on a coarse grid". Hence we first briefly outline the idea of a twogrid method hoping to guide the reader to the following precise algorithms:

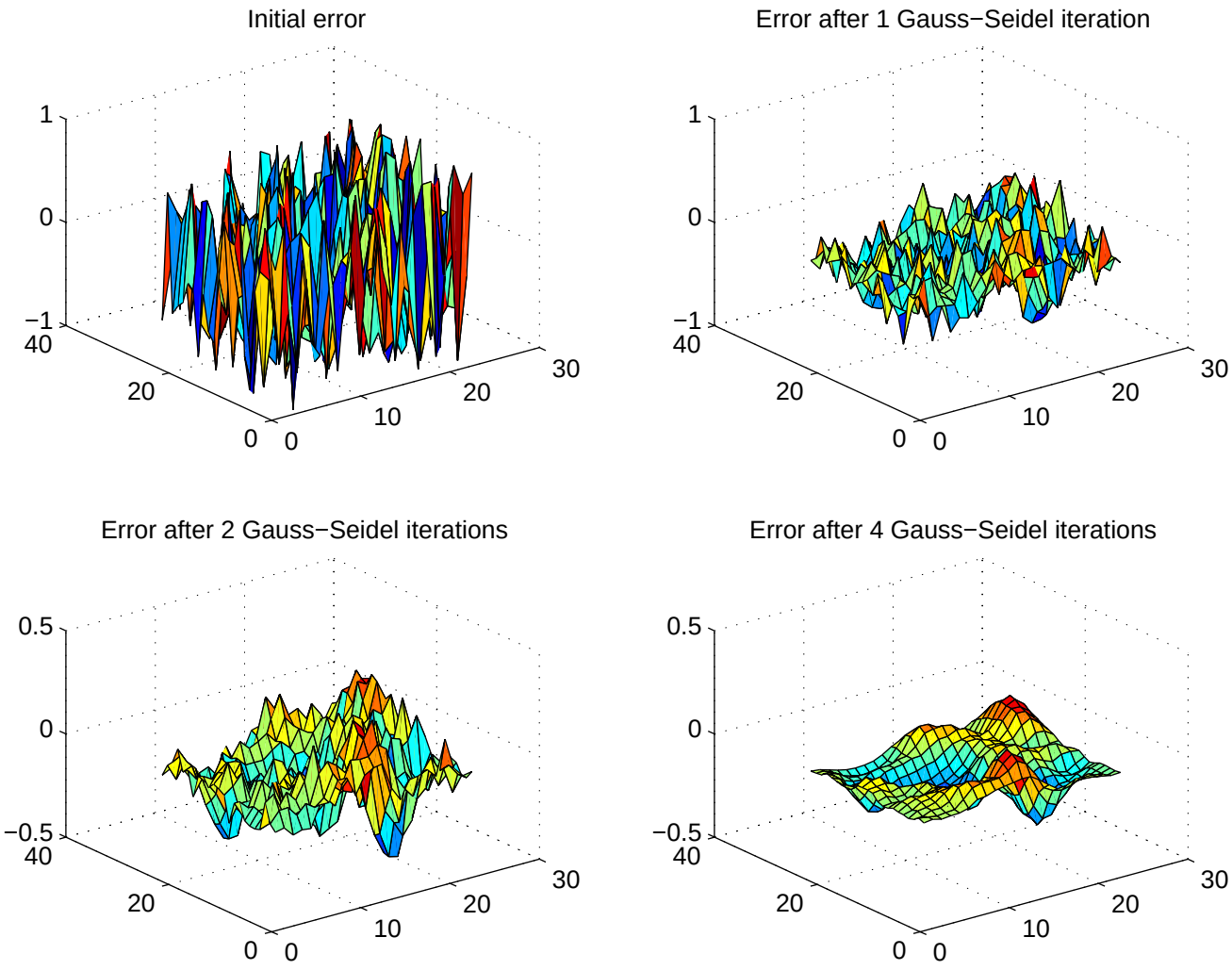

Figure 2.1: Smoothing effects of Gauss-Seidel iterations on the error for the discrete two-dimensional Laplace problem (2.4).

Let us think of our elliptic problem discretized on a sequence of grids

$$\Omega_h \longrightarrow \Omega_{2h} \longrightarrow \Omega_{4h} \longrightarrow \cdots$$

with growing mesh sizes $h, 2h, 4h, \ldots$, on which we can set up the linear systems

$$A_h u_h = f_h, \qquad A_{2h} u_{2h} = f_{2h}, \qquad \ldots$$

Now we treat the highly oscillatory error components by a basic iterative method – the so-called **smoother** – and reduce the slowly varying error components by **coarse grid correction**. Employing only a coarse and a fine grid we can formulate a twogrid step:
In order to solve the equation $A_h u_h = f_h$ on the fine grid Ω_h we first employ very few steps of our smoother to obtain an approximate solution $\tilde{u}_h$. Still, $\tilde{u}_h$ may differ from the exact solution u_h^* by a large – but relatively smooth – error $e_h := u_h^* - \tilde{u}_h$.
By coarse grid correction we approximate this smooth error e_h on the coarse grid Ω_{2h}: Thus we first compute the residual $r_h := f_h - A_h \tilde{u}_h$ and restrict it to the coarse grid $r_{2h} = R r_h$ employing the restriction operator R. Then we solve the error equation $A_{2h} v_{2h} = r_{2h}$ on the coarse grid and transfer the solution v_{2h} back to the fine grid $v_h = P v_{2h}$ using the prolongation operator P. Finally, we update our approximate solution $\tilde{u}_h := \tilde{u}_h + v_h$ and proceed iterating with our two-grid method.
We get a **multigrid cycle** by solving the error equation $A_{2h} v_{2h} = r_{2h}$ recursively by the same procedure on a sequence of coarser grids.

2.3.2 A twogrid algorithm

It is now time for us to formulate precise numerical algorithms. Our algorithms in this and the following subsection are mainly according to the book by Trottenberg, Oosterlee and Schüller [108], ch. 2, but the reader will also find excellent descriptions of multigrid algorithms e.g. in the books by Briggs [13],ch. 3, Greenbaum [54], ch. 12, or Hackbusch [62], ch. 3.
Let us begin with a detailed twogrid algorithm:

Algorithm 1 (Twogrid method)
Start with any initial guess $x^{(0)}$ for the solution of the linear system $A_h x = b$. Then proceed with the following iteration until stopping criterion is satisfied:
(1) Presmoothing:

Smooth using ν_1 steps of a stationary iterative method of the form (2.8), like e.g. Jacobi or Gauss-Seidel, in order to get a new iterate $\bar{x}^{(j)}$. We can write this formally as

$$\bar{x}^{(j)} = SMOOTH_1^{\nu_1}(x^{(j)}, A_h, b)$$

(2) Coarse grid correction:
(a) Compute the residual: $r_h = A_h x^{(j)} - b$
(b) Restrict the residual r_h, i.e. use the restriction operator R to perform a fine-to-coarse grid transfer: $r_{2h} = R r_h$
(c) Solve the error equation on the coarse grid Ω_{2h}, i.e. use a direct (or iterative) method to find the solution of $A_{2h} e_{2h} = r_{2h}$
(d) Interpolate the correction e_{2h}, i.e. use the prolongation operator P to perform a coarse-to-fine grid transfer: $\bar{e} = P e_{2h}$
(e) Compute the corrected approximation: $x^{(j+\frac{1}{2})} = \bar{x}^{(j)} + \bar{e}$

(3) Postsmoothing (optional):
Smooth again using ν_2 steps of a stationary iterative method of the form (2.8) to obtain a new iterate $x^{(j+1)}$:

$$x^{(j+1)} = SMOOTH_2^{\nu_2}(x^{(j+\frac{1}{2})}, A_h, b)$$

The following remark can be very helpful to understand and analyze the twogrid operator:

Remark 1 *The two-grid operator given by Algorithm 1 can be expressed in matrix terms as*

$$M_{TG} = G_2^{\nu_2} K G_1^{\nu_1} \tag{2.12}$$

where G_i^{ν} with $i = 1, 2$ stands for performing ν steps of a stationary iterative smoother (2.8) with iteration matrix $G_i = I - M_i A_h$ on the fine grid and

$$K = I - P A_{2h}^{-1} R A_h \tag{2.13}$$

expresses the coarse grid correction.

2.3.3 Multigrid algorithms: V-cycles and W-cycles

The problem of solving the error equation $A_{2h} e_{2h} = r_{2h}$ in step 2 (c) of Algorithm 1 might still be expensive and involve a huge number of unknowns. As we have already said, the key to get an efficient multigrid algorithm is to employ the twogrid idea **recursively**.

In order to perform any multigrid idea we first need a a finite sequence of coarser and coarser grids

$$\Omega_{h_l}, \Omega_{h_{l-1}}, \ldots \Omega_{h_0}$$

Here we associate the coarsest grid employed with the mesh size h_0 (index 0) and the index l with the mesh size we used to discretize our problem: $h_l = h$. Like in [108], sec. 2.4, we will, for simplicity, replace the index h_k by k.
The following algorithm summarizes both the so-called **V-cycle** and the so-called **W-cycle** algorithms:

Algorithm 2 (MULTIGRID — V-cycle and W-cycle algorithms)
Start with any initial guess $x^{(0)}$ for the solution of the linear system $A_h x = b$. First set $k = l$, $A_k = A_h$, $b_k = b$ and $j = 0$. Then proceed with the following multigrid cycle iteration

$$x_k^{(j+1)} = MGCYC(k, \gamma, x_k^{(j)}, A_k, b_k^{(j)}, \nu_1, \nu_2) \tag{2.14}$$

until stopping criterion is satisfied:
(1) Presmoothing:
Smooth using ν_1 steps of a stationary iterative method of the form (2.8), like e.g. Jacobi or Gauss-Seidel, in order to get a new iterate $\bar{x}_k^{(j)}$. We can write this formally as

$$\bar{x}_k^{(j)} = SMOOTH_1^{\nu_1}(x_k^{(j)}, A_k, b_k^{(j)})$$

(2) Coarse grid correction:
(a) Compute the residual: $r_k^{(j)} = Ax_k^{(j)} - b_k^{(j)}$
(b) Restrict the residual $r_k^{(j)}$, i.e. use the restriction operator R_k^{k-1} to perform a fine-to-coarse grid transfer: $r_{k-1}^{(j)} = R_k^{k-1} r_k^{(j)}$
(c) Compute an approximate solution to the error equation

$$A_{k-1} e_{k-1}^{(j)} = r_{k-1}^{(j)} \tag{2.15}$$

by

- *If $k = 1$, then we are on the coarsest level: Now use a direct (or iterative) solver for (2.15);*
- *If $k > 1$, then we have not yet reached the coarsest level: Solve (2.15) approximately performing γ "k-grid cycles" using the zero initial guess, i.e.*

$$e_{k-1}^{(j)} = MGCYC^{\gamma}(k-1, \gamma, 0, A_{k-1}, r_{k-1}^{(j)}, \nu_1, \nu_2)$$

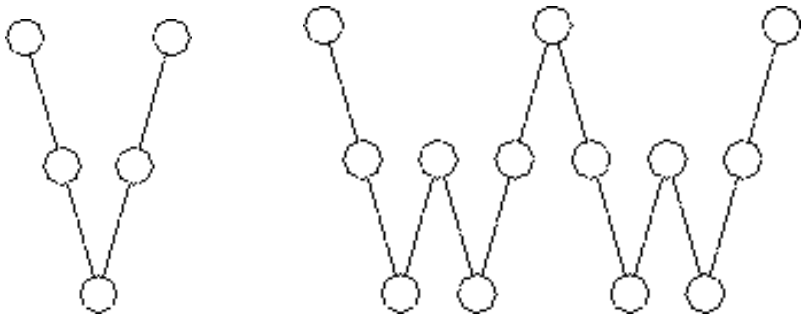

Figure 2.2: A V-cycle on the left and a W-cycle on the right.

- *For $\gamma = 1$ this algorithm is called a* **V-cycle**.
- *For $\gamma = 2$ we obtain a* **W-cycle**.

(d) Interpolate the correction $e_{k-1}^{(j)}$, i.e. use the prolongation operator P_{k-1}^{k} to perform a coarse-to-fine grid transfer: $\bar{e}_k^{(j)} = P_{k-1}^{k} e_{k-1}^{(j)}$
(e) Compute the corrected approximation: $x_k^{(j+\frac{1}{2})} = \bar{x}_k^{(j)} + \bar{e}_k^{(j)}$

(3) Postsmoothing (optional):
Smooth again using ν_2 steps of the above stationary iterative method to obtain a new iterate $x_k^{(j+1)}$:

$$x_k^{(j+1)} = SMOOTH_2^{\nu_2}(x_k^{(j+\frac{1}{2})}, A_k, b_k^{(j)})$$

Although it is already very general, Algorithm 2 still leaves room for further generalizations: First of all, we can certainly allow a different number of pre- and postsmoothing steps on different levels k. (On the other hand, note that Algorithm 2 already expresses that presmoother and postsmoother need not be identical.) Secondly, variations to the multigrid structure itself are possible: We are definitely not forced to program V-cycles and W-cycles – however, these two variants of multigrid cycles are the most common ones and we will not use other types of multigrid solvers in this thesis.
In Figure 2.2 we give a pictorial description of a V-cycle and a W-cycle for the case of 3 different levels of resolution.

2.3.4 Flavour of multigrid

Having introduced multigrid algorithms formally, we would like to summarize their underlying idea as the

Flavour of multigrid (see [108], sec. 1.5)
Assume we have an appropriate smoother for a given class of problems, i.e. we have a stationary iterative method (2.8) which damps the highly oscillatory error frequencies well. Furthermore, assume we have appropriate grid transfer operators and coarse grid representations.
Then employing the smoother on different grid levels (i.e. different levels of resolution) gives a fast reduction of the corresponding high frequency components – and as this process covers all ranges of frequencies, we can obtain a rapid reduction of the overall error.

2.4 The Multigrid components: Smoothers, transfer operators, coarse grid matrices

So far we have formulated precise twogrid and multigrid algorithms. However, we have not yet told the reader how to choose proper prolongations P and restrictions R. As for the coarse grid representations $A_{2h}, A_{4h}, \ldots$ we have simply assumed them to exist, but have not commented on how they ought to be picked. Certainly, we also need to say how to distinguish between good and bad smoothers. In this section the model problems from 2.1 will come out to be very helpful.

2.4.1 Smoothers

By smoothing we understand the capability of an iterative method (2.8) to reduce the highly oscillatory error components effectively.
Following [13], ch. 2, and [108], sec. 2.1, let us study the matrix (2.1) from Example 1, i.e. the one-dimensional Laplacian with eigenvalues and eigenvectors given by (2.2) and (2.3). We know that the subspace of high frequency components is spanned by the eigenvectors corresponding to the large eigenvalues, i.e.

$$\lambda_j = 2(1 - \cos(\frac{\pi j}{n})) \quad \text{with} \qquad \frac{n}{2} \leq j \leq (n-1).$$

Let us analyze the damped Jacobi method with damping parameter ω applied to problem (2.1). Its iteration matrix $R_{n-1}(\omega)$ is given by

$$R_{n-1}(\omega) = I_{n-1} - \frac{\omega}{2} A_{n-1},$$

its eigenvectors coincide with the eigenvectors of A_{n-1} given by (2.3) and the corresponding eigenvalues are

$$\mu_j = 1 - \frac{\omega}{2}\lambda_j = 1 - 2\omega(\sin(\frac{j\pi}{2n}))^2 \tag{2.16}$$

for $j = 1, \ldots, n-1$. First of all, we see from (2.16) that for the low frequencies we can not expect the damped Jacobi method to reduce the corresponding slowly varying error components effectively: It is obvious that for large n the reduction factor on the smoothest error components approaches 1 – and this strongly underlines the statements made in 2.2.
The **smoothing factor** represents the worst factor by which the high frequency error components are reduced per iteration step. Analyzing (2.16) for $\frac{n}{2} \leq j \leq (n-1)$ shows that the smoothing factor is given in term of ω as

$$s(\omega) = \max\{|1 - 2\omega|, |1 - \omega|\}.$$

We see that the optimal choice for the damping parameter within Jacobi is $\omega = \frac{2}{3}$ and that it leads to a smoothing factor $s(\frac{2}{3}) = \frac{1}{3}$. We would like to stress that this means that every component in the subspace spanned by the oscillatory eigenvectors is damped by at least by $\frac{1}{3}$ in each relaxation step. Finally, we would like to emphasize that this statement is completely independent of the matrix size – and hence it already indicates why multigrid gives an $O(n)$ solver on that problem.

Although damped Jacobi does an excellent job for the above simple model problem, it is well known that appropriate Gauss-Seidel type smoothers usually turn out to be highly superior to appropriate Jacobi type smoothers. (Anyway, Jacobi is regarded to be a much better smoother still than Richardson.) However, smoothing relaxations are not restricted to Gauss-Seidel, Jacobi and Richardson type iterations: For example, Wittum [115] picked the matrix M in (2.8) to be an incomplete LU-factorziation and showed its excellent smoothing properties for two-dimensional convection-diffusion problems both analytically and numerically. Very recently, Grote and Bröker proposed to use sparse approximate inverses for CFD problems in [14] and [15]

as they give good smoothing regardless of the flow pattern. They obtain the smoothers via the SPAI algorithm by Grote and Huckle [60], i.e. they solve the minimization problem $\|I-MA\|_F$ either for a prescribed sparsity pattern of M or for a fixed tolerance ε.
Within multigrid methods for image deblurring in Chapter 6 we will employ a semi-iterative smoother, i.e. there is no longer a stationary iteration matrix $G = I - MA$ as in (2.9), but the matrix varies from iteration to iteration.

In general situations, smoothing factors are of course not as easy to analyze as in our simple introductory example: One possibility is to perform so-called **local-mode analysis** on the given problem which is based on freezing coefficients locally. Local-mode analysis was first introduced in Brandt's seminal paper [10] in 1977. For details, we refer to [108], ch. 4, or to the Brandt's 1994 paper [11].
Hackbusch has formulated abstract concepts for the *smoothing property* which take into account the underlying differential operator (see [62], ch. 6). However, here we would only like to introduce an algebraic **smoothing condition** taken from the work by Ruge and Stüben [92], p. 82:

Definition 2 (Algebraic smoothing condition)
Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix and let $\|\cdot\|_A$ and $\|\cdot\|_{DA}$ be the norms associated with the inner products

$$\langle u, v\rangle_A = \langle Au, v\rangle, \qquad \langle u, v\rangle_{DA} = \langle diag(A)^{-1}Au, v\rangle,$$

A stationary iterative method of the form (2.8) with iteration matrix

$$G = I - M * A$$

is said to have the algebraic smoothing property, if there exists $\alpha > 0$ such that

$$\|Ge\|_A^2 \leq \|e\|_A^2 - \alpha\|e\|_{DA}^2 \qquad \textit{for all} \quad e \in \mathbb{R}^n \tag{2.17}$$

The motivation of Definition 2 is that in an algebraic understanding of multigrid an error component is called smooth if it is slow to converge, i.e. $\|Ge\|_A \approx \|e\|_A$. However, this generalization is not to be understood in a geometric sense, because in a general situation a problematic error component need not necessarily be associated with a smooth eigenvector.

In order to obtain an efficient multigrid algorithm one should try to keep the

number of smoothing steps low. As kind of a rule of thumb, the total number of pre- and postsmoothing steps should not be greater than 6 – and usually less smoothing will come out to be sufficient.
Finally, the reader might wonder about the purpose of postsmoothing and why the postsmoother might be different from the presmoother: The idea is that the presmoother works in a fixed subspace of highly oscillatory error components and that the coarse grid correction completes the reduction of the error on the complementary space of slowly varying error components. The postsmoother is intended to deal with "intermediate subspaces" on which neither the presmoother nor the coarse grid correction are highly contractive. Hence for postsmoothing frequently a larger damping parameter ω will be used than for presmoothing. More details and another notion of this subspace interpretation will be given in 2.4.4.

2.4.2 Grid transfer operators – Prolongation and restriction

Following [13], ch. 3, let us start with the 1D Laplacian (2.1) with Dirichlet boundary conditions in the form $A = (1/h^2) * tridiag(-1, 2, -1)$. For the rest of the subsection let us assume that $h = 1/n$ with $n = 2^l$, l integer, is the fine grid mesh size and $2h$ is the coarse mesh size. The simplest way to transfer a coarse grid vector v_{2h} of length $\frac{n}{2} - 1$ to the fine grid is to use **linear interpolation**, i.e.

$$\begin{aligned} v_h(2j) &= v_{2h}(j) \\ v_h(2j+1) &= \frac{1}{2}(v_{2h}(j) + v_{2h}(j+1)), \qquad 0 \le j \le \frac{n}{2} - 1. \end{aligned}$$

In matrix terms we can express the interpolation operator as a matrix $P_{1D} \in \mathbb{R}^{(n-1)\times(\frac{n}{2}-1)}$

$$\mathbf{P_{1D}} = \begin{pmatrix} 0.5 & & & & \\ 1 & & & & \\ 0.5 & 0.5 & & & \\ & 1 & & & \\ & \cdots & \cdots & & \\ & & 0.5 & 0.5 & \\ & & & 1 & \\ & & & 0.5 & \end{pmatrix} \tag{2.18}$$

The linear interpolation operator carries over to two-dimensional problems (2.4) in a straightforward way: Given the coarse grid vector v_{2h} of length $(\frac{n}{2}-1)^2$ we transfer it to the fine grid via **bilinear interpolation**, i.e. for $0 \leq i,j \leq \frac{n}{2}-1$ we compute

$$\begin{aligned} v_h(2i,2j) &= v_{2h}(i,j) \\ v_h(2i+1,2j) &= \frac{1}{2}(v_{2h}(i,j)+v_{2h}(i+1,j)) \\ v_h(2i,2j+1) &= \frac{1}{2}(v_{2h}(i,j)+v_{2h}(i,j+1)) \\ v_h(2i+1,2j+1) &= \frac{1}{4}(v_{2h}(i,j)+v_{2h}(i+1,j)+v_{2h}(i,j+1)+v_{2h}(i+1,j+1)) \end{aligned}$$

Note that the corresponding bilinear interpolation matrix P_{2D} can be written as Kronecker product

$$P_{2D} = P_{1D} \otimes P_{1D} \tag{2.19}$$

of the linear interpolation operator P_{1D}.

In order to restrict a fine grid vector v_h of length $n-1$ to the coarse grid in the one-dimensional case, we could use full weighting:

$$v_{2h}(j) = \frac{1}{4}(v_h(2j-1)+2v_h(2j)+v_h(2j+1)), \qquad 1 \leq j \leq \frac{n}{2}-1. \tag{2.20}$$

Now we observe that the one-dimensional full weighting operator $R_{1D} \in \mathbb{R}^{(\frac{n}{2}-1)\times(n-1)}$ can be expressed as

$$R_{1D} = c * P_{1D}^T \tag{2.21}$$

with $c = 0.5$. Equation (2.21) is often referred to as **Galerkin condition** or **variational condition** and it plays an important role in the mathematical understanding of multigrid algorithms.
Analogously to (2.19), the corresponding full weighting operator in two dimensions can be written as

$$R_{2D} = R_{1D} \otimes R_{1D} \tag{2.22}$$

and there holds

$$R_{2D} = c * P_{2D}^T \tag{2.23}$$

with $c = 0.25$. Instead of writing restriction operators as matrices it comes out to be much more convenient to describe the full-weighting operator (2.22) as a stencil (cf. [62], p. 65)

$$\mathbf{R_{2D}} = \begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix}.$$

In general the stencil

$$\mathbf{R} = \begin{bmatrix} \sigma_{-1,1} & \sigma_{0,1} & \sigma_{1,1} \\ \sigma_{-1,0} & \sigma_{0,0} & \sigma_{1,0} \\ \sigma_{-1,-1} & \sigma_{0,-1} & \sigma_{1,-1} \end{bmatrix} \tag{2.24}$$

denotes the weighted restriction

$$(Rv_h)(x,y) = \sum_{\alpha,\beta=-1}^{+1} \sigma_{\alpha,\beta} v_h(x+\alpha h, y+\beta h) \tag{2.25}$$

This stencil notation will also prove to be helpful when we study matrix-dependent prolongations and restrictions in Chapter 5.

We should not overlook that there is still a simpler way to define a restriction operators: By

$$v_{2h}(j) = v_h(j) \qquad \text{for} \qquad 1 \le j \le \frac{n}{2} - 1, \tag{2.26}$$

we get the so-called **trivial injection** (which simply has the stencil $R = [1]$). It is even cheaper than linear interpolation and may well be employed in certain situations (like e.g. in the context of multigrid for integral equations in Chapter 6), but except for lacking any variational property it also bears other disadvantages. We refer to [62], sec. 3.5, for a more detailed discussion.

Higher order interpolation and restriction methods, like e.g. cubic interpolation may also be used, but (as we shall also see in this thesis in Chapter 4) they rarely lead to a significant improvement of the algorithm. For a discussion, we again refer to [62], sec. 3.4 and 3.5. Anyway, one definitely should keep in mind that higher order interpolation will always be much more expensive computationally.

From now on we will refer to (bi)linear interpolation as standard prolongation and to the full weighting operator as standard restriction. These standard transfer operators are the most common choice within practical geometric multigrid algorithms; the monograph by Trottenberg, Oosterlee and Schüller [108] uses almost entirely standard prolongations and restrictions. In Chapters 4 and 5 we will explain from two different viewpoints why standard transfer operators are an optimal choice for discrete Poisson problems. However, in Chapter 5 we will also deal with cases when standard transfer operators no longer work properly and have to be replaced by matrix-dependent prolongations and restrictions.

With the material introduced about multigrid so far we can now understand why multigrid without smoothing can not work in general (see [62], p. 23):

Remark 2 *Using the coarse grid correction operator given by (2.13) to solve the linear system $A_h x = b$ without any smoothing does not lead to a convergent iteration.*

Proof. The restriction operator R has a nontrivial kernel. Use a starting vector $x^{(0)} = A_h^{-1}(b + w)$ with $0 \neq w \in ker(R)$. Then the residual will be $r = A_h * x^{(0)} - b = w$ and as $R * r = 0$ the iteration stagnates at $x^{(0)}$.

We would finally like to mention that in the seventies Schröder and Trottenberg proposed the so-called **total reduction method** [94], [95] to solve the two-dimensional Laplace matrix (2.4) in $O(n)$ time which works in a multigrid fashion but without smoothing. However, within the total reduction algorithm the coarse grid equations are not set up using a fixed restriction operator, but they are carefully constructed such that they are equivalent to the fine grid equations for the respective unknowns. Thus no smoothing is needed. However, the total reduction algorithm is tailored to the problem (2.4): It is basically only applicable to one very special block tridiagonal matrix – and thus has not succeeded. On the other hand, multigrid is not an algorithm for one special matrix problem, but a very general technique with a vast range of applicability – and, at the price of smoothing, the techniques to set up the coarse grid equations are pretty simple.

2.4.3 Coarse grid representations – rediscretization or Galerkin coarse grid operator

Looking at the matrices (2.1) and (2.4) from our model problems, there is a straightforward way to derive the coarse grid representation. We simply rediscretize the problem on a coarser mesh with mesh size $2h$. In other words: If our system matrix is

$$A_h = (1/h^2) * tridiag(-1, 2, -1),$$

then our coarse grid matrix will simply be

$$A_{2h} = (1/2h)^2 * tridiag(-1, 2, -1).$$

In the case of a rediscretization we will refer to A_{2h} as a **natural coarse grid operator**.

However, for the case $h = 1/n$ with $n = 2^l$, l integer, we can quickly check that with P_{1D} and R_{1D} from (2.18) and (2.20), respectively, there holds

$$A_{2h} = R_{1D} * A_h * P_{1D} = (1/2h)^2 * tridiag(-1, 2, -1),$$

i.e. we with the definition

$$A_{coarse} = R * A_{fine} * P \tag{2.27}$$

we obtain the identical coarse grid representation as with rediscretization for the model problem $A_h = (1/h^2) * tridiag(-1, 2, -1)$.
Equation (2.27) is referred to as the **Galerkin coarse grid operator** or **variational coarse grid operator**. It allows us to set up the coarse grid matrices in algebraic terms. Note that in general natural and Galerkin coarse grid operators do not coincide: Revisiting $A_h = (1/h^2) * tridiag(-1, 2, -1)$ in the case that the matrix size is even we see that the Galerkin coarse grid operator will usually differ from a rediscretization by a matrix of rank 2.

Certainly, the natural coarse grid operator looks much simpler and it is obvious that the preprocessing phase for setting up the coarse grid matrices will be much cheaper when we rediscretize instead of using (2.27). Still, the natural coarse grid operator is the variant of choice among multigrid practitioners: Almost the whole discussion in the recent book by Trottenberg, Oosterlee and Schüller [108] is based on algorithms employing natural coarse

grid operators. However, variational coarse grid operators (2.27) also have significant advantages: First of all, equation (2.27) also works out as well if the matrix A_{fine} stems from a discretization on an unstructured grid or if we do not have any notion of a physical grid at all. Secondly, the definition (2.27) also brings various favourable mathematical properties: In the case $R = c * P^T$ we introduce a variational principle which helps to improve multigrid algorithms significantly in certain cases and furthermore often facilitates the mathematical analysis a lot (or makes such an analysis possible at all).

It is a major point of this Ph.D. thesis to contribute to the discussion when natural coarse grid operators are helpful to simplify and accelerate algorithms and when it is inappropriate to use them. In order to perform this investigation, we first need to clarify that using natural coarse grid makes a significant difference within a multigrid algorithm (cf. also [108], sec. 2.7):

Remark 3 (Correct scaling of the restriction of defects)
When we would like to employ natural coarse grid operators in multigrid algorithms, we need to note that for second order finite difference discretizations the matrices A_{fine} and A_{coarse} bear the different factors $(1/h)^2$ and $(1/2h)^2$, respectively. In practical implementations one usually prefers to multiply the discrete equation by this factor h^2 – just as we did when we derived the discrete Laplacians in Examples 1 and 2. In this case great care must be taken to use the **correct factor $4h^2$ on the coarse grid**. *In other words: We must not forget to multiply the result $r_{coarse} = R * r_{fine}$ with a factor $fac = 4$, i.e. we could well say that within Algorithm 2 equation (2.15) is then actually to be read as*

$$A_{k-1} e_{k-1}^{(j)} = fac * r_{k-1}^{(j)} \tag{2.28}$$

Note that for second order discretizations $fac = 4$ independently of the dimension of the problem. For fourth order discretizations we would need to use $fac = 16$.

On the other hand, when using Galerkin coarse grid operators (2.27) there is never a need to worry about such scaling factors. Furthermore, the variational principle allows us simply to set $R = P^T$ and this will not affect the performance of the multigrid algorithms – whereas when using natural coarse grid operators we are forced to use standard prolongations and restrictions exactly in the forms (2.18),(2.20) or (2.19),(2.22), respectively. In other words: When working with rediscretizations we have to respect the idea of

the underlying physical grid within prolongations, restrictions and coarse grid matrices in our multigrid algorithms. On the other hand, Galerkin coarsening detracts us from the physical grid and allows us to understand multigrid much more as an algebraic technique based on variational principles.

2.4.4 Some additonal remarks

In Section 2.2 we have motivated multigrid in terms of a subspace splitting: We said that the smoother reduces the error in the subspace of highly oscillatory frequencies whereas the coarse grid correction handles the slowly varying error components.
Now that we have introduced Galerkin coarse grid matrices (2.27), we can use equations (2.12) and (2.13) to interpret a twogrid method in terms of an **algebraic subspace picture** (see e.g. [53], [54], ch. 12, or [46], ch. 7). However, we would only like to explain this idea very briefly: For details we refer to the original article by Greenbaum [53] from 1984 where the reader can find these issues treated with depth and rigour.
Following [53] the Galerkin coarse grid operator $A_{coarse} = P^T A P$ can be interpreted as the original matrix A_{fine} projected onto the subspace $\mathcal{S} = span\{p_1, \ldots, p_m\}$ spanned by the column vectors of the prolongation matrix $P = [p_1, \ldots, p_m]$. On the other hand, the matrix $A_{sub}^{-1} = P A_{coarse}^{-1} P^T$ can be interpreted as the inverse of the matrix A_{fine} on this subspace $\mathcal{S}$. Furthermore, Greenbaum [54] points out that $\mathcal{S}^\perp$ lies in the kernel of A_{sub}^{-1} and then concludes that the coarse grid correction matrix (2.13), i.e. $I - A_{sub}^{-1}$ is the identity on $\mathcal{S}^\perp$ and the null matrix on $\mathcal{S}$.
This allows us to see the picture of subspaces of different frequency components in a new light: The closer the subspace $\mathcal{S}$ to the subspace of the problematic slowly varying components and the more effective our smoother on $\mathcal{S}^\perp$ the faster our multigrid algorithm. In order words: We should try to pick the columns of our prolongation matrix $P = [p_1, \ldots, p_m]$ such that it mainly contains the slowly varying error components.

Thoughout our discussion we have so far all the time assumed that the fine and coarse grid mesh sizes differ by a factor of 2 and we will stick with this assumption during the whole work – except for a little section in Chapter 4 where we will point out that the standard choice, i.e. mesh sizes differing by a factor of 2, is algorithmically superior to restrictions picking every third or fourth column.

However, there are also other approaches, like e.g semi-coarsening where a two-dimensional mesh is coarsened in either x- or y-direction only. Finally, we would like to mention the celebrated **algebraic multigrid (AMG)** approach first proposed by Ruge and Stüben in [92]: As the name of the method says, AMG is based entirely on the system matrix A and needs no further information about underlying grids. Coarse grids are not fixed, but dynamically constructed on the basis of the entries of the system matrix.

Now that we know all about its components , we can estimate the **computational costs of multigrid**: Looking at Algorithm 2 we understand that the computationally most expensive parts are the matrix-vector multiplications on the fine grid needed in steps (1) and (3), i.e. in pre- and postsmoothing, and in step (2a) when we compute the fine grid residual.
If our system matrix A is sparse, then smoothers, prolongation and restriction are local operations computing weighted averages between grid points. As each component does a constant amount of work per grid point, the total amount of work on each grid is proportional to the number of unknowns on the respective grid. Let us assume that the coarsest grid is sufficiently coarse such that the solution of (2.15) is much cheaper than a relaxation step on the finest grid. As the different mesh sizes differ by a factor of two, the total amount of work for a V-cycle is approximately

$$\sum_{k=0}^{l} (\frac{1}{2^d})^k \tag{2.29}$$

times the computational work on the fine grid (with l denoting the number of grids and d the denoting the dimension of the problem). As the geometric series (2.29) is finite, we understand that the costs for a V-cycle are of complexity $O(n)$ (with n denoting the number of unknowns on the fine grid) – and, very similarly, we can draw the same conclusion for W-cycles. (See [108], sec. 2.4.3, for a more detailed discussion of work estimates for different types of multigrid cycles.)
Hence our goal will be to find a multigrid solution to a linear system $Ax = b$ with the same computational complexity as the matrix-vector multiplication Av. In this case we speak of a computationally optimal solver.

2.5 Multigrid as a preconditioner

2.5.1 Preconditioned Conjugate Gradients

Among the most powerful and versatile iterative linear system solvers are the family of algorithms known as **Krylov subspace methods**, of which the most well known is the Conjugate Gradient Method. As the n-th step of the iteration, $n = 1, 2, \ldots$, these methods choose an approximation to the solution of the linear system with the coefficient matrix A from the Krylov space

$$K_n(x^{(0)}, A) := span\{x^{(0)}, Ax^{(0)}, \ldots, A^{n-1}x^{(0)}\}$$

where $x^{(0)}$ is an initial guess. If used as stand-alone solvers for elliptic problems these methods are far from optimal. However, we may try to turn them into computationally optimal methods by preconditioning, i.e. instead of the original linear system $Ax = b$ we solve the equivalent system $PAx = Pb$ with a preconditioner P that is "easy to invert".

As this thesis deals entirely with the symmetric positive definite matrices, the only Krylov subspace method we will introduce is the Conjugate Gradient (CG) method. It was first proposed by Hestenes and Stiefel [69] in 1952 and is a technique for symmetric positive definite linear systems $Ax = b$. A good preconditioner for the CG method would be a symmetric positive matrix P satisfying the two following conditions:

- PA is well conditioned or has few extreme eigenvalues
- $Px = b$ is easy to solve numerically, i.e. we can obtain an approximate solution efficiently and with requirements for computing time and storage of the same complexity as a matrix-vector multiplication Av.

The following algorithm describes the preconditioned CG method. The unpreconditioned CG algorithm can simply be recovered from it by setting $P = I$. For more information about the CG algorithm, we refer to the detailed presentations in the books by Golub and Van Loan [51], pp. 520, and Axelsson [2], pp. 459. There the reader can learn more about CG as an optimization algorithm and get to understand how the preconditioned CG algorithm can be derived from the original algorithm from 1952.

Algorithm 3 (Preconditioned Conjugate Gradients)
We want to solve the symmetric positive definite linear system $Ax = b$. Let

$x^{(0)}$ be the initial guess, $r^{(0)} = b - Ax^{(0)}$ be the initial residual and the preconditioner M be a symmetric positive definite matrix. Furthermore set $\beta^{(0)} = 0$. In order to solve our linear system by the method of preconditioned conjugate gradients we proceed with the following iteration $k = 0, 1, \ldots$ until a stopping criterion is satisfied:
(1) Preconditioning step: Solve

$$Pz^{(k)} = r^{(k)}$$

(2) If $k > 0$ compute the "improvement this step":

$$\beta^{(k)} = \frac{\langle r^{(k)}, z^{(k)} \rangle}{\langle r^{(k-1)}, z^{(k-1)} \rangle}$$

(3) Update the "search direction": If $k = 0$ then $p^{(0)} = z^{(0)}$, otherwise compute

$$p^{(k)} = z^{(k)} + \beta^{(k)} p^{(k-1)}$$

(4) Perform the matrix-vector multipication:

$$q^{(k)} = Ap^{(k)} \tag{2.30}$$

(5) Update the "step length":

$$\alpha^{(k)} = \frac{\langle z^{(k)}, r^{(k)} \rangle}{\langle p^{(k)}, q^{(k)} \rangle}$$

(6) Update the approximate solution:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$$

(7) Update the residual:

$$r^{(k+1)} = r^{(k)} - \alpha^{(k)} q^{(k)}$$

It is important to note that Algorithm 3 requires only one multiplication with the matrix A in step (4).

We would very briefly like to remind the reader of a result describing how the speed of convergence of the conjugate gradient algorithm for the solution of $Ax = b$ relates to the 2-norm condition number of the the system matrix A (see e.g. Trefethen and Bau [107], p. 299):

Lemma 3 *Let the method of conjugate gradients be applied to a s.p.d matrix problem $Ax = b$.Let κ be the 2-norm condition number of A and the* energy norm $\|\cdot\|_A$ *be given by $\|x\|_A^2 = x^T Ax$.*
For the initial error $e^{(0)}$ and the error $e^{(n)}$ after n CG steps there holds:

$$\frac{\|e^{(n)}\|_A}{\|e^{(0)}\|_A} \leq 2(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^n. \tag{2.31}$$

It is worth noting that the bound (2.31) may sometimes be rather pessimistic and the CG algorithm may converge much faster, e.g. in the case that the eigenvalues of A are very well clustered (Cf. [107], p.299).

Lemma 3 implies that in order to reduce the energy norm of the error by a factor $\varrho \in]0,1[$ one needs at most

$$m \geq \frac{1}{2}\sqrt{\kappa}|\log(\frac{\varrho}{2})| \tag{2.32}$$

steps of the CG method.

2.5.2 Additive and multiplicative multigrid preconditioners

For linear systems arising from a discretization of elliptic problems – or other types of ill-conditioned matrices – we can precondition CG with a multigrid cycle, i.e. we simply perform one iteration step of a V-cycle or a W-cycle with zero initial guess in step (1) of Algorithm 3. (Note that it is important to start with the zero initial guess for otherwise we would have different preconditioners in subsequent steps of Algorithm 3). In the previous subsections we have introduced multigrid as an iterative solver and we have outlined why it gives a fast and efficient solver of optimal complexity. Hence the reader might wonder why we also consider using multigrid as a preconditioner. We continue with a quote on this issue taken from the book by Anne Greenbaum [54], p. 197:

Some multigrid aficionados will argue that if one has used the proper restriction, prolongation and relaxation operators, then the multigrid algorithm will require so few cycles (...) that it is almost pointless to try to accelerate it with CG-type methods. This may be true, but unfortunately such prolongation, restriction and relaxation operators are not always known.

We believe that this quote is certainly true for complicated problems in the field of Scientific Computing, including non-symmetric problems, adaptive discretizations on highly nonuniform (possibly unstructured) grids, and so on. However, this work deals with structured linear systems which are also symmetric positive definite. We will see that in the structured case it is usually possible to identify appropriate smoothers, transfer operators and coarse grid representations – and our numerical experiments will show that our multigrid cycle solvers usually perform very efficiently such that no acceleration with CG is needed.

In the case that a multigrid cycle is used for preconditioning we speak of **multiplicative multigrid preconditioning**. In contrast to the multiplicative multigrid algorithms discussed there are also additive preconditioners like the celebrated BPX-preconditioner [9] or the multilevel diagonal scaling method [120]. These methods are designed to work as preconditioners only and although they can rarely outperform their multiplicative counterparts on a serial computer, they are highly interesting for their usually superior parallel performance.
Additive preconditioners are distinct from multigrid solvers in the sense that they do not execute any successive coarse grid corrections. Instead, they simply set up coarse grid problems and the solutions on these coarse grid are finally added. It is rather obvious that such a procedure can not lead to a convergent solver, but it can be shown that additive preconditioners lead to optimal preconditioners, e.g. for certain types of elliptic problems.
Let

$$\bar{P}^{(j)} = \prod_{k=j+1}^{l} P_{(k-1)}^{(k)}$$

denote the product of prolongations on the different levels (with l denoting the finest level and 0 the coarsest). Then we can write Zhang's [120] multilevel diagonal scaling (MDS) preconditioner as (see also [102], p. 71)

$$B = \bar{P}^{(0)^T} A^{(0)^{-1}} \bar{P}^{(0)} + \sum_{j=1}^{l-1} \bar{P}^{(j)^T} D^{(j)^{-1}} \bar{P}^{(j)} + D^{(l)^{-1}} \tag{2.33}$$

In (2.33) $A^{(j)}$ denotes the matrices on the different levels of resolution and $D^{(j)}$ their diagonals. Note that in practice the exact solve $A^{(0)^{-1}}$ on the finest level in (2.33) may well be replaced with a simple diagonal solve, i.e. one

Jacobi step.
Similarly to (2.33), we can write the preconditioner by Bramble, Pasciak and Xu [9] in the two-dimensional case in the form (see [102], p. 74)

$$B = \bar{P}^{(0)^T} A^{(0)^{-1}} \bar{P}^{(0)} + \sum_{j=1}^{l-1} \bar{P}^{(j)^T} \bar{P}^{(j)} + I \tag{2.34}$$

The BPX preconditoner was originally developed for finite volume discretizations of the 2D Laplacian and there the diagonals $D^{(j)}$ scale like the identity. Historically, it was published before MDS, but it can be interpreted as a special case of MDS which uses less information about the underlying matrix. For more details on additive preconditioners and other variants of them we refer to chapter 3 of the book by Smith, Björstad and Gropp [102].

2.6 Some facts about multigrid convergence theory

We have said frequently that multigrid leads to $O(n)$ solvers for large classes of elliptic problems. This means that usually proofs concerning a multigrid algorithm involve two issues: First of all, one needs to establish convergence; secondly, one needs to show that multigrid gives a solver of optimal complexity, i.e. iteration counts are independent of the number of unknowns.
In general, such proofs are rather involved and technical: Even for the simple problem of a two-grid cycle for the one-dimensional Laplacian (2.1) with a natural coarse grid operator such a proof would cover several pages. Hence we decided not to give such a "model proof", but refer to the book by Hackbusch [62], p. 25-30. There Hackbusch establishes that the two-grid method for this one-dimensional problem converges at a rate independent of the number of unknowns involved.
The purpose of this work is to develop efficient multigrid algorithms. Hence our focus lies on the algorithms themselves and on numerical experiments underlining their efficiency – and it is certainly not our goal to contribute to multigrid theory. However, we would like to give a little overview about multigrid convergence theory in order to show that our algorithms are based on a solid theoretical base.
Studying multigrid convergence proofs one will make a number of observations:

- Usually the proofs first establish convergence for a twogrid cycle; then the analysis is carried over to W-cycles via perturbation arguments (see e.g. [62], ch. 7)

- Usually it is much simpler to establish convergence for W-cycles than for V-cycles: The first proofs of V-cycle convergence with a fixed number of smoothing steps for the Poisson problem was only given in 1981 by Braess [8].

- Normally multigrid convergence results are based on Galerkin coarsening $A_{coarse} = P^T A_{fine} P$; if this variational property does not hold proving theorems gets much more complicated. If natural coarse grid operators are employed, then usually convergence results can only be established if eigenvectors and eigenvalues of the system matrix are known exactly.

The foundations of a general multigrid theory were laid in the eighties by Hackbusch: In terms of the underlying differential operator he formulated equations characterizing the "approximation property" and the "smoothing property" (see [62], pp. 113). Based on these abstract properties Hackbusch was able to develop a general convergence theory for elliptic problems satisfying certain regularity conditions (i.e. excluding complications like e.g. L-shaped domains). Hence this theory is often referred to as *elliptic regularity theory*. It is interesting to note that elliptic regularity theory is the only way making it sometimes possible to prove convergence results for multigrid algorithms with natural coarse grid operators without a complete knowledge of eigenvalues and eigenvectors. However, elliptic regularity theory is basically restricted to symmetric positive definite problems (see [62], pp. 150).

In order to establish sharp results for additive multilevel preconditioners more abstract concepts were needed: The first proof of the optimality of BPX was presented in 1991 by Oswald [87] based on Besov scales; a little later a simpler proof by Bornemann and Yserentant [6] based on interpolation spaces followed. Oswald summarized and generalized his so-called *multigrid subspace correction theory* in his 1994 book [88]. In chapter 4 of his book [88] and in a joint article with Griebel [59] Oswald pointed out how additive and multiplicative multigrid methods can be seen in a unified framework. We will say more about these results when we make use of them in Chapter 5. Multigrid subspace correction is a very abstract and powerful framework.

However, to the multigrid practitioner it also has certain drawbacks: In particular, it gives entirely qualitative results. Although elliptic regularity theory is more restricted in scope, it sometimes also allows for estimates on the convergence speed which are no longer possible in the subspace correction framework. Furthermore, subspace correction theory requires an underlying differential or integral operator and it entirely allows the study of symmetric positive definite problems. Finally, Galerkin coarsening $A_{coarse} = P^T A_{fine} P$ is obligatory within the theory.
For more details on elliptic regularity and subspace correction theory we refer to the excellent overview articles by Yserentant [119] and Xu [116].
Although multigrid theory is in good shape for symmetric positive definite problems, hardly anything is known in the nonsymmetric or indefinite case – see the article by Cai and Widlund [16] for a discussion of the problems of an abstract framework. In particular, hardly any theoretical results are known about convergence of multigrid algorithms for convection-diffusion problems. Although such solvers are highly relevant in practice, proofs could so far mainly be accomplished in the constant-coefficient case on rectangular domains with periodic boundary conditions [90].

For the algebraic multigrid algorithm by Ruge and Stüben [92] proofs could only be established for the twogrid method, but not for V-cycles or W-cycles. Their convergence proof for a twogrid method relies on the following theorem taken from [92], p. 89, which we will also make use of in Chapter 4.

Theorem 1 (Criterion for twogrid convergence)
We would like to solve the linear system $Ax = b$ with a twogrid iteration employing only one presmoothing step and an exact solve on the coarse grid. Let n_h and n_H denote the number of unknowns on the fine and coarse grid, respectively, let $A \in \mathbb{R}^{n_h \times n_h}$ be a symmetric positive definite matrix, let the prolongation matrix $P \in \mathbb{R}^{n_h \times n_H}$ have full rank, let the restriction R be defined by $R = P^T$ and the coarse grid matrix by $A_{coarse} = P^T AP$.
Finally, let $\|\cdot\|_A$, $\|\cdot\|_D$ and $\|\cdot\|_{DA}$ be the norms associated with the three inner products

$$\begin{aligned} \langle u, v\rangle_A &= \langle Au, v\rangle, && (2.35)\\ \langle u, v\rangle_D &= \langle diag(A)u, v\rangle, && (2.36)\\ \langle u, v\rangle_{DA} &= \langle diag(A)^{-1}Au, v\rangle, && (2.37) \end{aligned}$$

let our smoother be an iterative method of the form (2.8) with iteration matrix $G = I - M*A$ *having the algebraic smoothing property (2.17), i.e. there exists* $\alpha > 0$ *such that*

$$\|Ge^h\|_A^2 \leq \|e^h\|_A^2 - \alpha\|e^h\|_{DA}^2 \qquad \textit{for all} \quad e^h \in \mathbb{R}^{n_h}$$

and let the twogrid operator M_{TG} *and the coarse grid correction matrix* K *be defined as in (2.12) and (2.13), respectively.*
If we can prove that there exists $\beta > 0$ *such that*

$$\min_{e^H \in \mathbb{R}^{n_H}} \|e^h - Pe^H\|_D^2 \leq \beta\|e^h\|_A^2 \qquad \textit{for all} \quad e^h \in \mathbb{R}^{n_h}, \tag{2.38}$$

then $\beta \geq \alpha$ *and for the convergence factor of the twolevel method in the energy norm* $\|\cdot\|_A$ *there holds*

$$\|M_{TG}\|_A = \|GK\|_A \leq \sqrt{1 - \frac{\alpha}{\beta}} \tag{2.39}$$

Certainly, the choice to go for only one presmoothing step in Theorem 1 means no restriction: The general convergence factor for a twogrid method (2.12) is simply $\|G_2^{\nu_2} K G_1^{\nu_1}\|_A$.

We can summarize that multigrid theory for symmetric positive definite problem is very powerful: In particular, this holds if a variational coarse grid operator is used and the problem results from the discretization of an elliptic operator. However, very often multigrid theories provide abstract asymptotic results – and these can only rarely tell the multigrid practitioner what is the best prolongation, restriction or smoothing operator for a specific problem at hand.

Chapter 3

Toeplitz matrices: Properties, fast algorithms and their implementation

This chapter presents some basic facts about Toeplitz systems, i.e. matrices with constant entries along diagonals. In the first section we point out that certain Toeplitz matrices can be connected to generating functions. This will be helpful in Chapter 4 when we try to translate a multigrid cycle into terms of functions. Generating functions also tell us a lot about the spectrum of Toeplitz matrices and thus equip the set of the associated Toeplitz matrices with a kind of "quasi-algebra" structure that can be used for preconditioning. In particular, we will also introduce band Toeplitz preconditioners. Section 2.1 summarizes results of the book by Grenander and Szegö [55], ch. 5, and the overview article by Chan and Ng [19] and is also strongly influenced by the thesis of Fiorentino [46], ch. 2.
We then present a fast algorithm for multiplying a Toeplitz matrix with a vector: Introducing the algebra of circulant matrices we explain how this operation can be performed efficiently at $O(n \log n)$ speed via the FFT and circulant embedding. Analogous ideas can be applied for Block Toeplitz matrices with Toeplitz Blocks (BTTB matrices), i.e. the bivariate counterparts of Toeplitz systems, employing the two-dimensional FFT. Furthermore, circulant preconditioners and Block circulant preconditioners with circulant blocks, respectively, come up along the way. Sections 2.2 and 2.3 are presented along the lines of the upcoming book by Vogel [112], ch. 5, and the book by Van Loan, [110], ch. 4.

3.1 Toeplitz matrices and generating functions

3.1.1 Introduction

Let $f(x)$ be a real-valued continuous function on the interval $[-\pi, \pi]$ and periodically extended to the whole real axis. Given the Fourier coefficients of $f(x)$

$$a_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(\theta) e^{ik\theta} d\theta, \qquad \text{for } k \text{ integer},$$

we can define the sequence of Toeplitz matrices $\{A_n \equiv T_n(f)\}_n$ associated with the generating function $f(x)$. Its entries are given by $(A_n)_{l,m} = a_{l-m}$:

$$A_n = \begin{pmatrix} a_0 & a_{-1} & \cdots & \cdots & a_{1-n} \\ a_1 & a_0 & a_{-1} & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & a_1 & a_0 & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_1 & a_0 \end{pmatrix}$$

Note that the matrices A_n are Hermitian, since $f(x)$ is real-valued. In case $f(x)$ is an even function, we are dealing with a sequence of real symmetric Toeplitz matrices and can represent $f(x)$ as a cosine series allowing us to find the entries a_k of A_n via

$$a_k = \frac{1}{\pi} \int_0^{\pi} f(x) \cos(kx) dx, \qquad k = 0, 1, \ldots \tag{3.1}$$

We will now see that the generating function tells us a lot about the eigenvalues of the associated Toeplitz matrices.

3.1.2 Distribution of eigenvalues

The following definition is taken from [55], p. 62:

Definition 3 (Equal distribution)
For all $n > 0$ let $\mathcal{A}_n = \{a_\nu^{(n)}\}_{\nu=1}^n$ and $\mathcal{B}_n = \{b_\nu^{(n)}\}_{\nu=1}^n$ be two sequences of real numbers such that

$$|a_\nu^{(n)}| < K, \quad |b_\nu^{(n)}| < K$$

with $K > 0$ independent of ν and n.
We define $\mathcal{A}_n$ and $\mathcal{B}_n$ to be **equally distributed** *on the interval $[-K, K]$ for $n \to \infty$ if the condition*

$$\lim_{n\to\infty} \frac{\sum_{\nu=1}^{n}[F(a_\nu^{(n)}) - F(b_\nu^{(n)})]}{n} = 0$$

is satisfied for any continuous function $F : [-K, K] \to \mathbb{R}$.

Definition 3 helps us to establish an important result concerning the spectrum of a Toeplitz matrix that is associated with a continuous generating function (see [55], pp. 64 and p. 72):

Theorem 2 (Eigenvalues of Toeplitz matrices):
Let the sequence of Toeplitz matrices $\{A_n \equiv T_n(f)\}_n$ be generated by the continuous real-valued function $f(x)$ and let

$$m_f = \min_{x\in[-\pi,\pi]} f(x), \qquad M_f = \max_{x\in[-\pi,\pi]} f(x)$$

(i) If we order the eigenvalues $\lambda_\nu^{(n)}$ of A_n in a nondecreasing way, i.e. $\lambda_1^{(n)} \leq \lambda_2^{(n)} \leq \ldots \leq \lambda_n^{(n)}$, then they satisfy the relation

$$\lim_{n\to\infty} \frac{1}{n} \sum_{\nu=1}^{n} F(\lambda_j^{(n)}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(f(x))dx$$

for every continuous function $F : [m_f, M_f] \to \mathbb{R}$.
In particular, the sets of values $\{\lambda_\nu^{(n)}\}_{\nu=1}^n$ and $\{f(\frac{\nu\pi}{n+1})\}_{\nu=1}^n$ are equally distributed as $n \to \infty$ in the sense of Definition 3.

(ii) Furthermore, for all n the spectrum of A_n is contained in the interval $[m_f, M_f]$ and there holds

$$m_f = \inf_{n\in\mathbb{N}} \lambda_1^{(n)}, \qquad M_f = \inf_{n\in\mathbb{N}} \lambda_n^{(n)} \tag{3.2}$$

Theorem 2 (i) tells us that the eigenvalues of A_n are not clustered and (ii) states that the extreme eigenvalues of A_n tend to the extremal values of $f(x)$.

Of particular interest will be the case of a generating function $f(x)$ with $m_f = 0$. From Theorem 2 we learn that the sequence of Hermitian Toeplitz matrices $\{A_n = T_n(f)\}_n$ will be asymptotically ill-conditioned.

In the case of a generating function with an isolated zero x_0 of finite order, it has furthermore been pointed out that the order of the zero x_0 governs how quickly the smallest eigenvalue $\lambda_1^{(n)}$ goes to zero (see e.g. [76] or [18] for details). Finally, we would like to assure the reader that in the case of a nonnegative generating function f with a finite number of zeros of finite order the Toeplitz matrices $A_n = T_n(f)$ will always be positive definite, i.e. the case $\lambda_1^{(n)} = 0$ never occurs (see e.g. [18]).

However, as soon as the generating function changes signs, matters become much less pleasant (see [114] or [46], p. 16):

Lemma 4 *Let the sequence of Toeplitz matrices $\{A_n \equiv T_n(f)\}_n$ be generated by a continuous function $f : [-\pi, \pi] \to \mathbb{R}$ with extremal values such that $\min_{x\in[-\pi,\pi]} f(x) = m_f < 0 < M_f = \max_{x\in[-\pi,\pi]} f(x)$. Then there holds:*
(i) The 2-norm condition number $\kappa(A_n)$ tends to infinity as n tends to infinity in a nonmonotone way.
(ii) The matrix A_n can be singular for some values of n.

From Lemma 4 (ii) we can conclude that if an indefinite Hermitian Toeplitz matrix belonging to a nondefinite generating function f arises in a real-world application we need to check very carefully whether the underlying mathematical model is well-posed in the sense of Hadamard.

Let us finally study a simple example.

Example 1 (revisited): The well-known matrix $tridiag(-0.5, 1, -0.5)$ – i.e. the 1D Laplacian – is related to the function $f(x) = -0.5e^{-ix} + 1 - 0.5e^{ix} = 1 - cos(x)$. The eigenvalues of $A_n = T_n(f)$ are contained in the interval $]0, 2[$. The small eigenvalues of A_n that lead to the large condition numbers are caused by the zero $x_0 = 0$ of f, $f(x_0) = f(0) = 0$, of multiplicity two.

3.1.3 Banded preconditioners

Obviously, the set of Toeplitz matrices does not form an algebra: Neither will the product of two Toeplitz matrices in general be Toeplitz nor will the inverse of a Toeplitz matrix share this structure – and it is almost superficial to say that we can not simply find the inverse of a Toeplitz matrix $A_n = T_n(f)$ generated by the function f via computing $T_n(\frac{1}{f})$, because $\frac{1}{f}$ might have singularities and its Fourier coefficients might not exist at all.
However, it has been observed that for nonnegative continuous functions f

and g the spectra of the matrices $M_1 = T_n(f) * T_n(g)$ and $M_2 = T_n(g)^{-1} T_n(f)$ can still be described in terms of the functions $f * g$ and f/g, respectively (see e.g. [38]). In other words: The generating functions equip the set of Toeplitz matrices generated by nonnegative continuous functions with a kind of "quasi-algebra" structure.

For our purpose the case $h = f/g$ is particularly interesting: We can think of f representing the original (possibly ill-conditioned) matrix and of g standing for the preconditioner. In [98] Serra was able to prove

Theorem 3 *Let f and g be two continuous nonnegative real-valued functions. Let r be the essential infimum of f/g and R its essential supremum. Then the eigenvalues of the matrix*

$$T_n(g)^{-1} T_n(f)$$

lie in the interval $]r, R[$ for all n and if we order them in a nondescreasing way there holds

$$\lim_{n\to\infty} \lambda_1^{(n)} = r, \qquad \lim_{n\to\infty} \lambda_n^{(n)} = R.$$

Theorem 3 is very interesting in the case of a dense Toeplitz matrix generated by a nonnegative function with a finite number of zeros of finite order. What could be simpler than using a sparse Toeplitz matrix generated by a function with identical zeros of the respective orders for preconditioning? If such a sparse matrix could be found, then we know that the spectrum of the preconditioned linear system would be contained in an interval $[a, b]$ with $0 < a \leq b < \infty$, and hence the preconditioned CG method would give a computationally optimal solver.
This idea underlies the so-called band Toeplitz preconditioners [18], [21]: First of all, remember from (3.1) that such a banded Toeplitz preconditioner needs to have a representation as a finite cosine series. Thus this idea is basically only applicable if there are exclusively zeros of even order. We can summarize the idea of banded preconditioners in the following theorem (see [18], sec. 4):

Theorem 4 (Band Toeplitz preconditioners)
Let f be a continuous real-valued function with the n isolated zeros

$$x_0, x_1, \ldots, x_n \in]-\pi, \pi] \quad \textit{of orders } 2\rho_0, 2\rho_1, \ldots, 2\rho_n.$$

Let $T_n(g)$ be the banded Toeplitz matrix generated by

$$g(x) = \prod_{j=0}^{n} [2 - 2\cos(x - x_j)]^{\rho_j}$$

Then the condition number of the preconditioned system

$$T_n(g)^{-1} T_n(f)$$

is bounded by a positive constant independent of the matrix size n.

3.2 Fast algorithms for Toeplitz systems

In this section we will point out in detail how a dense Toeplitz matrix can be multiplied with a vector at $O(n \log n)$ speed. This property is a major motive for numerical analysts to study Toeplitz systems – and it is also the reason why we worry about preserving Toeplitz structure in the coarse grid representations within our multigrid algorithms presented in Chapter 4.

3.2.1 Circulant matrices and the FFT

Definition 4 (Circulant matrices)
A matrix $C \in \mathbb{C}^{n \times n}$ is called circulant if it is Toeplitz and if its rows are circular right shifts of the elements of the preceding row, i.e. the entries of the matrix "wrap-around". We can identify the matrix

$$\mathbf{C} = \begin{pmatrix} c_0 & c_{n-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & \dots & c_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_{n-2} & \ddots & c_1 & c_0 & c_{n-1} \\ c_{n-1} & c_{n-2} & \dots & c_1 & c_0 \end{pmatrix}$$

with its first column $c = C(:,1) = (c_0, c_1, \dots, c_{n-1}) \in \mathbb{C}^n$ and write $C = circulant(c)$.

It can be shown that the set of circulant matrices forms the algebra of matrices that can be diagonalized by the Fast Fourier Transform ([112], p. 79):

Lemma 5 *Let $C \in \mathbb{C}^{n\times n}$ be a circulant matrix and $c \in \mathbb{C}^n$ be its first column. Then*

$$C = F^* diag(\hat{c}) F, \tag{3.3}$$

where $[F]_{jk} = w^{jk}/\sqrt{n}$ with $w = e^{i*2\pi/n}$ is the discrete Fourier matrix , F^* denotes its conjugate transpose and

$$\hat{c} = \sqrt{n} F c = \mathbf{fft}(\mathbf{c}) \tag{3.4}$$

In other words: The eigenvalues of the circulant matrix C are given by the components of $\hat{c}$ and the corresponding eigenvectors are the columns of F^*.

For a proof of Lemma 5 and more background material on the FFT we refer to the book by Van Loan [110].
Lemma 5 also tells us how to compute circulant matrix-vector products and inverses of nonsingular circulants efficiently (see [112], p. 79):

Algorithm 4 (Fast operations with circulants)
Let $C \in \mathbb{C}^{n\times n}$ be a circulant matrix, $c \in \mathbb{C}^n$ its first column and $b \in \mathbb{C}^n$ a column vector. The matrix-vector product $v = Cb$ can be computed at $O(n \log n)$ costs on the basis of the two equations (3.3), (3.4) via the following algorithm:
(1) Compute $\hat{c} = Fc = \mathbf{fft}(\mathbf{c})$ and $\hat{b} = Fb = \mathbf{fft}(\mathbf{b})$
(2) Compute the componentwise vector product $\hat{v} = \hat{c}. * \hat{b}$
(3) Compute $v = F^*\hat{v} = \mathbf{ifft}(\hat{\mathbf{v}})$
On the basis of the equation

$$C^{-1} = F^* diag(1./\hat{c}) F$$

nonsingular circulant systems $Cx = b$ can be solved via the same algorithm.

3.2.2 Efficient implementation of Toeplitz times vector multiplications

Definition 5 (Identifier of a Toeplitz matrix)
For the Toeplitz matrix $T \in \mathbb{C}^{n\times n}$ given by

$$\mathbf{T} = \begin{pmatrix} t_0 & t_{-1} & \dots & t_{2-n} & t_{1-n} \\ t_1 & t_0 & t_{-1} & \dots & t_{2-n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ t_{n-2} & \ddots & t_1 & t_0 & t_{-1} \\ t_{n-1} & t_{n-2} & \dots & t_1 & t_0 \end{pmatrix}$$

let us call $t = (t_{1-n}, \ldots, t_{-1}, t_0, t_1, \ldots, t_{n-1}) \in \mathbb{C}^{2n-1}$ *its identifier and write* $T = toeplitz(t)$.

Note that the identifier of the Toeplitz matrix $T \in \mathbb{C}^{n\times n}$ is a vector with $2n-1$ components whereas a circulant matrix $C \in \mathbb{C}^{n\times n}$ is identified with its first column (which has only n components).

How can we use Lemma 5 and Algorithm 4 to multiply a Toeplitz matrix $T \in \mathbb{C}^{n\times n}$ with a vector at $O(n \log n)$ costs? The answer is simply to embed T into a circulant matrix $C^{ext} \in \mathbb{C}^{2n\times 2n}$ with the first column

$$C^{ext}(:,1) = \begin{bmatrix} T(1:n,1) \\ 0 \\ T(1,n:-1:2).' \end{bmatrix}$$

written in MATLAB notation (and $a.'$ meaning transposition of a without conjugation).

Let us explain this idea a little more slowly and in more detail (see [112], p. 79): First we define the Toeplitz matrix $S \in \mathbb{C}^{n\times n}$ with the identifier

$$s = (t_1, t_2, \ldots, t_{n-1}, 0, t_{1-n}, \ldots, t_{-2}, t_{-1}) \in \mathbb{C}^{2n}$$

Now we have embedded the Toeplitz matrix $T \in \mathbb{C}^{n\times n}$ into a $2n$-by-$2n$ circulant matrix

$$C^{ext} = \begin{pmatrix} T & S \\ S & T \end{pmatrix} \tag{3.5}$$

and we call $C^{ext} \in \mathbb{C}^{(2n)\times(2n)}$ the *circulant extension* of the Toeplitz matrix T. Obviously, for $b \in \mathbb{C}^n$ there holds

$$C^{ext} \begin{pmatrix} b \\ 0_{n\times 1} \end{pmatrix} = \begin{pmatrix} Tb \\ Sb \end{pmatrix}$$

Reminding ourselves that $C^{ext} = circulant(c^{ext})$ with

$$c^{ext} = (t_0, t_1, \ldots, t_{n-1}, 0, t_{1-n}, \ldots, t_{-1}) \in \mathbb{C}^{2n} \tag{3.6}$$

everything is in place to formulate our Toeplitz times vector algorithm:

Algorithm 5 (Fast computation of Toeplitz times vector products) *Let* $T \in \mathbb{C}^{n\times n}$ *be a Toeplitz matrix and* $b \in \mathbb{C}^n$ *be a vector. We can compute*

the matrix-vector product $Tb = v$ with $O(n \log n)$ costs by the following algorithm:
(1) Set up the vector c^{ext} according to (3.6)
(2) Extend b, i.e. compute $b^{ext} = zeros(2 * n, 1)$ and set $b^{ext}(1:n) = b$
(3) Compute $\hat{c}^{ext} = \mathbf{fft}(\mathbf{c^{ext}})$ and $\hat{b}^{ext} = \mathbf{fft}(\mathbf{b^{ext}})$
(4) Compute the componentwise vector product $\hat{v}^{ext} = \hat{c}^{ext} .* \hat{b}^{ext}$
(5) Compute $v^{ext} = \mathbf{ifft}(\hat{\mathbf{v}}^{\mathbf{ext}})$
(6) Extract the first n components of v^{ext} to get v, i.e. $v = v^{ext}(1:n)$

3.2.3 Circulant preconditioners

Circulant matrices possibly give the most powerful class of preconditioners for Hermitian Toeplitz matrices generated by strictly positive functions.
Here we would mainly like to introduce the famous T. Chan preconditioner [28]. For a detailed survey on circulant preconditioning we refer to the article [19], ch. 2.

Lemma 6 (T. Chan circulant preconditioner)
Let $T \in \mathbb{C}^{n \times n}$ be a Toeplitz matrix with the identifier

$$t = (t_{1-n}, \ldots, t_{-1}, t_0, t_1, \ldots, t_{n-1}).$$

Then we call the circulant matrix $C(T) = circulant(c)$ with the first column c given by

$$c_j = \frac{(n-j)t_j + jt_{j-n}}{n}, \qquad j = 0, 1, \ldots, n-1 \tag{3.7}$$

the **T. Chan circulant preconditioner**.
Furthermore, the matrix $C(T)$ can be shown to give the best approximation from the algebra of circulant matrices to T in the Frobenius norm, i.e.

$$C(T) = arg \min_{C \text{ is circulant}} \|C - T\|_F$$

Obviously, the setup (3.7) for the preconditioner costs only $O(n)$ operations. When we use the T. Chan preconditioner inside CG, we simply compute $C(T)^{-1}z$ in $O(n \log n)$ operations via Algorithm 4.
The T. Chan circulant preconditoner is a very general approach: It can also be defined for an arbitrary matrix to yield the best circulant approximation in the Frobenius norm (see [28], [20] or [19] for details).

Of course, there are also other ideas for efficient circulant preconditioners: For example, Strang [103] proposed a preconditoner that simply copies the central diagonals $t_0, t_1, \ldots, t_{\lfloor n/2 \rfloor}$ of the Toeplitz matrix $T \in \mathbb{C}^{n \times n}$ and reflects them around to obtain a circulant matrix. We refer to [17] for more details and an analysis of Strang's preconditioner.
Other ideas use the embedding (3.5) to derive a preconditioner: In the simplest case, one could use $\hat{v}^{ext} = (1/\hat{c}^{ext}) .* \hat{b}^{ext}$ in step (4) of Algorithm 5 for preconditioning. For more ideas based on embedding see the article by Ku and Kuo [79].

Anyway, within our numerical experiments and comparisons in Chapters 4 and 6 our focus will lie on the preconditioner introduced in Lemma 6.

3.3 Fast algorithms for BTTB matrices

Toeplitz systems can usually only represent univariate models. We now come to Block Toeplitz matrices with Toeplitz blocks, i.e. the bivariate counterparts of Toeplitz systems, which can also be employed within multidimensional models.

3.3.1 Introduction

Definition 6 (BTTB matrices)
Let $T \in \mathbb{C}^{MN \times MN}$ be the block Toeplitz matrix

$$\mathbf{T} = \begin{pmatrix} T_0 & T_{-1} & \ldots & T_{2-M} & T_{1-M} \\ T_1 & T_0 & T_{-1} & \ldots & T_{2-M} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ T_{M-2} & \ddots & T_1 & T_0 & T_{-1} \\ T_{M-1} & T_{M-2} & \ldots & T_1 & T_0 \end{pmatrix}$$

with the N-by-N Toeplitz blocks

$$\mathbf{T_j} = \begin{pmatrix} t_{j,0} & t_{j,-1} & \ldots & t_{j,2-N} & t_{j,1-N} \\ t_{j,1} & t_{j,0} & t_{j,-1} & \ldots & t_{j,2-N} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ t_{j,N-2} & \ddots & t_{j,1} & t_{j,0} & t_{j,-1} \\ t_{j,N-1} & t_{j,N-2} & \ldots & t_{j,1} & t_{j,0} \end{pmatrix}$$

for $j = 1-M, 2-M, \ldots, 0, 1, \ldots, M-1$. Then the matrix $T \in \mathbb{C}^{MN \times MN}$ is called a Block Toeplitz Toeplitz Block matrix – and we shall from now on abbreviate: T is a **BTTB matrix**.
Furthermore, we will write $T = BTTB(t)$ with

$$t = (t_{1-M}, t_{2-M}, \ldots, t_0, \cdots, t_{M-1}) \in \mathbb{C}^{(2N-1)\times(2M-1)} \tag{3.8}$$

such that

$$T_j = toeplitz(t_j),$$

i.e. $t \in \mathbb{C}^{(2N-1)\times(2M-1)}$ is the matrix whose columns $t_j \in \mathbb{C}^{2N-1}$ are the identifiers of the Toeplitz block T_j. Furthermore, t is called the identifier of the BTTB matrix T.
In case the Hermitian BTTB matrix T is generated by a continuous function $f : [\pi, \pi]^2 \to \mathbb{R}$, then we find its entries via the two-dimensional semi-discrete Fourier transform, i.e.

$$t_{j_1,j_2} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} f(x,y) e^{-i(j_1 x + j_2 y)} dx dy$$

As for the relations between the spectra of BTTB matrices and their generating functions virtually everything presented in Section 3.1 carries over (see e.g. the work by Jin [74], [75] and Serra [97], [98]): In particular, the spectrum still lies in the range of f and in the case of a nonnegative generating function with an isolated zero (x_0, y_0), the order of the zero still governs the ill-conditioning of the corresponding sequence of BTTB matrices. Finally, the statements from Lemma 4 are also valid in the bivariate case, i.e. matrices generated by nondefinite functions might be singular for certain matrix sizes.
Let us look at a well-known example again.

Example 2 (revisited): The 2D-Laplacian given by (2.4) is related to the function

$$f(x,y) = 2 - 0.5 * (e^{-iy} + e^{-ix} + e^{ix} + e^{iy}) = 2 - \cos(x) - \cos(y)$$

The eigenvalues of the corresponding sequence of block tridiagonal matrices are contained in the interval $]0, 4[$. The small eigenvalues that lead to the large condition numbers are caused by the zero $(x_0, y_0) = (0, 0)$ of of f at the origin of multiplicity two.

3.3.2 BCCB matrices and the twodimensional FFT

We shall now introduce the two-dimensional analogue of circulant matrices (see also [112], pp. 81):

Definition 7 (BCCB matrices)
A matrix $C \in \mathbb{C}^{MN \times MN}$ is called block circulant with circulant blocks, abbreviated BCCB, if it satisfies the following conditions:
(a) C is BTTB.
(b) Each block is an N-by-N circulant matrix.
(c) The $N \times N$ block rows of C are circular right shifts of each other, i.e. the block structure also "wraps around".
We can identify the BCCB matrix

$$\mathbf{C} = \begin{pmatrix} C_0 & C_{M-1} & \dots & C_2 & C_1 \\ C_1 & C_0 & C_{M-1} & \dots & C_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ C_{M-2} & \ddots & C_1 & C_0 & C_{M-1} \\ C_{M-1} & C_{M-2} & \dots & C_1 & C_0 \end{pmatrix}$$

with the first columns $c_j \in \mathbb{C}^N$ of the circulant blocks $C_j = circulant(c_j)$, i.e. with $c = (c_0, c_1, \dots, c_{M-1}) \in \mathbb{C}^{N \times M}$. We write $C = BCCB(c)$ and call c the identifier of C. (Note that like in the 1D case the identifiers of BCCB matrices are much smaller than the identifiers of BTTB matrices.)

Like in one dimension, it can be shown that BCCB systems of given block sizes form an algebra of matrices that can be diagonalized by the appropriate two-dimensional Fast Fourier Transform (see also [112], p. 82):

Lemma 7 *Let $C \in \mathbb{C}^{MN \times MN}$ be a circulant matrix and $c \in \mathbb{C}^{N \times M}$ be its first column. Then*

$$C = (F_M \otimes F_N)^* diag(\hat{c})(F_M \otimes F_N), \tag{3.9}$$

where $[F_M]_{jk} = w_M^{jk}/\sqrt{M}$ with $w_M = e^{i*2\pi/M}$ and $[F_N]_{jk} = w_N^{jk}/\sqrt{N}$ with $w_N = e^{i*2\pi/N}$ are the discrete Fourier matrices of sizes $M \times M$ and $N \times N$, respectively, and

$$\hat{c} = \sqrt{MN}(F_M \otimes F_N)\bar{c} \tag{3.10}$$

with $\bar{c} = reshape(c, MN, 1)$. In other words: The eigenvalues of the BCCB matrix C are given by the components of $\hat{c}$ and the corresponding eigenvectors are the columns of $(F_M \otimes F_N)^*$.

For a proof of Lemma 7 and more background material on the two-dimensional FFT we again refer to the book by Van Loan [110].
The algorithm for computing circulant matrix-vector products is thus completely analogous to 1D. We shall to try formulate it in "MATLAB-like" notation: Note that **fft2** operates on N-by-M matrices and gives N-by-M matrices as the output in MATLAB.

Algorithm 6 (Fast operations with BCCB matrices)
Let $C \in \mathbb{C}^{MN\times MN}$ be a BCCB matrix, $c \in \mathbb{C}^{N\times M}$ its identifier and $b \in \mathbb{C}^{MN}$ a column vector. The matrix-vector product $v = Cb$ can be computed at $O(n \log n)$ costs on the basis of the two equations (3.9), (3.10) via the following algorithm:
(1) Reshape b, i.e. $b = reshape(b, N, M)$
*(2) Compute $\hat{c} =$ **fft2**$(\mathbf{c})$ and $\hat{b} =$ **fft2**$(\mathbf{b})$*
(3) Compute the componentwise matrix product $\hat{v} = \hat{c}. * \hat{b}$
*(4) Compute $v =$ **ifft2**$(\hat{\mathbf{v}})$*
(5) Reshape v, i.e. $v = reshape(v, MN, 1)$
If we change step (3) to

$$\hat{x} = (1./\hat{c}). * \hat{b}$$

we can solve nonsingular BCCB systems $Cx = b$ via the same algorithm.

3.3.3 Efficient implementation of BTTB times vector multiplications

We can embed a BCCB matrix $T = BTTB(t) \in \mathbb{C}^{MN\times MN}$ with identifier

$$\mathbf{t} = \begin{bmatrix} t_{1-N,1-M} & \dots & t_{1-N,0} & \dots & t_{1-N,M-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ t_{0,1-M} & \dots & t_{0,0} & \dots & t_{0,M-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ t_{N-1,1-M} & \dots & t_{N-1,0} & \dots & t_{N-1,M-1} \end{bmatrix}$$

into a BCCB matrix $C^{ext} = BCCB(c^{ext}) \in \mathbb{C}^{4MN \times 4MN}$ with identifier $c^{ext} \in \mathbb{C}^{2N \times 2M}$ given by

$$\mathbf{c^{ext}} = \begin{bmatrix} t_{0,0} & \dots & t_{0,M-1} & 0 & t_{0,1-M} & \dots & t_{0,-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ t_{N-1,0} & \dots & t_{N-1,M-1} & 0 & t_{N-1,1-M} & \dots & t_{N-1,-1} \\ 0 & \vdots & 0 & 0 & \vdots & \vdots & 0 \\ t_{1-N,0} & \dots & t_{1-N,M-1} & 0 & t_{1-N,1-M} & \dots & t_{1-N,-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ t_{-1,0} & \dots & t_{-1,M-1} & 0 & t_{-1,1-M} & \dots & t_{-1,-1} \end{bmatrix} \tag{3.11}$$

Note that the columns of c^{ext} simply represent the 1D circulant extensions of the $N \times N$ Toeplitz blocks and the respective ordering of the rows of c^{ext} guarantees that the block rows of C^{ext} are circular right shifts of each other. We call C^{ext} the *BCCB extension* of the BTTB matrix T.

To multiply $T \in \mathbb{C}^{MN \times MN}$ with a vector $b \in \mathbb{C}^{MN}$, we first need reshape it to $b = reshape(b, N, M) \in \mathbb{C}^{N \times M}$. Then we have to extend it with zeros via

$$b^{ext} = \begin{pmatrix} b & 0_{N \times M} \\ 0_{N \times M} & 0_{N \times M} \end{pmatrix} \tag{3.12}$$

Now b^{ext} can be treated by the following BTTB times vector algorithm. Again we use a "MATLAB-like" notation:

Algorithm 7 (Fast computation of BTTB times vector products) *Let $T \in \mathbb{C}^{MN \times MN}$ be a BTTB matrix and $b \in \mathbb{C}^{MN}$ be a vector. We can compute the matrix-vector product $Tb = v$ with $O(n \log n)$ costs by the following algorithm:*
(1) Set up the $c^{ext} \in \mathbb{C}^{2N \times 2M}$ according to (3.11)
(2) Extend b via (3.12), i.e. compute $b^{ext} = zeros(2 * N, 2 * M)$ and set

$$b^{ext}(1:N, 1:M) = reshape(b, N, M)$$

(3) Compute $\hat{c}^{ext} = \mathbf{fft2}(\mathbf{c^{ext}})$ and $\hat{b}^{ext} = \mathbf{fft2}(\mathbf{b^{ext}})$
(4) Compute the componentwise matrix product $\hat{v}^{ext} = \hat{c}^{ext} .* \hat{b}^{ext}$
(5) Compute $v^{ext} = \mathbf{ifft2}(\hat{\mathbf{v}}^{\mathbf{ext}})$
(6) Extract the first $N \times M$ subblock

$$v = v^{ext}(1:N, 1:M)$$

and, finally, set $v = reshape(v, 1, MN)$.

3.3.4 BCCB preconditioners

BCCB preconditioners are a very common choice for preconditioning BTTB systems. Like in the 1D case, they are particularly successful if the BTTB matrix belongs to a strictly positive function.

In 1994, T. Chan and J. Olkin [31] published the BCCB analogue of the T. Chan circulant preconditioner [28]: It can again be defined for arbitrary matrices and shown to be the best approximation from the algebra of BCCB matrices with a given block size in the Frobenius norm.
In the following we will only describe algorithmically how the **BCCB preconditioner by T. Chan and J. Olkin** can be set up efficiently for BTTB matrices:

Remark 4 (BCCB preconditioner by T. Chan and J. Olkin)
Let $T = BTTB(t) \in \mathbb{C}^{MN \times MN}$ be a BTTB matrix with identifier $t \in \mathbb{C}^{(2N-1)\times(2M-1)}$. In a first step we replace all the Toeplitz blocks $T_j \in \mathbb{C}^{N\times N}$; We can perform this operation efficiently by applying (3.7) to all columns of t, i.e. for $l = 1-M, \ldots, 0, 1, \ldots, M-1$ we compute the column vectors $\tilde{c}_l \in \mathbb{C}^N$ with components

$$\tilde{c}_l(j) = \frac{(N-j)t_{j,l} + jt_{j-N,l}}{N}, \qquad j = 0, 1, \ldots, N-1$$

Now we have obtained a "BTCB" matrix which can be represented by the array $\tilde{c} = (\tilde{c}_{1-N}, \ldots, \tilde{c}_0, \tilde{c}_1, \ldots, \tilde{c}_{N-1}) \in \mathbb{C}^{N\times(2M-1)}$.
However, we still need to make the block rows of our preconditioners circular right shifts of each other, i.e. we must apply a second level of circulant approximation and use (3.7) on the rows of $\tilde{c}$. Then for $k = 0, 1, \ldots N-1$ we compute

$$c_{k,l} = \frac{(M-l)\tilde{c}_{k,l} + l\tilde{c}_{k,l-M}}{M}, \qquad l = 0, 1, \ldots, M-1$$

With $c = (c_{k,j}) \in \mathbb{C}^{N\times M}$ we have found the identifier our BCCB preconditioner $C = BCCB(c)$.

Although there are also other BCCB preconditioners for Toeplitz systems, we will again put our major focus on the approach by T. Chan and J. Olkin [31] described above.
However, when we study BTTB systems arising from image deblurring in

Chapter 6, we will also work with the following Block circulant extension preconditioner which was proposed by Vogel in [112], p. 88, and goes back to an idea by Hanke and Nagy [66]. The preconditioner is simply the inverse of the BCCB extension matrix C^{ext} given by its identifier (3.11). We also give the algorithm for the Block circulant extension preconditioner, but remark that – except for a pretty evident change in step (4) – it is actually identical to Algorithm 7:

Algorithm 8 (Block circulant extension preconditioner)
Let $T \in \mathbb{C}^{MN \times MN}$ be a BTTB matrix, $r \in \mathbb{C}^{MN}$ a vector and $P \in \mathbb{C}^{MN \times MN}$ be the block circulant extension preconditioner. We can compute $P^{-1}r = z$ with $O(n \log n)$ costs by the following algorithm:
(1) Set up the $c^{ext} \in \mathbb{C}^{2N \times 2M}$ according to (3.11)
(2) Extend r via (3.12), i.e. compute $r^{ext} = zeros(2 * N, 2 * M)$ and set

$$r^{ext}(1:N, 1:M) = reshape(r, N, M)$$

(3) Compute $\hat{c}^{ext} = \mathbf{fft2}(\mathbf{c^{ext}})$ and $\hat{r}^{ext} = \mathbf{fft2}(\mathbf{r^{ext}})$
(4) Compute the componentwise matrix product $\hat{z}^{ext} = (1./\hat{c}^{ext}). * \hat{r}^{ext}$
(5) Compute $z^{ext} = \mathbf{ifft2}(\hat{\mathbf{z}}^{\mathbf{ext}})$
(6) Extract the first $N \times M$ subblock

$$z = z^{ext}(1:N, 1:M)$$

and, finally, set $z = reshape(z, 1, MN)$.

Chapter 4

Multigrid algorithms with natural coarse grid operators for Toeplitz systems

In this chapter we discuss multigrid algorithms for Toeplitz matrices generated by nonnegative functions with a finite number of zeros of finite order. We start with a motivating heuristics pointing out how prolongation and restriction operators can be interpreted as projected Toeplitz matrices and present some introductory reasoning on how the positions of the zeros affect a multigrid algorithm. Then we briefly summarize existing results on multigrid algorithms for Toeplitz matrices by Serra [47], [48] and R.Chan and collaborators [104], [25]. All previous approaches have been using a Galerkin coarse grid operator and, in general, this will result in a loss of Toeplitz structure on the coarse levels.
To overcome this problem we propose to employ natural coarse grid operators, i.e. our coarse grid representation is simply a Toeplitz matrix of half size belonging to the original function. We first point out how this idea can be applied in the case of a single zero $x_0 \in]-\pi, \pi]$; then we carry over our new approach to a finite number of equidistant zeros in $]-\pi, \pi]$. Our numerical tests confirm that we have developed optimal order algorithms.
Natural coarse grid operators also work very well for BTTB systems with a single zero $x_0 \in]-\pi, \pi]^2$; in that case the advantages of using a natural coarse grid operator are even more striking. The chapter ends with a new phenomenological characterization of well-known difficulties encountered in multigrid for indefinite BTTB matrices.

Numerical experiments showing the efficiency of our new algorithms play a major role within our presentation: In the whole chapter we always employ the following stopping criterion to obtain the iteration counts we list in our tables:

$$\frac{\|r^{(j)}\|_\infty}{\|r^{(0)}\|_\infty} \leq 10^{-6}$$

Here $r^{(j)}$ denotes the residual after j iterations and $r^{(0)}$ the original residual, i.e. we stop iterating when the relative residual corresponding to the maximum norm is less or equal 10^{-6}.

Parts of this chapter have already been published in the article [73]. However, this is also the place to acknowledge that some of the results from [73] are taken from an unpublished preprint (having the same title as as [73]) from March 1999 authored by T. Huckle alone. In particular, most of the ideas presented in the introductory section 4.1 can already be found in that preprint and subsection 4.5.2 on indefinite BTTB matrices has been taken over from there virtually unchanged. In other words: Original contributions of this chapter are the multigrid algorithms with natural coarse grid operators for Toeplitz and BTTB matrices generated by nonnegative functions, i.e. sections 4.3 and 4.4 and subsection 4.5.1.

4.1 Introduction

4.1.1 Our basic heuristics

Let $A_n = T_n(f) \in \mathbb{C}^{n\times n}$ be a Toeplitz matrix generated by the function f. How can we "translate" a multigrid cycle into terms of functions?
Let us take a look at the restriction R and the prolongation P: If we use a Galerkin approach (2.27) with $R = P^T$ to set up the coarse grid operator then we can write the coarse grid matrix for a twogrid step as

$$A_{n/2} \;=\; I_{n,n/2}^T * B_n^T * A_n * B_n * I_{n,n/2} \;=\; P_n^T * A_n * P_n$$

with a Toeplitz matrix B_n related to a function $b(x)$, and the elementary projection matrix

$$I_{n,n/2} \;=\; \begin{pmatrix} 1 & & \\ 0 & 0 & \\ 0 & 1 & 0 \\ & 0 & 0 \\ & 0 & 1 \\ & & \cdot \end{pmatrix} \;=\; I(1:n, 1:2:n)$$

in MATLAB-notation with the identity matrix I. Here we will mainly consider only symmetric B_n with real-valued generating function $b(x)$.

As a starting point for this chapter let us introduce the following heuristics: With

$$\tilde{f}(x) = b(x) * f(x) * b(x)$$

the entries of the matrix

$$B_n^T * A_n * B_n$$

are "asymptotically given" by the coefficients of $\tilde{f}(x)$; therefore the coefficients of $A_{n/2}$ can – up to a perturbation of low rank – be found by deleting every second entry in $\tilde{f}(x)$:

$$f_2(x) = (1/2) * \left(b^2(\frac{x}{2}) f(\frac{x}{2}) + b^2(\frac{x}{2} + \pi) f(\frac{x}{2} + \pi) \right) \tag{4.1}$$

Let us assume that $f(x)$ has a unique zero x_0 of finite order $2k$ in the interval $]-\pi, \pi]$. Now the new matrix $A_{n/2}$ should be closely related to the original A_n. Hence the related function $f_2(x)$ should have a zero with the same multiplicity as $f(x)$.

In view of $f(x) \geq 0$ this is only possible if $b(x_0 + \pi) = 0$. Therefore, we can easily motivate to use a prolongation operator of the form

$$b(x) \;=\; (\cos(x_0) + \cos(x))^k \;. \tag{4.2}$$

Remark 5 *Note that in general a suitable prolongation operator $b(x)$ may have an additional zero $b(x_1) = 0$ without generating an additional zero in $f_2(x)$ as long as $b(x_1 + \pi) f(x_1 + \pi) \neq 0$. More generally we could even use prolongation operators of the form $b(x) * h(x)$ with any nonnegative function h; but in most cases we are strongly interested in retaining sparsity by setting $h(x) \equiv 1$.*

4.1.2 Very important sparse cases

Let us study some immediate consequences of equation (4.2):

Remark 6 (Prolongations for the case $f(x) = 1 \mp \cos(m * x)$)
Let $A_n = T_n(f)$ be a Toeplitz generated by $f(x) = 1 - \cos(m * x)$ with m integer. Then (4.2) tells us that $b(x) = 1 + \cos(m * x)$ is an appropriate prolongation operator. Vice versa, for the nonnegative matrices generated by $f(x) = 1 + \cos(m * x)$, a suitable prolongation in the sense of (4.2) is given by $b(x) = 1 - \cos(m * x)$. In both cases we obtain $f_2(x) = 1 - \cos(m * x)$.

Proof: Choose $f(x) = 1 - \cos(m * x)$, apply (4.2) and then there holds

$$f_2(x) = (1 + \cos(\frac{m * x}{2}))^2 * (1 - \cos(\frac{m * x}{2})) +$$

$$+(1 + \cos(\frac{m * x}{2} + \pi)) * (1 - \cos(\frac{m * x}{2} + \pi))^2 \ =$$

$$= (1+\cos(\frac{m * x}{2}))^2 * (1-\cos(\frac{m * x}{2})) + (1-\cos(\frac{m * x}{2}))^2 * (1+\cos(\frac{m * x}{2})) \ =$$

$$= 2 * (1 - (\cos(\frac{m * x}{2}))^2) = 1 - \cos(m * x) \ .$$

In the case $f(x) = 1 + \cos(m * x)$ a completely analagous calculation shows that again $f_2(x) = 1 - \cos(m * x)$.

Finally, let us emphasize on the consequences of Remark 6 in the case $m = 1$: For $f(x) = 1 - \cos(x)$ this is nothing but the Toeplitz interpretation of standard prolongation and restriction. In the case $f(x) = 1 + \cos(x)$ we are dealing with an ill-conditioned matrix without negative entries: Usually multigrid approaches tend to run into trouble in such cases – and even the celebrated and very general AMG algorithm by Ruge and Stüben [92] will fail, because it can not find any "negative connections". However, the Toeplitz interpretation tells us how we can trace this problem back the 1D Laplacian on the level with $\frac{n}{2}$ unknowns by applying suitable transfer operators on the finest level.

4.1.3 The position of the zero

In this and the following two subsections we will assume that our Toeplitz matrix A_n is connected with a generating function f in the Wiener class having a single zero $x_0 \in [0, \pi]$ of finite order. Although we are actually interested

in dense Toeplitz matrices the following ideas are most easily explained by considering sparse linear systems.

Example 3 *Our standard example in the following subsections will be the sparse matrix belonging to the generating function*

$$f(x) = (\cos(x_0) - \cos(x))^2 \tag{4.3}$$

with $x_0 \in [-\pi, \pi] \backslash \{\pm\frac{\pi}{2}\}$. *Thus* f *has the two zeros* $\pm x_0$.

The matrices from Example 3 are strongly related to the indefinite matrices corresponding to $\tilde{f}(x) = \cos(x_0) - \cos(x)$ which can be seen as the result of a uniform finite difference discretization of the 1D Helmholtz equation

$$u_{xx} \ + \ \kappa^2 u \ = \ g$$

Note that the matrices from Example 3 will in general differ from the Helmholtz normal equations by a perturbation of low rank.

Let us consider Toeplitz matrices A_n generated by (4.3): According to (4.2) we use a function with zeros at $\pm x_0 + \pi$ as prolongation operator, namely $(\cos(x_0) + \cos(x))^k$. The corresponding prolongation matrices B_n are:

$tridiag(0.5, \cos(x_0), 0.5)$,
$pentadiag(0.25, \cos(x_0), \cos(x_0)^2 + 1/2, \cos(x_0), 0.25)$,
$septadiag(\frac{1}{8}, \frac{3}{4}\cos(x_0), \frac{3}{2}\cos(x_0)^2 + \frac{3}{8}, \cos(x_0)^3 + \frac{3}{2}\cos(x_0), \frac{3}{2}\cos(x_0)^2 + \frac{3}{8}, \frac{3}{4}\cos(x_0), \frac{1}{8})$,

and so on. The Galerkin coarse grid matrix matrix $A_{n/2}$ of half size is – up to a low rank term – related to the function

$$\begin{aligned} f_2(x) \ = \ (1/2)\Big(& (\cos(x_0) + \cos(\frac{x}{2}))^{2k} (\cos(x_0) - \cos(\frac{x}{2}))^2 + \\ & + (\cos(x_0) - \cos(\frac{x}{2}))^{2k} (\cos(x_0) + \cos(\frac{x}{2}))^2 \Big) = \end{aligned}$$

$$(\cos(x_0)^2 - \cos(\frac{x}{2})^2)^2 * \Big((\cos(x_0) + \cos(\frac{x}{2}))^{2k-2} + (\cos(x_0) - \cos(\frac{x}{2}))^{2k-2} \Big)/2 =$$

$$(\cos(2x_0) - \cos(x))^2 * \Big((\cos(x_0) + \cos(\frac{x}{2}))^{2k-2} + (\cos(x_0) - \cos(\frac{x}{2}))^{2k-2} \Big)/8.$$

That way our heuristics points out that $f_2(x)$ has the zeros $\pm 2x_0$ with the same multiplicity as $f(x)$. The new function $f_2(x)$ can be seen as a slightly changed version of the original f with the new zeros $\pm 2x_0$. We observe that the case $x_0 = 0$ is exceptional because $2x_0 = x_0 = 0$ and we can use the same prolongation and restriction operators in every step.

Remark 7 *In general, this change of the zeros $\pm x_0$, $\pm 2x_0$, $\pm 4x_0$, and so on, can lead to difficulties if in the course of the Multigrid method we reach a zero near $(2j+1)\pi/2$; then x_0 and $x_0 + \pi$ are both zeros of f, and f_2 will have $2x_0$ as a zero of higher multiplicity than $f(x)$; then our reasoning shows that the above approach will lead to a deterioration of the condition number of the related linear system.*

We would now like to confirm Remark 7 in numerical experiments. Therefore we employ a very simple BPX-type preconditioner: However, we do not invert the matrix on the coarsest level like in (2.34), but employ only a Richardson solve there instead, i.e. our preconditioner is

$$I + P_1(I + P_2(I + \ \cdots \ (I + P_k P_k^T)\cdots)P_2^T)P_1^T \tag{4.4}$$

with the matrices $P_1, \ldots, P_k$ denoting the prolongation operators on the individual levels. The following tables compare iteration numbers for this preconditioner employed inside the CG method.

number of unknowns	$\epsilon = 0.2$	$\epsilon = 0.15$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.001$
256	60	88	133	159	166
512	83	111	200	246	265

Table 4.1. CG Iteration numbers for additive preconditioners:
$f(x) = (\cos(\phi_0) - \cos(x))^2$, $b(x) = (\cos(\phi_0) + \cos(x))^2$ and $\phi_0 = \pi/4 + \epsilon$.

number of unknowns	$\epsilon = 0.2$	$\epsilon = 0.15$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.001$
256	158	187	213	221	232
512	294	350	396	416	422

Table 4.2. CG Iteration number for additive preconditioners:
$f(x) = (\cos(\phi_0) - \cos(x))^2$, $b(x) = (\cos(\phi_0) + \cos(x))^2$ with $\phi_0 = \pi/2 + \epsilon$.

4.1.4 Projections onto every m-th column – the first idea for a resort

In order to avoid the problem outlined in Remark 7 we could e.g. introduce elementary projections onto every third, fourth, or general m-th column/row

of A_n. Instead of reducing A_n to half size we use $A_{n/m}$. To this aim we apply elementary projections $I_{n,n/m}$. Making use of our heuristics (4.1) once again this is related to picking every m-th entry out of $\tilde{f}(x) = b(x)^2 f(x)$. Then we get

$$f_m(x) = \frac{1}{m} \sum_{j=0}^{m-1} \tilde{f}(\frac{x + 2j\pi}{m}) = \frac{1}{m} \sum_{j=0}^{m-1} b^2(\frac{x + 2j\pi}{m}) f(\frac{x + 2j\pi}{m}) \ ,$$

which is again a 2π-periodic function. If f has a zero x_0 we have to generate a zero with the same multiplicity in f_m. This can be achieved by defining

$$b(x) = \Big(\prod_{j=1}^{m-1} (\cos(x_0) - \cos(x - \frac{2j\pi}{m}) \Big)^k \ .$$

Then the function f_m will have a zero at mx_0 with the desired multiplicity. Therefore, by choosing m in every step of the multigrid method we could at least avoid the exceptional case $x_0 \approx (2j + 1)\pi/2$.

However, it has long been known that multigrid algorithms usually work best if the restriction yields a reduction to every second column. This has been confirmed in all our numerical experiments which have been leading us to the conclusion that the new algorithmic idea outlined in this subsection is not very recommendable for use in practice. In the following table we simply compare iteration counts for additive preconditioners for $A = tridiag(-1, 2, -1)$.

number of unknowns	128	256	512	1024	2048	4096
reduction 1:2 per step; 5 levels used	18	18	19	20	21	21
reduction 1:4 per step; 3 levels used	37	40	41	41	41	41

Table 4.3. CG Iteration numbers for additive preconditioners of the form (4.4): We clearly observe that a reduction to every second column is superior to a reduction to every fourth column.

4.1.5 Diagonal scaling – the better resort

Regarding the fact that

$$(\cos(x_0) - \cos(x))^2 = (1 - \cos(x - x_0)) * (1 - \cos(x + x_0)) \qquad (4.5)$$

there is a much simpler strategy to solve the problems from Example 3. The product form (4.5) allows us to devise a simple and effective preconditioner: We solve the two matrix problems related to $1 - \cos(x \pm x_0)$ (e.g. by multigrid) and use the result to precondition conjugate gradients. Note that the matrices related to $1 - \cos(x \pm x_0)$ can be treated very efficiently by multigrid, because they are nothing else than diagonally scaled versions of $tridiag(-0.5, 1, -0.5)$ – i.e. the one-dimensional Laplacian from Example 1: With the two (orthogonal) diagonal matrices

$$Q_1 = diag(1, e^{ix_0}, e^{2ix_0}, ...), \qquad Q_2 = diag(1, e^{-ix_0}, e^{-2ix_0}, ...) \tag{4.6}$$

we can write

$$Q_1 * tridiag(-0.5, 1, -0.5) * Q_2 = tridiag(-0.5\, e^{ix_0},\ 1,\ -0.5\, e^{-ix_0}). \tag{4.7}$$

It is plain that the diagonal scaling strategy (4.7) can be applied to any Toeplitz matrix in order to shift the generating function along the x-axis. Furthermore, in the case that we have only got a single zero $x_0 \in]-\pi, \pi]$, we see that the whole multigrid algorithm is simplified by shifting x_0 to the origin: Then we can use the same kind of transfer operators in every step – i.e. standard prolongations and restrictions according to $b(x) = (1+\cos(x))^k$. This fact will be of major importance when we introduce our new multigrid algorithms with natural coarse grid operators in section 4.3.

4.2 Existing results on multigrid for Toeplitz systems

4.2.1 Suitable smoothers

Obviously, it would not make sense to employ Gauss-Seidel as a smoother in a multigrid algorithm for Toeplitz systems unless the matrices were sparse. Hence we will use this chance to study the smoothing property of the Jacobi method in more detail. The following theorem proves that Jacobi with an appropriate damping parameter satisfies the algebraic smoothing property (2.17) introduced in subsection 2.4.1; our presentation is according to the seminal paper by Ruge and Stüben [92], pp. 84:

Theorem 5 (Algebraic smoothing property of damped Jacobi)
Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix and

$$\eta = \rho(diag(A)^{-1}A).$$

Let the norms $\|\cdot\|_A$ and $\|\cdot\|_{DA}$ be defined as in (2.35) and (2.37), respectively, and $G = I - \omega diag(A)^{-1}A$ denote the iteration matrix of the damped Jacobi method.
Then Jacobi relaxation with damping parameter $\omega \in]0, \frac{1}{\eta}[$ satisfies the algebraic smoothing property (2.17)

$$\|Ge\|_A^2 \leq \|e\|_A^2 - \alpha\|e\|_{DA}^2$$

with $\alpha \leq \omega(2-\omega\eta)$ for all $e \in \mathbb{R}^n$. Furthermore, the choice $\omega^* = 1/\eta$ gives the optimal damping parameter (in the sense of leading to the smallest α).

Proof. First, let us abbreviate: $D = diag(A)$. Now a simple calculation shows that

$$\|Ge\|_A^2 = \|e\|_A^2 - \langle(\frac{2}{\omega}D - A)\omega D^{-1}Ae, \omega D^{-1}Ae\rangle,$$

i.e. we only need to prove that

$$\alpha\|e\|_{DA} = \alpha\langle D^{-1}Ae, Ae\rangle \leq \langle(\frac{2}{\omega}D - A)\omega D^{-1}Ae, \omega D^{-1}Ae\rangle,$$

which in turn can be written as

$$\langle Ae, e\rangle \leq (\frac{2}{\omega} - \frac{\alpha}{\omega^2})\langle De, e\rangle \tag{4.8}$$

As it is well-known that for s.p.d. matrices B_1 and B_2 there holds

$$\langle B_1 e, e\rangle \leq c\langle B_2 e, e\rangle \quad \Leftrightarrow \quad \rho(B_2^{-1}B_1) \leq c$$

we can follow from (4.8) that

$$\eta = \rho(D^{-1}A) \leq \frac{2}{\omega} - \frac{\alpha}{\omega^2}$$

or expressed in terms of α

$$\alpha \leq \omega(2 - \omega\eta)$$

The right hand side is positive for $0 < \omega < 2/\eta$ and takes its minimum for $\omega = 1/\eta$ – and hence we have proved Theorem 5.

From Theorem 2 (ii) we know that the spectrum of a Toeplitz matrix A_n generated by a continuous function f lies in the range of f – and hence the above Theorem 5 is very helpful in finding appropriate damping parameters for the Jacobi method: Obviously,

$$\omega_1 = \frac{a_0}{\max_{\theta\in[-\pi,\pi]} f(\theta)} \quad \text{and} \quad \omega_2 = \frac{2a_0}{\max_{\theta\in[-\pi,\pi]} f(\theta)} \tag{4.9}$$

give sensible damping parameters for presmoothing and postsmooting, respectively. (Note that a_0 stands for main diagonal entry of A_n.)

In all the following numerical experiments in this chapter we employ two steps of the Jacobi method for pre- and postsmoothing in our multigrid cycles. The smoothing parameters will be the values from (4.9), i.e. we use ω_1 for pre- and ω_2 for postsmoothing.

Finally, note that for Toeplitz matrices there is no actual distinction between Jacobi and Richardson iterations. To be more precise: A damped Jacobi method with damping parameter ω applied to a Toeplitz system can be interpreted as a damped Richardson iteration with parameter $\tilde{\omega} = \omega/a_0$ for this system. This means that as long as our systems are Toeplitz, we can also see the smoothers we use as damped Richardson methods with the pre- and postsmoothing parameters $\tilde{\omega}_1 = \omega_1/a_0$ and $\tilde{\omega}_2 = \omega_2/a_0$, respectively.

4.2.2 The work of R. Chan and collaborators

R. Chan and his former Ph.D. students H. Sun and Q. Chang were among the first researchers who studied multigrid methods for Toeplitz matrices in [25] and [104]. Our research is based on their results and as our approach to the subject is relatively close to theirs, we would very briefly like to summarize the results of their paper [25]:

First, the paper deals with twogrid methods: They are investigating on conditions under which Theorem 1 is satisfied – and hence their reasoning is based entirely on Galerkin coarsening. They prove the following theorem (see [25], Th. 4):

Theorem 6 (Twogrid for Toeplitz with Galerkin coarsening)
Let $A \in \mathbb{R}^{n\times n}$ be a Toeplitz matrix generated by an even function f that satisfies

$$\min_{\theta\in[-\pi,\pi]} \frac{f(\theta)}{1 \pm \cos(l\theta)} > 0 \tag{4.10}$$

for some integer l. If $a_l < 0$ we choose the prolongation operator according to $b(x) = 1 + \cos(lx)$. If $a_l > 0$ we use the prolongation operator according to $b(x) = 1 - \cos(lx)$, instead.
Then the condition (2.38) from Theorem 1 on twogrid convergence with Galerkin coarse grid operators is satisfied, i.e. there exists $\beta > 0$ such that

$$\min_{e^H\in\mathbb{R}^{n_H}} \|e^h - Pe^H\|_D^2 \le \beta\|e^h\|_A^2 \qquad \text{for all} \quad e^h \in \mathbb{R}^{n_h}.$$

If the smoother satisfies the algebraic smoothing condition (2.17), then the convergence factor of the twogrid method is uniformly bounded below 1 independent of the matrix size.

The proof of Theorem 6 in [25] consists of two separate steps: First, it is established that (2.38) holds for weakly diagonally dominant Toeplitz matrices. Then it is pointed out that (2.38) carries over to problems satisfying (4.10) via arguments from Theorem 2 on the spectrum of Toeplitz matrices. Note that Theorem 6 also proves formally that for Toeplitz problems generated by $f(x) = 1\pm\cos(m*x)$ the transfer operators suggested in Remark 6 are indeed appropriate and lead to a convergent twogrid algorithms. However, this is certainly not surprising since we can interpret any matrix generated by $f(x) = 1 - \cos(m * x)$ as a discretization of the 1D Laplacian.
Furthermore, under the conditions of Theorem 6 the paper [25] also proves convergence for V-cycle solvers. However, in general this result is level-dependent. In case the generating function $f(x)$ satisfies

$$c_2(1 \pm \cos(lx)) \ge f(x) \ge c_1(1 \pm \cos(lx))$$

for some integer l and positive constants c_1 and c_2, then it can even be proved that the convergence factor of the V-cycle solver is also uniformly bounded below 1 independent of the matrix size. (Note that proofs for V-cycles always imply that the results also hold for W-cycles.)
In their numerical experiments R. Chan, Q. Chang and H. Sun choose Jacobi smoothing – and it was them who suggested the value of the damping parameters for pre- and postsmoothing we introduced in (4.9).

Finally, we would like to emphasize that due to (4.10) this theory only includes nonnegative generating functions with zeros of orders less or equal 2.

4.2.3 The work of Serra

Multigrid methods for Toeplitz systems were first proposed by Fiorentino and Serra in the two papers [47] and [48]. In [49] they try to extend their work to indefinite symmetric Toeplitz systems via an additive algorithm. In all their papers, the main focus lies on Toeplitz systems with a generating function in the Wiener class having a single zero $x_0 \in]-\pi, \pi]$ of finite order. Fiorentino and Serra use prolongations and restrictions corresponding to the function (4.2) and they always employ Galerkin coarse grid operators and Richardson smoothers in their algorithms.

Serra still remains a very active researcher in this field: Very recently, he also gave a detailled proof of convergence for twogrid solvers based on the sole assumption that the generating function f has a single zero x_0 of finite order at the origin in [99]. To be more precise: He refines Theorem 6 for the case $l = 1$ in the sense that the zero single $x_0 = 0$ may have order higher than 2 and that he allows higher order transfer operators (see [99], Lemma 5.2). Again, the proof is based on Theorem 1, i.e. Serra works on the basis of Ruge's and Stüben's results [92] and establishes convergence showing that (2.38) is satisfied.

The work of Serra and Fiorentino is driven by pointing out close relations between Toeplitz matrices and matrices from trigonometric algebras: In particular, they also give multigrid convergence proofs for τ-matrices, i.e. the algebra of matrices that can be diagonalized by the fast sine transform. For τ-matrices Galerkin coarsening is rather convenient, because it can be shown that under certain conditions the coarse grid representation will again be a τ-matrix (see [47], [48] for details). Very recently, Serra and Tablino also presented multigrid approaches for circulant matrices in [100] and [101].

4.3 Generating functions with a single zero in $]-\pi,\pi]$

In this section we shall assume that our Toeplitz matrices are related to a nonnegative generating function with a unique zero $x_0 \in]-\pi,\pi]$. (Note that this does neither include the matrices from Example 3 nor the sparse matrices from Remark 6 in the case $m > 1$.)

4.3.1 Natural coarse grid operator

In subsection 4.1.5 we have presented a number of arguments for scaling the Toeplitz system A_n with the (orthogonal) diagonal matrices given by (4.6) before treating it by multigrid, i.e. using

$$Q_1 = diag(1, e^{ix_0}, e^{2ix_0}, ...), \qquad Q_2 = diag(1, e^{-ix_0}, e^{-2ix_0}, ...)$$

we scale the original matrix A_n such that

$$A_n^{(scaled)} = Q_2 * A_n * Q_1 \tag{4.11}$$

has a single isolated zero at the origin. In particular, we are then enabled to use standard transfer operators corresponding to $b(x) = (1 - \cos(x))^k$ in every step.
However, we have not yet presented a way to handle the problem that our Galerkin coarse grid operators lose their Toeplitz structure. R. Chan and collaborators already pointed out in [25] that whenever standard linear interpolation corresponding to $b(x) = 1 + cos(x)$ is appropriate (i.e. in the case $l = 1$ in Theorem 6) then we can only be sure to preserve Toeplitz structure on all the coarse levels if the size n of the matrix is of the form $n = 2^q - 1$ with q integer. Otherwise perturbations of low rank can be introduced. But note that this loss of Toeplitz structure may cause severe difficulties when we go down to lower levels.

There is a very simple resort: **First scale the matrix according to (4.11) – and then employ a natural coarse grid operator!** Anyway, let us start from scratch: Considering the one-dimensional Laplace problem from Example 1 with the system matrices

$$A_n = (1/n^2) * tridiag(-1, 2, -1)$$

we have already mentioned in 2.4.3 and 2.6 that natural coarse grid operators work out perfectly and convergence proofs have been established since the late seventies.
Now let us switch over to general Toeplitz matrices belonging to a generating function f that satisfies (4.10) in Theorem 6 with $l = 1$ and has a single zero at the origin, i.e.

$$\min_{\theta \in [-\pi,\pi]} \frac{f(\theta)}{1 - cos(\theta)} > 0 \tag{4.12}$$

We have pointed out in the last section that the convergence proofs for twogrid algorithms with Galerkin operators for Toeplitz systems are all based on showing that condition (2.38) is satisfied, i.e. that

$$\min_{e^H \in \mathbb{R}^{n_H}} \|e^h - Pe^H\|_D^2 \leq \beta \|e^h\|_A^2 \qquad \text{holds for all} \quad e^h \in \mathbb{R}^{n_h};$$

and again this is done by tracing these matrices back to the sparse matrices from Remark 6. However, the Galerkin operator itself is not needed in establishing (2.38), it only comes in via Theorem 1 taken from Ruge's and Stüben's theory [92]. Furthermore, we know from Theorem 2 that a Toeplitz matrix generated by a function with a single zero of order 2, like e.g. $f(x) = x^2$, will have almost the same spectral properties as the 1D Laplacian (2.1).
These observations strongly motivate the idea simply to mimic a multigrid algorithm with a natural coarse grid operator for general Toeplitz matrices satisfying (4.12), i.e. just like we would do it for the Laplacian our coarse level matrix is nothing but an appropriately scaled Toeplitz matrix of half size $n/2$ corresponding to the same generating function $f(x)$.
When we program such a multigrid algorithm with a natural coarse grid operator, it is crucial to be aware of Remark 3 on the correct scaling of the defects: Note that the scaling factor fac in equation (2.28), i.e.

$$A_{k-1} e_{k-1}^{(j)} = fac * r_{k-1}^{(j)},$$

needs to reflect the order l of the zero of the function f, i.e. $fac = 2^l$. (In particular, as there is no physical grid connected with a Toeplitz matrix generated by a function like e.g. $f(x) = x^2$, we will most certainly prefer to have that factor on the right hand side of our coarse grid equation.) In other words: Our multigrid algorithms need to take into account very carefully the orders of the zeros of the generating functions involved.
Finally, we wish to emphasize once again that our idea of using a natural

coarse grid operator crucially depends on the fact that our single zero $x_0 \in]-\pi, \pi]$ is indeed shifted to the origin for otherwise (4.12) could not be satisfied. (Furthermore, it is plain from Remark 6 that a natural coarse grid operator could never be appropriate in case of a single isolated zero of finite order at $x_0 = \pi$.)

4.3.2 Numerical results for zeros of order at most two

In order to show the computational feasibility of our approach to employ natural coarse grid operators within multigrid algorithms for Toeplitz matrices we shall present plenty of numerical experiments: We will deliberately choose the matrix sizes in our numerical experiments to be of the forms $n = 2^q$ or $n = 2^q + 1$ in most cases; for these matrix sizes Toeplitz structure would get lost on the coarse levels if Galerkin coarsening were used. We have programmed W-cycle algorithms which we will use both as solvers and as preconditoners, abbreviated by "MG-Solver" and "MG-Prec." in our tables. As we already said in section 2.1, we always use two steps of damped Jacobi for pre- and postsmoothing employing the damping parameters from (4.9). We will only give numerical results for dense Toeplitz matrices as preserving Toeplitz structure on coarser levels is only an issue in this case. Our W-cycle algorithms will be compared to unpreconditioned CG, circulant preconditioned CG using the T. Chan preconditioner [28] introduced in subsection 3.2.3 and – whenever there are only zeros of even order – to the banded preconditioners introduced in Theorem 4. The abbreviations in our tables will be "CG", "CIRCULANT" and "BANDED", respectively.

Example 4 (Generating functions with zeros of order at most 2) *We will study dense Toeplitz matrices with a single zero of order at most two at the origin generated by the following functions:*
(a) $f_1(x) = x^2$ *with the Fourier expansion*

$$f_1(x) = \frac{\pi^2}{3} + 4 * \sum_{j=1}^{\infty} \frac{(-1)^j}{j^2} \cos(j * x)$$

(b) $f_2(x) = (x/4) * \sin(x/2)$ *with the Fourier expansion*

$$f_2(x) = \frac{1}{\pi} + \frac{2}{\pi} * \sum_{j=1}^{\infty} \frac{(-1)^j * (4 * j^2 + 1)}{(2 * j - 1)^2 * (2 * j + 1)^2} \cos(j * x)$$

(c) $f_3(x) = |x|$ *with the Fourier expansion*

$$f_3(x) = \frac{\pi}{2} - \frac{4}{\pi} * \sum_{j=1}^{\infty} \frac{2}{(2*j-1)^2} \cos((2*j-1)*x)$$

(d) $f_4(x) = |\sin(x/2)|$ *with the Fourier expansion*

$$f_4(x) = \frac{2}{\pi} - \frac{4}{\pi} * \sum_{j=1}^{\infty} \frac{1}{(2*j-1)*(2*j+1)} \cos(j*x)$$

According to our theory the prolongation operator to be used for multigrid treatment of all the matrices from Example 4 is standard linear interpolation corresponding to $b(x) = 1 + cos(x)$.

Let us first take a look at $f_1(x) = x^2$. Here we have a single zero of order 2; hence like in the case of the Laplacian we have to use $fac = 4$ in (2.28) in our multigrid algorithms with natural coarse grid representations:

number of unknowns	MG-Solver	CG	CIRCULANT	BANDED
1024	12	1723	39	9
2048	12	> 2000	52	9
4096	12	> 2000	61	9
8192	12	> 2000	72	9
16384	12	> 2000	92	9
32768	12	> 2000	118	9

Table 4.4. Iteration numbers for generating function $f_1(x) = x^2$.

For $f_2(x) = (x/4) * \sin(x/2)$ we have again a zero of order 2 and we can expect pretty much the same behaviour as for $f_1(x)$.

number of unknowns	MG-Solver	CG	CIRCULANT	BANDED
256	11	183	22	7
512	11	360	27	7
1024	12	773	37	7
2048	12	1406	49	7
4096	12	> 2000	59	7
8192	12	> 2000	68	7

Table 4.5. Iteration numbers for generating function $f_2(x) = (x/4)*\sin(x/2)$.

In the case $f_3(x) = |x|$ the zero is no longer of even order and hence the banded preconditioners from [18] and [21] are no longer available. As f_3 has a discontinuous derivative at the origin, we are thinking of it as a zero of order 1 and use $fac = 2$ to scale the defects in (2.28):

number of unknowns	MG-Solver	MG-Prec.	CG	CIRCULANT
2049	5	5	162	10
4097	5	5	227	11
8193	5	5	317	11
16385	5	5	444	13
32769	5	5	620	13
65537	5	5	867	13

Table 4.6. Iteration numbers for generating function $f_3(x) = |x|$.

For $f_4(x) = |\sin(x/2)|$ the situation is the same as for $f_3(x)$:

number of unknowns	MG-Solver	MG-Prec.	CG	CIRCULANT
2049	5	7	131	10
4097	5	7	184	11
8193	5	7	258	11
16385	5	7	360	12
32769	5	7	503	13

Table 4.7. Iteration numbers for generating function $f_4(x) = |\sin(x/2)|$.

Tables 4.4 to 4.7 show very clearly that our new multigrid algorithms lead to fast convergence with iteration counts independent of the number of unknowns involved. Hence they give very efficient solvers of optimal computational complexity $O(n \log n)$. Furthermore, our multigrid method has no problem at all with the fact that $f_3(x) = |x|$ and $f_4(x) = |sin(x/2)|$ are not differentiable at the origin: On the contrary, the fact that the order of the zero is lower than 2 leads to even faster convergence. We also observe that our W-cycle solvers perform so efficiently that there is no point in employing the W-cycles as preconditioners for CG.
For the T. Chan circulant preconditioner [28] we observe that the iteration numbers grow the faster the higher the order of the zero. Furthermore, the

rapidly rising iteration counts for unpreconditioned CG underline that the matrices we studied are very ill-conditioned. Finally, it is not surprising that the banded Toeplitz preconditioners from [18], [21] do an excellent job for zeros of even order.

4.3.3 Numerical results for zeros of higher order

Example 5 (Generating functions with zeros of higher order)
We will study generating functions for dense Toeplitz matrices with a single zero of order higher than 2 at the origin generated by the following functions:
(a) $f_5(x) = x^4$ *with the Fourier expansion*

$$f_5(x) = \frac{\pi^4}{5} + \sum_{j=1}^{\infty}\left(\frac{48}{(2*j-1)^4} - \frac{8\pi^2}{(2*j-1)^2}\right)\cos((2*j-1)*x) + \\ + \sum_{j=1}^{\infty}\left(\frac{2\pi^2}{j^2} - \frac{6}{2*j^4}\right)\cos(2*j*x)$$

(b) $f_6(x) = |x|^3$ *with the Fourier expansion*

$$f_6(x) = \frac{\pi^3}{4} + \frac{2}{\pi} * \sum_{j=1}^{\infty}\left(\frac{12}{(2*j-1)^4} - \frac{3\pi^2}{(2*j-1)^2}\right)\cos((2*j-1)*x) + \\ + \sum_{j=1}^{\infty}\frac{6\pi}{(2*j)^2}\cos(2*j*x)$$

Now looking at (4.2) we expect to use a higher order prolongation operator corresponding to $b(x) = (1+\cos(x))^2$ – this will be abbreviated by "(P2)" in the following tables. Anyway, it will most certainly be very interesting try standard linear interpolation corresponding to $b(x) = 1+\cos(x)$ (abbreviated by "(P1)" in our tables) as well:

Let us first take a look at $f_5(x) = x^4$. We are dealing with a zero of fourth order and we need to use $fac = 16$ in (2.28) to scale the defects:

number of unknowns	MG-Solver (P1)	MG-Solver (P2)	CIRCULANT	BANDED
511	29	33	108	15
1023	29	33	137	15
2047	29	33	240	15
4095	29	33	520	15
8191	29	33	905	15
16383	29	33	1632	15
32767	29	33	> 2000	15
65535	29	33	> 2000	15

Table 4.8. Iteration numbers for generating function $f_5(x) = x^4$.

The third derivate of the function $f_6(x) = |x|^3$ is discontinuous at the origin – and hence we treat it as a zero of order 3 and use $fac = 8$ in (2.28):

number of unknowns	MG-Solver (P1)	MG-Solver (P2)	MG-Prec. (P1)	MG-Prec. (P2)
2047	14	19	13	11
4095	14	19	13	11
8191	14	19	13	11
16383	14	19	13	11
32767	14	19	13	11
65535	14	19	13	11

Table 4.9. Iteration numbers for generating function $f_6(x) = |x|^3$.

We observe that in practice it is sufficient to use standard linear interpolation for prolongation and restriction. Surprisingly, if we use our W-cycles as stand-alone solvers then in both cases iteration counts are even smaller if we use the transfer operators corresponding to $b(x) = 1 + \cos(x)$. Anyway, this confirms the well known advice of multigrid practitioners that higher order interpolations might frequently not pay off.

4.3.4 Summary

In this section we have been presenting a new efficient way to solve Toeplitz systems corresponding to an underlying function having a single zero $x_0 \in]-\pi, \pi]$ of finite order: One first scales the matrix with the diagonal matrices (4.6) according to (4.11) in order to shift the zero to the origin and then solves the scaled system by a multigrid algorithm employing a natural coarse grid operator.

4.4 Generating functions with equidistant zeros of finite order

4.4.1 Equidistant zeros

The case when the generating function has more than one zero of finite order is certainly much more complicated.
However, we have already addressed a fairly simple example in Remark 6: For generating functions of the form $f(x) = 1 \mp \cos(m*x)$, m integer, appropriate prolongation operators are given by $b(x) = 1 \pm \cos(m * x)$.

As we learn from Theorem 6 the prolongation operators $b(x) = 1 + \cos(m*x)$ are applicable in case the generating function of our Toeplitz matrix has m equidistant zeros of order at most 2 in the interval $[0, 2\pi[$ one of which needs to be at the origin, i.e. the generating function has the zeros $x_0 = 0, x_1 = \frac{2\pi}{m}, \ldots, x_{m-1} = \frac{2*(m-1)*\pi}{m}$.
Now we can again apply our reasoning from the previous section: In case none of our m equidistant zeros of order at most 2 is at the origin, we first scale the matrix according to (4.11). Then we observe that $f(x) = 1 - \cos(m*x)$ can be again interpreted as a discretization of the 1D Laplacian – and hence multigrid algorithms with natural coarse grid operators will work very well. Like in the previous section we can try to mimic such multigrid algorithms with natural coarse grid operators for Toeplitz matrices generated by functions f which satisfy (4.10) and have a zero at the origin, i.e. there needs to hold

$$\min_{\theta \in [-\pi, \pi]} \frac{f(\theta)}{1 - cos(m * \theta)} > 0 \tag{4.13}$$

Thus we shall try to use multigrid algorithms with natural coarse grid operators and the prolongations $b(x) = 1 + \cos(m * x)$ for functions satisfying

relation (4.13).

4.4.2 A block interpretation

At this place we would like to report an interesting observation: Let us again take a look at the matrix connected with $f(x) = 1 - \cos(m * x)$ and the corresponding transfer operators $b(x) = 1 + \cos(m * x)$. Now we can interpret this also in terms of matrix valued functions:

$$f(x) = I_m - \cos(I_m * x) = I_m * (1 - \cos(x))$$

is treated by prolongations of the form

$$b(x) = I_m + \cos(I_m * x) = I_m * (1 + \cos(x))$$

with I_m denoting the m-by-m identity matrix. Thus we can view this case as standard multigrid applied to Block Toeplitz matrices with m-by-m blocks. By inserting block matrices different from the identity we can carry over this idea to general Block Toeplitz matrices (i.e. also without Toeplitz blocks). This will be subject to future research.
However, note that the strategy outlined in subsection 4.4.1 also applies to cases like e.g. Toeplitz matrices belonging to $f(x) = x * \sin(x)$ which are not covered by the above block interpretation (see Example 6 (c) in subsection 4.4.4 for the Fourier expansion). As $f(x) = x * \sin(x)$ has the two zeros $x_0 = 0$ and $x_1 = \pi$ we can interpret the appropriate prolongation

$$b(x) = (1 - \cos(x - x_0)) * (1 - \cos(x - x_1)) = \frac{1}{2} * (1 - \cos(2 * x)) \quad (4.14)$$

analogously to (4.2) as the product of the two prolongations corresponding to x_0 and x_1. This interpretation has previously been given by Serra in [99], although he has not published any numerical experiments to confirm it.

4.4.3 Algorithmic issues

Before presenting any results of our multigrid algorithms we would like to state clearly how the transfer operators connected to $b(x) = 1 + \cos(m * x)$ need to be implemented. As first described by R. Chan, Q. Chang and H. Sun

in [25] the prolongation operator belonging to $b(x) = 1 + \cos(m * x)$ needs to match our block interpretation and have the following structure

$$\mathbf{P_m} = \begin{pmatrix} 0.5I_m & & & & \\ I_m & & & & \\ 0.5I_m & 0.5I_m & & & \\ & I_m & & & \\ & \cdots & \cdots & & \\ & & 0.5I_m & 0.5I_m & \\ & & & I_m & \\ & & & 0.5I_m & \end{pmatrix} \tag{4.15}$$

Of course, the restriction is simply $R_m = P_m^T$. Below, we are also giving a sample MATLAB implementation of the prolongation operator P_2 applicable in the case that the number of unknowns is of the form $n = 2^q$:

```
% n is the number of unknowns: It needs to be a multiple of 4
% P is the prolongation operator
%
vec = ones(n,1);
T = spdiags([0.5*vec 0*vec vec 0*vec 0.5*vec], -2:2, n, n);
%
for index=1:(n/4)
P(:,2*index-1)= T(:,4*index-1);
P(:,2*index)= T(:,4*index);
end;
%
```

4.4.4 Numerical results

In the following we will test our multigrid algorithms employing natural coarse grid operators for problems with equidistant zeros in $[0, 2\pi[$.

Example 6 (Generating functions with zeros 0 and π)
We will study generating functions for dense Toeplitz matrices with the two zeros $x_0 = 0$ and $x_1 = \pi$ of orders at most 2 generated by the following functions:
(a) $f_7(x) = x^2 * (x - \pi)^2$ (– defined on $[0, \pi]$ and then evenly extended to

$[-\pi, 0[$ *–) with the Fourier expansion*

$$f_7(x) = \frac{\pi^4}{30} - \sum_{j=1}^{\infty} \frac{6}{2 * (2 * j)^4} * \cos(2 * j * x)$$

(b) $f_8(x) = |\sin(x)|$ *with the Fourier expansion*

$$f_8(x) = \frac{2}{\pi} - \frac{4}{\pi} * \sum_{j=1}^{\infty} \frac{1}{(2 * j - 1) * (2 * j + 1)} * \cos(2 * j * x)$$

(c) $f_9(x) = x * \sin(x)$ *with the Fourier expansion*

$$f_9(x) = 1 - \frac{1}{2}\cos(x) - 2 * \sum_{j=2}^{\infty} \frac{(-1)^j}{(j - 1) * (j + 1)} * \cos(j * x)$$

$f_7(x) = x^2 * (x - \pi)^2$ has two isolated zeros of order 2. Hence we expect to treat it like $f(x) = 1 - \cos(2 * x)$ and we use $fac = 4$ in (2.28) for scaling the defects. Furthermore, note that the matrices generated by $f(x) = 1 - \cos(2 * x)$ are available as band Toeplitz preconditioners in this case:

number of unknowns	MG-Solver	CG	CIRCULANT	BANDED
513	11	303	25	7
1025	12	624	31	7
2049	12	1238	41	7
4097	12	> 2000	41	7
8193	12	> 2000	55	7
16385	12	> 2000	63	7

Table 4.10. Iteration numbers for generating function $f_7(x) = x^2 * (x - \pi)^2$.

$f_8(x) = |\sin(x)|$ has discontinuous derivatives at both $x_0 = 0$ and $x_1 = \pi$. Like for $f_4(x)$ from Example 4 we use $fac = 2$ in (2.28):

number of unknowns	MG-Solver	MG-Prec.	CG	CIRCULANT
2049	5	6	119	13
4097	5	6	166	13
8193	5	6	216	13
16385	5	6	264	14
32769	5	6	365	14
65537	5	6	509	16

Table 4.11. Iteration numbers for generating function $f_8(x) = |\sin(x)|$.

$f_9(x) = x * \sin(x)$ is certainly the most challenging example; the zero $x_0 = 0$ has order 2 whereas $x_1 = \pi$ has order 1. But how should we scale the defects correctly in (2.28) in this case? The arithmetic means of the orders of the two zeros is 1.5; hence we use $fac = 2^{1.5} = 2\sqrt{2}$:

number of unknowns	MG-Solver	MG-Prec.	CG	CIRCULANT
1025	9	9	452	32
2049	9	9	794	43
4097	9	9	1476	58
8193	9	9	> 2000	68
16385	9	9	> 2000	83
32769	9	9	> 2000	104

Table 4.12. Iteration numbers for generating function $f_9(x) = x * \sin(x)$.

Again, we observe optimal computational behaviour of our multigrid algorithms for all problems from Example 6. Like in the case of a single isolated zero studied in the previous section, our W-cycle solvers are so efficient that there is no point in using them as preconditioners. The natural coarse grid operators take into account very carefully the orders of the zeros of the generating function. Thus we can confirm numerically that the multigrid algorithms suggested in section 4.3 carry over to the case of generating functions with m equidistant zeros in $[0, 2\pi[$.

4.4.5 Outlook, conclusions and further remarks

In this and the previous section we have introduced multigrid algorithms with natural coarse grid operators for Toeplitz matrices generated by non-negative functions with zeros of finite order. The idea is based on the fact that multigrid methods with natural coarse grid operators are known to work well for matrices generated by $f(x) = 1 - \cos(m * x)$; and we mimic these algorithms for dense Toeplitz matrices with a very similar spectral behaviour. Our numerical results point out strikingly that our algorithms lead to solvers of optimal complexity $O(n \log n)$ for dense Toeplitz systems. As our new algorithms take into account very carefully the orders of the zeros of the generating function, they also have no problems at all in dealing with zeros of odd order. Even a case like $f_9(x) = x * \sin(x)$ with two isolated zeros of different orders can be handled without problems.

By employing natural coarse grid representations we will normally violate the variational principle underlying Galerkin coarsening. Hence it might be interesting to check whether we could get lower iteration counts by employing Galerkin coarse grid operators. We have done such comparisons: However, as soon as we build Galerkin operators Toeplitz structure will normally be lost on the coarse grids – and hence, for simplicity, we implemented the coarse grid representations as full matrices what limited the size of the problems that could be approached significantly.

number of unknowns	257	513	1025	2049	4097
$f_3(x) = \|x\|$, MG(Natural)	5	5	5	5	5
$f_3(x) = \|x\|$, MG(Galerkin)	5	5	5	5	5
$f_8(x) = \|sin(x)\|$, MG(Natural)	5	5	5	5	5
$f_8(x) = \|sin(x)\|$, MG(Galerkin)	4	4	4	4	4
$f_9(x) = x * \sin(x)$, MG(Natural)	9	9	9	9	9
$f_9(x) = x * \sin(x)$, MG(Galerkin)	9	9	9	9	9

Table 4.13. Galerkin vs. natural coarse grid operators.

Table 4.13 shows clearly that we can hardly gain any improvement in terms of iteration counts by using natural coarse grid operators. On the other hand, the disadvantages of Galerkin coarsening are severe: Toeplitz structure on the coarse levels is lost and the low rank perturbations introduced on every

level lead to a significantly more expensive setup phase.

Proving convergence of our multigrid algorithms with natural coarse grid operators will be subject to future research, but we feel that it comes out to be very difficult: As there is no longer any underlying variational principle, the only resort we can think of is to establish that our coarse grid operators satisfy the *approximation property* in the sense of Hackbusch (see [62], pp. 114), i.e. the aim would be to show that there exists a constant C_A such that

$$\|A_l^{-1} - P * A_{l-1}^{-1} * R\| \leq C_A h_n^{\alpha} \qquad \text{for all } n \geq 1$$

with A_l and A_{l-1} denoting the Toeplitz matrices on the fine and coarse level, respectively, and α standing for a constant which is determined by the smoother.
In order to prove such a statement we would need to think in terms of underlying infinite-dimensional Toeplitz operators. However, it seems to be very doubtful if we tried to pose a Dirichlet boundary value problems for the Toeplitz operator generated by e.g. $f(x) = x^2$. After all, we do not even have a discretization or an underlying physical grid.

4.5 A short view on BTTB matrices

4.5.1 Positive definite problems

In the 2D-case we consider Hermitian BTTB matrices related to a function of the form

$$f(x,y) = \sum a_{j,k} e^{ijx} e^{iky}$$

e.g. $f(x,y) = 2 - cos(x) - cos(y)$ for the 2D Laplacian from Example 2. In that case the bad condition numbers of the corresponding sequence of BTTB matrices are again caused by the zero $(x_0, y_0) = (0,0)$ of $f(x,y)$.
We are in the simple case as long as the function f has only a unique isolated zero $(x_0, y_0) \in]-\pi, \pi]^2$. Then we can try to proceed with multigrid algorithms similar to Section 4.1. For simplicity, let us first take a look at the case of a single isolated zero (x_0, y_0) of order 2. In a multigrid approach we can choose

$$b(x,y) = (\cos(x_0) + \cos(x)) * (\cos(y_0) + \cos(y)) \tag{4.16}$$

for prolongation and restriction. Note that this is nothing else than the Kronecker product of the corresponding 1D matrices. According to our heuristics

(4.1) the function f_2 associated with the Galerkin coarse grid operator is the reduction of $\tilde{f}(x,y) = b(x,y)f(x,y)b(x,y)$ to every second coefficient relative to x and y. For the matrix this is nothing else than the projection onto every second row/column and row/column block, respectively. Therefore it results

$$f_2(x,y) = \frac{1}{4} * \left(\tilde{f}(\frac{x}{2}, \frac{y}{2}) + \tilde{f}(\frac{x}{2}+\pi, \frac{y}{2}) + \tilde{f}(\frac{x}{2}, \frac{y}{2}+\pi) + \tilde{f}(\frac{x}{2}+\pi, \frac{y}{2}+\pi) \right) \quad (4.17)$$

Hence, f_2 will have the isolated zero $(2x_0, 2y_0)$ – and the prolongation $b(x,y)$ needs to have the three zeros $(x_0+\pi, y_0)$, $(x_0, y_0+\pi)$ and $(x_0+\pi, y_0+\pi)$.

In [105] Chang, Jin and Sun proved an analogous result to Theorem 6 for twogrid algorithms with Galerkin coarsening for BTTB matrices with a single isolated zero of order at most 2 at either $(x_0, y_0) = (0,0)$ or $(x_0, y_0) = (\pi, \pi)$, i.e. for the case of generating functions $f(x,y)$ satisfying

$$\min_{(x,y)\in[-\pi,\pi]^2} \frac{f(x,y)}{2 - \cos(x) - \cos(y)} > 0 \quad (4.18)$$

or

$$\min_{(x,y)\in[-\pi,\pi]^2} \frac{f(x,y)}{2 + \cos(x) + \cos(y)} > 0, \quad (4.19)$$

respectively. Again, the proof consists of establishing (2.38) such that Theorem 1 is applicable.
Anyway, for BTTB matrices it is even more important to use a natural coarse grid operator instead of Galerkin coarsening. As pointed out in [105] when using Galerkin coarsening and standard transfer operators we can only be sure to preserve BTTB structure on every coarse grid if the matrix size is of the form $n = (2^p - 1) * (2^q - 1)$ with p and q integer. More importantly, the perturbations introduced via Galerkin operators are no longer of low rank like in the Toeplitz case, but normally grow proportional to the matrix size.

However, as long as there is only a single isolated zero the resort is as simple as in the Toeplitz case: For a single zero $(x_0, y_0) \in]-\pi, \pi]^2$ we can scale our linear system first via the matrices

$$I \otimes diag(1, e^{\pm i x_0}, e^{\pm 2 i x_0}, ...) \qquad \text{and} \qquad diag(1, e^{\pm i y_0}, e^{\pm 2 i y_0}, ...) \otimes I,$$

respectively, and thus shift the zero to the origin. Then it is guaranteed that we can apply standard transfer operators corresponding to $b(x) = 2 + \cos(x) + \cos(y)$ in every step. Analogously to 4.3.1 we recall the fact that multigrid

algorithms with natural coarse grid operators have been long been known to converge for two-dimensional Laplace problems (2.4). Furthermore, we know that the 2D Laplacian (2.4) and matrices generated by e.g. $f(x,y) = x^2 + y^2$ share very similar spectral properties (see e.g. [97], [98]). Hence like before we carry over our reasoning to employ a natural coarse grid operator to functions $f(x,y)$ satisfying (4.18).
Note that (4.18) certainly includes non-separable generating functions, like e.g. $f(x,y) = 20 - 8 * \cos(x) - 8 * \cos(y) - 4 * \cos(x) * \cos(y)$ which corresponds to a 9-point discretization of the Laplacian on the unit square. However, we shall give only numerical results for separable problems in the following tables. There we list iteration counts for multigrid algorithms with natural coarse grid operators for separable BTTB problems related to generating functions from Example 4. For comparison we will also list the iteration counts for unpreconditioned CG, for the optimal BCCB preconditioner by T. Chan and J. Olkin [31] introduced in subsection 3.3.4 and – whenever the zero is of even order – for the banded BTTB preconditioner. The abbreviations will again be "CG", "CIRCULANT" and "BANDED", respectively.

$f(x,y) = x^2 + y^2$ has a zero of order 2: We need to set $fac = 4$ in (2.28) and we can also use the 2D Laplacian (2.4) as a banded preconditioner:

number of unknowns	MG-Solver	CG	CIRCULANT	BANDED
256	14	31	17	9
1024	14	77	22	9
4096	14	153	31	9
16384	14	301	44	9
65536	14	589	69	9

Table 4.14. Iteration numbers for $f(x,y) = x^2 + y^2$.

For $f(x,y) = x^2 + (y/4) * \sin(y/2)$ we are again dealing with a zero of order 2 and hence expect a similar behaviour to the previous example:

number of unknowns	MG-Solver	CG	CIRCULANT	BANDED
256	23	50	20	15
1024	24	99	26	15
4096	24	192	39	15
16384	24	368	57	15
65536	24	708	92	15

Table 4.15. Iteration numbers for $f(x,y) = x^2 + (y/4) * \sin(y/2)$.

For $f(x,y) = |x| + |y|$ we regard the zero to be of order 1. Analogously to the one-dimensional problem studied in table 4.6, we set $fac = 2$ in (2.28):

number of unknowns	MG-Solver	CG	CIRCULANT
256	7	23	9
1024	8	30	11
4096	8	40	13
16384	8	57	15
65536	8	82	16

Table 4.16. Iteration numbers for $f(x,y) = |x| + |y|$.

For the problem $f(x,y) = |x/\pi| + |\sin(y/2)|$ we again regard the zero to be of order 1 and expect similar results as for the preceding example:

number of unknowns	MG-Solver	CG	CIRCULANT
256	8	22	10
1024	9	31	12
4096	9	43	14
16384	10	61	16
65536	10	86	17

Table 4.17. Iteration numbers for $f(x,y) = |x/\pi| + |\sin(y/2)|$.

Our final example will be $f(x,y) = x^2 + |y|$: We assign the order 1.5 to the zero and the scale the defects in (2.28) with $fac = 2^{1.5} = 2\sqrt{2}$.

number of unknowns	MG-Solver	CG	CIRCULANT
256	15	41	14
1024	15	59	17
4096	15	90	20
16384	15	143	25
65536	15	234	32

Table 4.18. Iteration numbers for $f(x,y) = x^2 + |y|$.

Again, our multigrid algorithms give efficient solvers of optimal computational complexity $O(n \log n)$. The natural coarse grid operators take into account very carefully the order of the zero – and thus the algorithms are not affected at all in case the zero at the origin is of order less than 2. On the contrary, they can even take advantage from the lesser degree of ill-conditioning in that case. Not to our suprise, the iterations for the optimal BCCB preconditioner tend to grow when applied to the ill-conditioned matrices studied in tables 4.13 to 4.17.

However, our approach runs into trouble as soon as there is more than a single zero of finite order: According to (4.16) and (4.14) we would need to build prolongations $b(x,y)$ incorporating all the zeros. However, this forces us to build prolongations which are much too dense. For example, for BTTB matrices belonging to the function $f(x,y) = 2-\cos(2x)-\cos(2y)$ we would – in view of (4.16) – need to work with prolongations involving 8 "elementary" factors corresponding to the 4 zeros $(0,0), (0,\pi), (\pi,0), (\pi,\pi)$. This does not lead to computationally feasible algorithms.

4.5.2 Indefinite Problems

The situation also gets more complicated if the condition $f(x,y) = 0$ has a whole curve $(x(t), y(t))$ as solution. Certainly, we can no longer "shift" the curve of zeros to the origin by diagonal scaling. For the prolongation in view of (4.2) we need a function with zeros at $(x(t)+\pi, y(t))$, $(x(t), y(t)+\pi)$, and $(x(t)+\pi, y(t)+\pi)$. We can build such a function by setting

$$b(x,y) = f(x+\pi, y) * f(x, y+\pi) * f(x+\pi, y+\pi) \; .$$

Again, the disadvantage of this approach is that the resulting matrices connected to $f_2(x,y)$ are getting more and more dense – and we can not expect

to obtain a practical algorithm.
Let us take a look at shifted Laplacians with the underlying function of the form

$$f(x,y) = 2 - \alpha - \cos(x) - \cos(y)$$

For small α the curve described by $f(x(t), y(t)) = 0$ is nearly the circle around the origin with radius $\sqrt{2\alpha}$.

Asymptotically the eigenvalues of these BTTB matrices are given by (see e.g. the paper by Serra [98])

$$f(x_j, y_j) = 2 - \alpha - \cos(\frac{\pi j}{n+1}) - \cos(\frac{\pi k}{n+1}) \approx \frac{\pi^2(j^2+k^2)}{(n+1)^2} - \alpha \quad , \quad j,k = 1, ..., n \ .$$

As we are dealing with shifted 2D-Laplacians our matrices can be diagonalized by the 2D-Sine Transform matrix with $S_1 = \sqrt{\frac{2}{n+1}}(\sin(\pi jk/(n+1))_{j,k=1}^n$, $S_2 = S_1 \otimes S_1$, and

$$S_2 \, BT \, S_2 = diag(\lambda_j + \lambda_k - \alpha)$$

where λ_j are the eigenvalues of the 1D-Laplacian from (2.2). Hence, the eigenvalues are exactly given by

$$f(x_j, y_j) = 2 - \alpha - \cos(\frac{\pi j}{n+1}) - \cos(\frac{\pi k}{n+1}) \quad , \quad j,k = 1, ..., n$$

and the eigenvectors related to the near-zero eigenvalues are of the form

$$\sin(\pi jm/(n+1))_{m=1}^n \otimes \sin(\pi km/(n+1))_{m=1}^n$$

with

$$j^2 + k^2 \approx \alpha(n+1)^2/\pi^2 \ . \tag{4.20}$$

Hence we have to design a method that can deal with the error components in these directions. For the same problem a very sophisticated and highly promising algorithm that is related to this idea has been introduced by Brandt and Livshits based on a totally different approach [12]. There more than one coarse grid is employed in order to resolve the problematic error components.
Finally, we wish to emphasize that the above indefinite model problem should not be viewed as a Helmholtz problem: Helmholtz equations usually model scattering phenomena on an exterior domain and the system matrices can

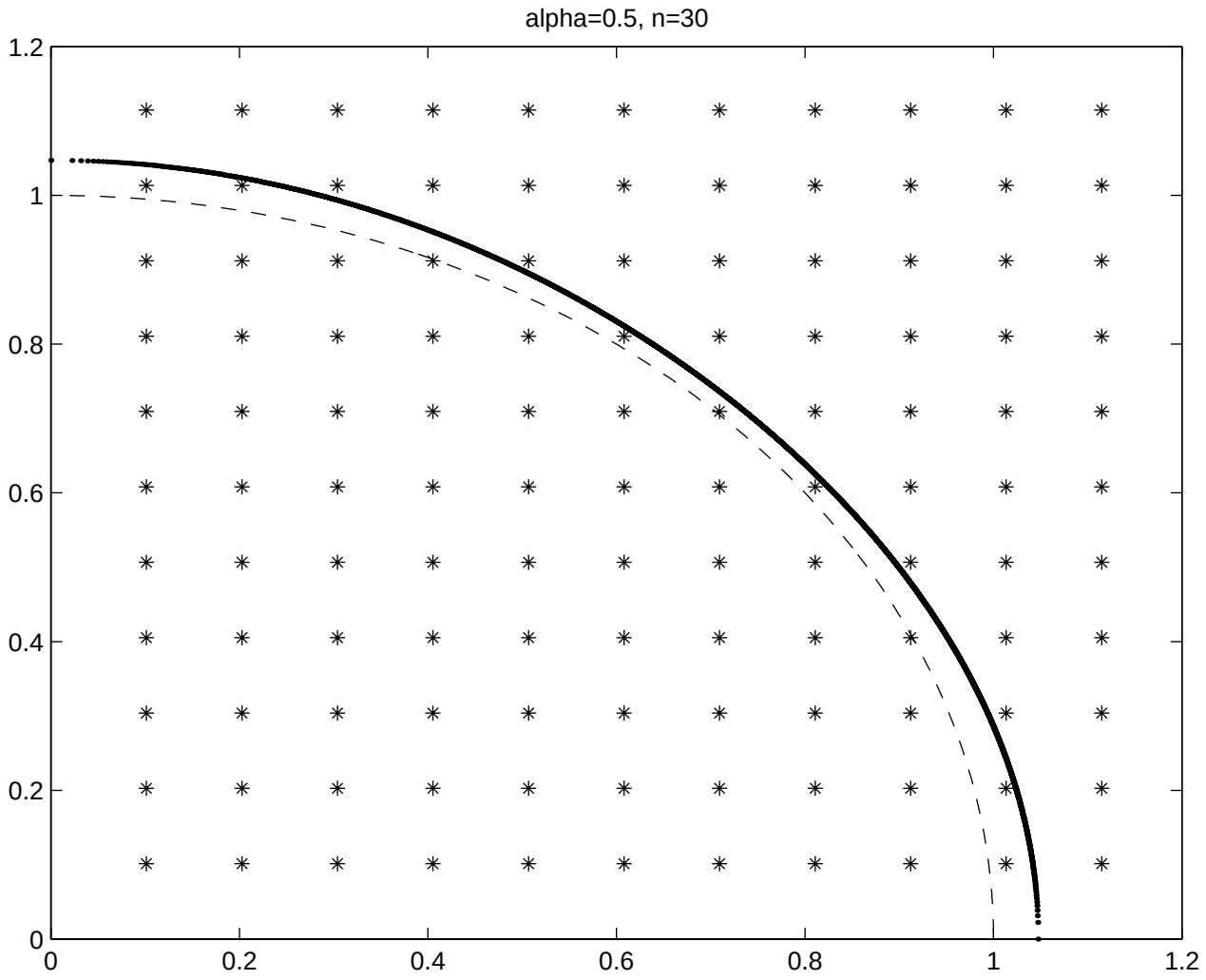

Figure 4.1: Curve $f(x, y) = 0$ and approximate circle

never be expected to have Toeplitz structure. Furthermore, absorbing boundary conditions have to be introduced to guarantee a unique solution; they turn the system complex-symmetric (see e.g. [50] for details). For a state of the art algorithm for multigrid for Helmholtz problems that is also applicable to the non-constant coefficient case we refer to recent work by Elman, Ernst and O'Leary [39], [40].

In Figure 4.1 we display the (j,k)-grid (4.20) with the curve $f(x,y)=0$ and the approximating circle in the (x,y)-plane. Figure 4.2 shows the exact eigenvalues of the matrix on the mesh in the positive (x,y)-quadrant and the curve with $f(x,y)=0$. The mesh also models the surface described by the function f.

4.5.3 Outlook and conclusions

We have investigated multigrid methods for symmetric BTTB matrices. If the matrix is related to a nonnegative function with a single isolated zero $x_0 \in]-\pi,\pi]^2$, then the methods presented in this section are applicable. In particular, the need to use a natural coarse grid operator is even more prominent. However, if the function has a nontrivial curve of zeros then more advanced algorithms, possibly employing more than one coarse grid, need to be developed.

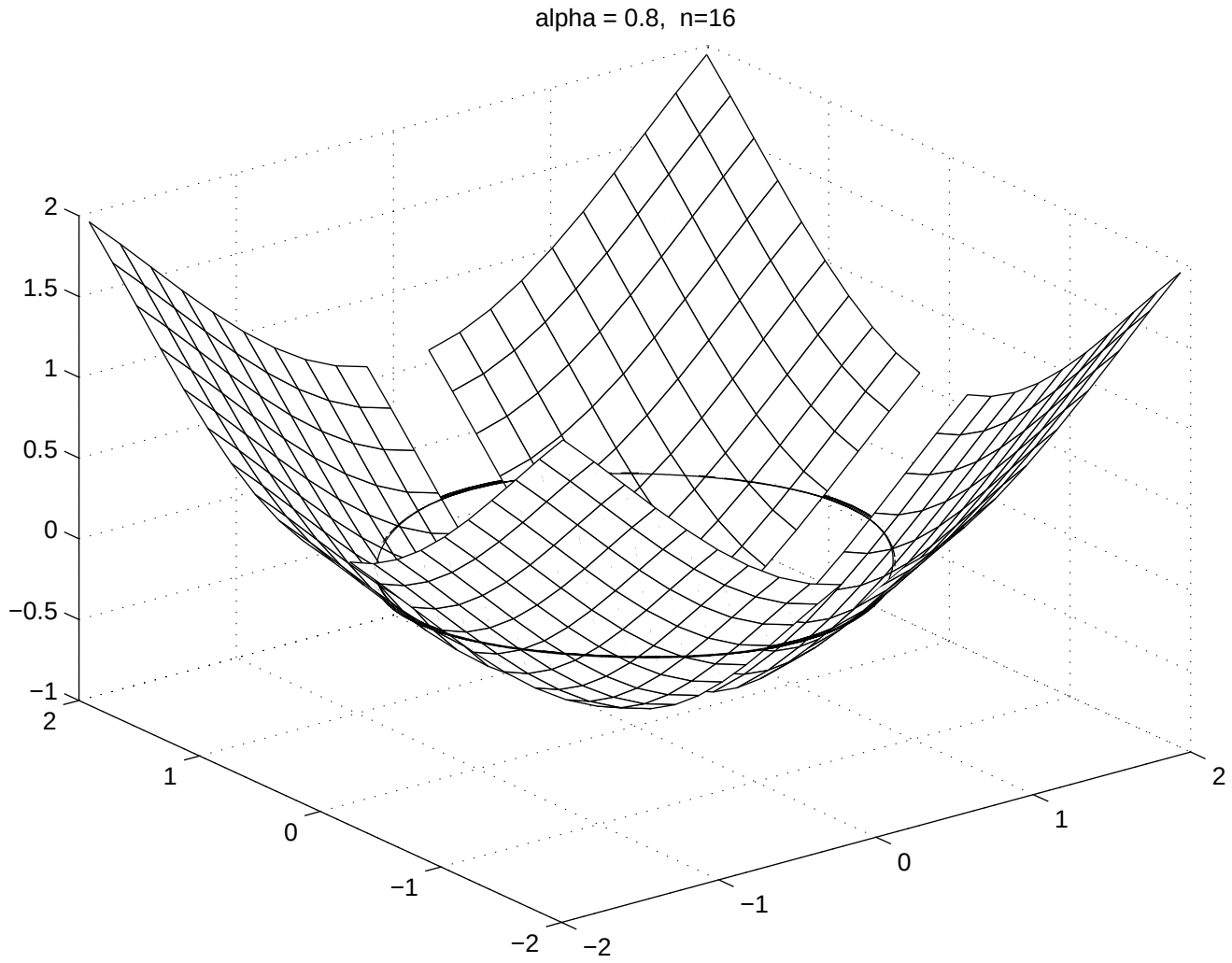

Figure 4.2: Function $f(x, y)$, mesh points as eigenvalues

Chapter 5

The Matrix Multilevel Method

In this chapter we will carry over ideas from the previous chapter for sparse Toeplitz matrices to general sparse banded problems. We will derive new prolongation operators which can be used for general symmetric weakly diagonally dominant matrices. Our new multigrid approach will be called the "Matrix Multilevel Method", abbreviated "MML".

We will first motivate the MML via additive preconditioners for problems in one dimension. Then we shall explain why the MML prolongations can also be used in multiplicative multigrid algorithms and generalize the MML step by step to separable and to general elliptic problems in higher dimensions.

In the last but one section we will come up with a very fast and efficient MML algorithm for general weakly diagonally dominant symmetric linear systems having the same sparsity pattern as a uniform 9-point discretization of the two-dimensional Laplacian. Finally, we will go for a little case study on elliptic problems with highly oscillatory coefficients and compare our MML algorithms to an approach by Engquist and Luo [82], [44] who proposed to employ homogenized coarse grid operators.

Part of the results from this chapter will be published in the paper [72] in the journal BIT. However, our presentation goes beyond the scope of that paper and plenty of results are generalized and extended. On the other hand, our presentation wishes to reflect the chronological order in which we derived certain results and hence we deliberately did very frequently not go for the most straightforward way to motivate an algorithm or a theorem:

In particular, note that the idea of the MML was initially presented by T. Huckle in 1998 in the technical report [71]. However, this first version was still restricted to additive preconditioners for one-dimensional problems. Nevertheless, sections 5.2 and 5.3 could basically be taken over from [71] and the same also refers to most of section 5.4.

In other words: The main **original contributions** of this chapter are

- carrying over the idea of an additive preconditioner to a multiplicative multigrid solver
- the analysis of the one-dimensional case presented in section 5.5
- carrying over the MML idea to higher dimensions – and analyzing and testing it for various examples (section 5.6)
- the case study for elliptic equations with highly oscillatory coefficients presented in section 5.7

However, the author also wishes to acknowledge that the efforts to extend the MML to higher dimensions need to be regarded as joint work with his supervisor – and to thank the latter for having handed him eigenvalue-based MATLAB codes of Algorithms 14 and 15.

5.1 Motivation: From sparse Toeplitz systems to general banded matrices

In Remark 6 we studied the matrices $T_1 = tridiag(-1, 2, -1)$, i.e. the 1D Laplacian from Example 1, and $T_2 = tridiag(1, 2, 1)$ in the last section in the context of sparse Toeplitz matrices: We saw that in a twogrid algorithm the appropriate prolongation operator for T_2 was given by $P_2 = T_1(:, 2 : 2 : n)$ (– in MATLAB-notation with n denoting the size of the linear system –) whereas for T_1 we could simply use the "standard choice" $P_1 = T_2(:, 2 : 2 : n)$.
We have already mentioned that existing multigrid approaches tend to have trouble with matrices having positive off-diagonal entries: Although the celebrated and very general AMG algorithm by Ruge and Stüben [92] could still handle so-called "essentially positive type" matrices, i.e. matrices as they arise from fourth-order discretizations of elliptic problems having relatively small positive off-diagonal entries (see [92], p. 93), AMG might fail for a matrix as simple as T_2.
How can we get a robust algorithm for weakly diagonally dominant matrizes – and not only for M-matrices? Our starting point will be the following observation: From Theorem 2 (ii) we know that $\lambda = 4$ will be a very good upper estimate for the maximum eigenvalue of both T_1 and T_2. Hence with the scheme

$$P_{1,2} = C_{1,2}(:, 2 : 2 : n) \qquad \text{with} \qquad C_{1,2} = \lambda I - T_{1,2}$$

we can derive a suitable prolongation operator for both T_1 and T_2 with one and the same formula. In the following we shall point out that the above concept is sensible and try to carry over this idea to general banded matrices.

5.2 The additive twolevel method

We consider a linear system of equations $Ax = b$ with a sparse ill-conditioned weakly diagonally dominant $n \times n$ matrix A. The aim is to design a purely algebraic multilevel method that can be applied to any matrix in order to reduce the condition number. Here we restrict ourselves to the symmetric positive definite case. We start in an additive setting.

5.2.1 A preconditioner derived via generating systems

Let us first consider only the transfer operators without any smoothing. In the symmetric case our method is based on a mapping C for the restriction and prolongation of the original linear system on a coarser problem. Then we get $C^T AC$, e.g. as an $n/2 \times n/2$ matrix related to the original problem formulated on a coarse grid. Following Griebel [57], [58] and the idea of understanding multigrid algorithms as iterative methods on generating systems we can write the sequence of matrices on different levels also as a sequence of matrix extensions of the form

$$A^{(1)} = A \ , \ \ A^{(2)} = \begin{pmatrix} A & AC \\ C^T A & C^T AC \end{pmatrix} \ = \ \begin{pmatrix} I \\ C^T \end{pmatrix} A \, (\, I \quad C \,) \ . \qquad (5.1)$$

Let us first analyse the relation between the original equation $Ax = b$ and the extended matrix $A^{(2)}$. If $(y^T \ z^T)^T$ is a solution of the extended system

$$\begin{pmatrix} A & AC \\ C^T A & C^T AC \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} \ = \ \begin{pmatrix} b \\ a \end{pmatrix} \ ,$$

we have to set $a = C^T b$, and then $x = y + Cz$ gives us the solution of the original problem $Ax = b$. Furthermore, in view of (5.1), the kernel of $A^{(2)}$ is spanned by the vectors that fulfill $y = -Cz$, and hence the kernel is given by all vectors of the form $\begin{pmatrix} -C \\ I \end{pmatrix} z$. Similarly with (5.1) the range of $A^{(2)}$ is of the form $\begin{pmatrix} I \\ C^T \end{pmatrix} x$.

The convergence of an iterative method applied on the extended linear system depends on the generalized condition number - the quotient of λ_{max} over the smallest nonzero eigenvalue. To describe the nonzero eigenvalues of $A^{(2)}$ we consider the Rayleigh Quotient relative to the space that is orthogonal to the null space $y = \begin{pmatrix} I \\ C^T \end{pmatrix} x$. With

$$\frac{y^T \begin{pmatrix} I \\ C^T \end{pmatrix} A \, (\, I \quad C \,) \, y}{y^T y} \ = \frac{x^T \, (\, I \quad C \,) \begin{pmatrix} I \\ C^T \end{pmatrix} A \, (\, I \quad C \,) \begin{pmatrix} I \\ C^T \end{pmatrix} x}{x^T \, (\, I \quad C \,) \begin{pmatrix} I \\ C^T \end{pmatrix} x} =$$

$$\frac{x^T (I + CC^T) A (I + CC^T) x}{x^T (I + CC^T) x} \ = \ \frac{z^T (I + CC^T)^{1/2} A (I + CC^T)^{1/2} z}{z^T z} \ ,$$

we see that the nonzero part of the spectrum of $A^{(2)}$ is given by the eigenvalues of

$$(I + CC^T)A \ , \tag{5.2}$$

and furthermore the nonzero eigenvalues of $A^{(2)}$ are related to the eigenvalues of A by $\lambda(A^{(2)}) = \lambda(A)(1+\epsilon)$ with $0 \leq \epsilon \leq \lambda_{max}(CC^T)$.

Now we can think of the prolongation C as a preconditioner of the form $I + CC^T$ applied to the original matrix A (see also [9], [70]). A good preconditioner would be one that enlarges only the small eigenvalues of A without changing $\lambda_{max}(A)$. Then the condition number of the preconditioned system would be improved.
Hence the main task is to find a sparse matrix C such that its range contains the span of the eigenvectors associated with the small eigenvalues of the matrix A. Similar problems are considered and solved in [41]; but to obtain the exact solution would be too expensive and can not be used here.

5.2.2 Two special cases

Special case 1: C is an $n \times 1$ matrix of rank 1. In this case the matrix C is reduced to one column vector. Then an optimal preconditioner should enlarge $\lambda_1 = \lambda_{min}$ without changing $\lambda_n = \lambda_{max}$. Based on the eigensystem for A of the form $A = Q^T \Lambda Q$ the problem can be written as follows: Find a vector w such that the matrix $\tilde{\Lambda} = (\, I \quad w\,)^T \, \Lambda \, (\, I \quad w\,)$ has minimum condition number (neglecting the zero eigenvalues.) In view of the interlace property (see e.g. [89]) we get

$$0 = \tilde{\lambda}_1 \leq \lambda_1 \leq \tilde{\lambda}_2 \leq \lambda_2 \quad , \qquad \tilde{\lambda}_n \leq \lambda_n \leq \tilde{\lambda}_{n+1} \ ,$$

and the new condition number is bounded by

$$cond((I + ww^T)\Lambda) = \frac{\tilde{\lambda}_{n+1}}{\tilde{\lambda}_2} \geq \frac{\lambda_n}{\lambda_2} \ .$$

An optimal solution is therefore given by $C = \rho u_{\min}$, where u_{min} is an eigenvector related to the smallest eigenvalue λ_1. Then ρ has to be chosen such that $\lambda_2 \leq (1+\rho^2)\lambda_1 \leq \lambda_n$, and the condition number is improved by a factor λ_1/λ_2.

In general we are interested in prolongations C with larger rank. This is also necessary to lead to a notable improvement of the condition number for

ill-conditioned matrices. Therefore we now write C in the form

$$C \;=\; B\,P \;=\; B(:,J) \tag{5.3}$$

with an $n \times n$ matrix B and an elementary projection P. Now multiplying with P from the right yields a reduction to the columns given by the index set J.

Special case 2: C is an $n \times n$ matrix. To make things easier we first describe the case that $P = I$ and $C = B$. Now we can choose $B = \beta(\alpha I - A)$ with $\alpha = \lambda_{max}(A)$. This matrix has the desired property: λ_{min} becomes large in (5.2) and λ_{max} remains the same. For this special case we can fully analyze the resulting preconditioned system in order to find an optimal value for β.

Let u be any eigenvector of A with length 1 related to an eigenvalue λ, $\lambda_1 \leq \lambda \leq \alpha$. Then β has to be chosen as large as possible with

$$u^T\Big(I + \beta^2(\alpha I - A)(\alpha I - A)^T\Big)Au \;=\; \lambda\Big(1 + \beta^2(\alpha - \lambda)^2\Big) \leq \alpha\ . \tag{5.4}$$

Hence, $\beta^2 \leq \frac{1}{\lambda(\alpha-\lambda)}$. The function on the right hand side takes its minimum value for $\lambda = \alpha/2$, which leads to the optimal value $\beta = 2/\alpha$. The change of the eigenvalues of A under the transformation (5.2) is described by

$$\lambda \;\longrightarrow\; f(\lambda) = \lambda(1 + \frac{4}{\alpha^2}(\alpha - \lambda)^2)\ .$$

In the interval $[\lambda_1, \alpha]$ the function f has a relative maximum at $\lambda = \alpha/2$ of size $f(\alpha/2) = \alpha$, a relative minimum for $\lambda = 5\alpha/6$ with $f(5\alpha/6) = 50\alpha/54$, a global maximum for $\lambda = \alpha$ with $f(\alpha) = \alpha$, and a global minimum for $\lambda = \lambda_1$ with $f(\lambda_1) = \lambda_1(1 + \frac{4}{\alpha^2}(\alpha - \lambda_1)^2) \approx 5\lambda_1$. Hence, by applying $I + BB^T$ as a preconditioner the condition number is approximately improved by a factor of 5. Note that not only the smallest eigenvalue is enlarged, but the whole spectrum is compressed. For example, all eigenvalues of A in the interval $[\alpha/8, \alpha]$ are mapped into the interval $[65\alpha/128, \alpha] \approx [\alpha/2, \alpha]$, and in the interval $[\alpha/3, \alpha]$ into $[25/27\alpha, \alpha]$.

5.2.3 A twolevel preconditioner

Now let us return to the twolevel approach with P and index set J. Then the above relation (5.4) translates into

$$u^T(I+\beta^2(\alpha I-A)PP^T(\alpha I-A)^T)Au \;=\; \lambda(1+\beta^2(\alpha-\lambda)^2\|P^Tu\|^2) \leq \alpha\ . \tag{5.5}$$

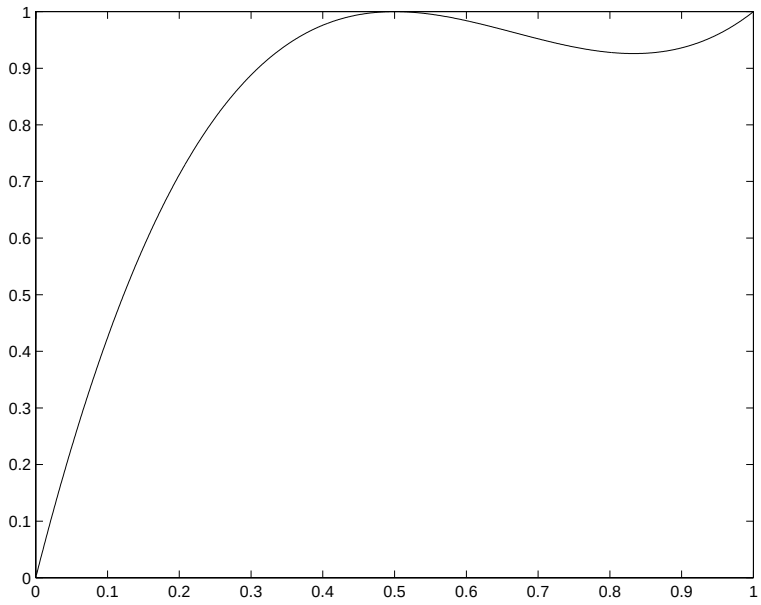

Figure 5.1: Function $f(\lambda)$ for $\alpha = 1$

Therefore P has to be chosen carefully in such a way that for every small eigenvalue $P^T u$ does not become too small. If the eigenvectors are continuous in the sense that they can be seen as values $g(j/n)$ for a continuous function g, then e.g. $P = I(:, 2:2:n)$ gives $\|P^T u\|^2 \approx 1/2$ for all eigenvectors. This leads to an optimal value of $\beta^2 = 8/\alpha^2$ and we can expect that the smallest eigenvalue is approximately improved by a factor of 5; but now the related mapping C has only half the number of entries. Note that for $A = (1/4) * tridiag(-1, 2, -1)$ this optimal factor 8 also appears by diagonal (Jacobi) scaling of the extended system (5.1). In this case the spectrum of $A^{(1)}$ is no longer contained in the interval $[5\lambda_{min}, \alpha]$, but all eigenvalues of A_2 are again smaller than α.

Note that the eigenvalues of $A_2 = C^T A C$ are closely related to the function $g(\lambda) = \beta^2(\alpha - \lambda)^2\lambda$, and therefore the spectrum of A_2 is contained in the interval

$$[\beta^2(\alpha - \lambda_{min})^2\lambda_{min}(A), (\beta\alpha)^2(4/27)\alpha] \ .$$

For $P = I(:, 2:2:n)$, an eigenvector u corresponding to a small eigenvalue of A leads to a small value of the Rayleigh Quotient related to the matrix A_2

and the vector $P^T u$. Therefore we may expect that $P^T u$ is mainly contained in a subspace spanned by eigenvectors of A_2 that belong to small eigenvalues.

We finally remark that we will not have the optimal factor β in any practical algorithm. We shall see in the following section how this role will more or less be taken over by diagonal scalings.

5.3 The Matrix Multilevel Method: Additive and multiplicative variants

The analysis of the previous section describes an additive twolevel method where the smoothing is reduced to a scaling factor. Now we have to generalize the approach in order to make it work in a multilevel fashion.

5.3.1 Going multilevel

So far we have arrived at the representation $A = A_1 = A^{(1)}$, $A_2 = C_1^T A_1 C_1$,

$$A^{(2)} \;=\; \begin{pmatrix} A_1 & A_1 C_1 \\ C_1^T A_1 & C_1^T A_1 C_1 \end{pmatrix} \;=\; \begin{pmatrix} I & C_1 \end{pmatrix}^T A_1 \begin{pmatrix} I & C_1 \end{pmatrix} \;,$$

or in preconditioned form

$$(I + C_1 C_1^T) A_1 \;.$$

Let us assume that the prolongation C_1 is chosen properly such that its range contains the eigenvectors associated with the small eigenvalues of A. Then, on the next level we can restrict ourselves to operators of the form $C = C_1 C_2$. Now we can apply (5.2) a second time and arrive at a preconditioner

$$(I+C_1C_2C_2^TC_1^T)(I+C_1C_1^T)A_1 \;=\; (I+C_1C_1^T+C_1C_2C_2^TC_1^T+C_1C_2C_2^TC_1^TC_1C_1^T)A_1 \;.$$

To make the preconditioner symmetric positive definite we delete the last nonsymmetric term and use only

$$I + C_1C_1^T + C_1C_2C_2^TC_1^T \;=\; I + C_1(I + C_2C_2^T)C_1^T \;.$$

Then we have different formulas for the extended system:

$$A^{(3)} = \begin{pmatrix} A & AC_1 & AC_1C_2 \\ C_1^T A & C_1^T AC_1 & C_1^T AC_1C_2 \\ C_2^T C_1^T A & C_2^T C_1^T AC_1 & C_2^T C_1^T AC_1C_2 \end{pmatrix} = \begin{pmatrix} I \\ C_1^T \\ C_2^T C_1^T \end{pmatrix} A \begin{pmatrix} I & C_1 & C_1C_2 \end{pmatrix}$$

$$= (\, I \quad C_1\,(\, I \quad C_2\,)\,)^T\, A\,(\, I \quad C_1\,(\, I \quad C_2\,)\,) =$$

$$= \begin{pmatrix} I & 0 & 0 \\ 0 & I & C_2 \end{pmatrix}^T \begin{pmatrix} A & AC_1 \\ C_1^T A & C_1^T A C_1 \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & I & C_2 \end{pmatrix} =$$

$$= \begin{pmatrix} I & 0 & 0 \\ 0 & I & C_2 \end{pmatrix}^T (\, I \quad C_1\,)^T\, A\,(\, I \quad C_1\,) \begin{pmatrix} I & 0 & 0 \\ 0 & I & C_2 \end{pmatrix} ,$$

and in preconditioned form

$$(I + C_1 C_1^T + C_1 C_2 C_2^T C_1^T) A = (I + C_1 (I + C_2 C_2^T) C_1^T) A \tag{5.6}$$

or

$$\begin{pmatrix} I & 0 \\ 0 & I + C_2 C_2^T \end{pmatrix} \begin{pmatrix} A & AC_1 \\ C_1^T A & A_2 \end{pmatrix} = \begin{pmatrix} A & AC_1 \\ (I + C_2 C_2^T) C_1^T A & (I + C_2 C_2^T) A_2 \end{pmatrix} .$$

This leads to different heuristics for choosing C_2. In view of the above equation we can think of C_2 as a second preconditioning step related to $A_2 = C_1^T A C_1$ and therefore we can set

$$C_2 = \beta_2 (\alpha_2 I - A_2)\, P_2 \; . \tag{5.7}$$

We can also derive (5.7) based on another approach. The new prolongation C_2 defines the preconditioned system

$$(I + C_1 C_1^T + C_1 C_2 C_2^T C_1^T) A$$

and thus in the sense of (5.4) and (5.5) we get

$$u^T C_1 C_2 C_2^T C_1^T u = \beta_1^2 u^T (\alpha I - A) P C_2 C_2^T P^T (\alpha I - A)^T u$$

$$= \beta_1^2 (\alpha - \lambda_{min})^2 (u_{min}^T P) C_2 C_2^T (P^T u_{min}) =$$

$$= \beta_1^2 (\alpha - \lambda_{min})^2 (u_{min}^T P_1) B_2 P_2 P_2^T B_2 (P_1^T u_{min}) \; .$$

Now B_2 should be chosen in such a way that it gets large for the vectors $P_1^T u$ related to small eigenvalues of A. In view of the previous remark at the end of Section 1 we can expect that the vectors $P_1^T u$ are related to small eigenvalues of A_2 which again suggests to define C_2 via (5.7).
We could also formulate another way for choosing C_2. Note that the eigenvalues of A_2 are closely related to the function $g(\lambda) = \beta^2 (\alpha - \lambda)^2 \lambda$. This shows that the large eigenvalues of A are also translated into very small eigenvalues

of A_2. If we define C_2 with (5.7), then in this second step we try to enlarge these originally large eigenvalues together with the small eigenvalues of A. This suggests another way to define C_2, namely again as a projection of the first-level matrix $\alpha I - A$. For example, if A is a Toeplitz matrix like in the previous section then we can consider the Toeplitz matrix $\alpha I - A = T$ and choose C_j as a submatrix $T(1 : 2^l, 2 : 2 : 2^l)$.
In order to obtain similar improvements on the condition number on every level it is necessary that all the derived smaller systems have similar properties as the original matrix A. For example, if A_2 is well conditioned then obviously going to a coarser grid will lead to no improvement of the spectrum. Hence we have to choose C and P in such a way that the matrix $\hat{A} = P^T B^T ABP$ inherits important properties of A. In many cases the behaviour of A on the vector $e_n = (1, ..., 1)^T$ is very important - this is related to the property that the rowsum of entries is often zero. Hence we may ask that $e_{n/2}^T \hat{A} e_{n/2} = e_n(J)^T B^T AB e_n(J) \approx e_n^T A e_n/2$. We obtain this property by choosing B such that $Be_n(J) = e_{n/2}/\sqrt{2}$. For $B = \sqrt{2} * tridiag(1/4, 1/2, 1/4)$ and $J = (2, 4, 6, ..., n)$ (– the usual multigrid prolongation –) this is obviously fulfilled. After diagonal scaling this is also nearly satisfied for the mapping $B = \lambda_{max} I - A$ in many cases.

5.3.2 Including a smoother

Now we have defined a multilevel method based only on the original matrix A and the maximum eigenvalues of the resulting systems A_j. It is necessary to include also some kind of smoothing operation on every level in order to get fast convergence. Here we will mainly consider the Jacobi method for smoothing. In (5.1) or (5.6) the Jacobi smoothing is nothing else than diagonal preconditioning. Note that in the same way one can use Gauss-Seidel or any other levelwise method.

To include Jacobi smoothing on every level let us consider the enlarged problem

$$A^{(k)} = \begin{pmatrix} A & AC_1 & AC_1C_2 & \cdots & AC_1..C_k \\ C_1^T A & C_1^T AC_1 & C_1^T AC_1C_2 & \cdots & C_1^T AC_1..C_k \\ \vdots & & \cdot & & \vdots \\ \vdots & & & \cdot & \vdots \\ C_k^T..C_1^T A & \cdots & \cdots & & C_k^T..C_1^T AC_1..C_k \end{pmatrix} =$$

$$= (\, I \quad C_1 \quad C_1C_2 \quad \cdots \quad C_1..C_k \,)^T \, A \, (\, I \quad C_1 \quad C_1C_2 \quad \cdots \quad C_1..C_k \,) =$$

$$= (\, I \quad C_1(I \,\cdots\, (I\, C_k)\cdots)))^T\, A\, (\, I \quad C_1(I \,\cdots\, (I\, C_k)\cdots)))\ \ (5.8)$$

and in preconditioned form

$$\begin{aligned}(I + C_1C_1^T + C_1C_2C_2^TC_1^T + \,\cdots\, + C_1..C_kC_k^T..C_1^T)A = \\ (I + C_1(I + C_2(I + \,\cdots\, (I + C_kC_k^T)\cdots)C_2^T)C_1^T)A = M^{(k)}A\ . \qquad (5.9)\end{aligned}$$

In the form (5.8) we can comprise any preconditioner on the matrix $A^{(k)}$, for example Jacobi, Gauss-Seidel or ILU preconditioners, and employ the conjugate gradient method with zero starting vector. But usually we want to compute only the small matrices A_j and the prolongations C_j on every level, but neither the whole system $A^{(k)}$ nor the - nearly dense - preconditioner $M^{(k)}$. Therefore, we will use only levelwise block-diagonal preconditioners based on the level matrices A_j.
From (5.8) we can translate preconditioners very easily to the form (5.9). In the Jacobi case for example we have

$$D_j = diag(A_j) = diag(C_j^T \cdots C_1^T A C_1 \cdots C_j)$$

and for every matrix A_j we can use $D_j^{-1/2}$ as left and right preconditioner. Then, with

$$D = diag(D_1^{-1/2}, \cdots, D_k^{-1/2})\ ,$$

(5.8) translates into

$$D \begin{pmatrix} A & AC_1 & AC_1C_2 & \cdots & AC_1..C_k \\ C_1^TA & C_1^TAC_1 & C_1^TAC_1C_2 & \cdots & C_1^TAC_1..C_k \\ \vdots & & \cdot & & \vdots \\ \vdots & & & \cdot & \vdots \\ C_k^T..C_1^TA & \cdots & \cdots & & C_k^T..C_1^TAC_1..C_k \end{pmatrix} D \ =$$

$$\left(\, I \quad D_1^{1/2}C_1D_2^{-1/2} \quad D_2^{1/2}C_2D_3^{-1/2} \quad \cdot\right)^T D_1^{-1/2}AD_1^{-1/2} \left(\, I \quad D_1^{1/2}C_1D_2^{-1/2} \quad \cdot\right).$$

Hence, we only have to replace A by $\tilde{A} = D_1^{-1/2}AD_1^{-1/2}$, and each C_j by $\tilde{C}_j = D_j^{1/2}C_jD_{j+1}^{-1/2}$. This leads to the new preconditioned form

$$(I + \tilde{C}_1(I + \tilde{C}_2(I + \,\cdots(I + \tilde{C}_k\tilde{C}_k^T)\cdots)\tilde{C}_2^T)\tilde{C}_1^T)\tilde{A}\ . \qquad (5.10)$$

5.3.3 Improving the preconditioner

It turns out that we can improve the preconditioner (5.10) a lot by using the diagonally scaled matrix $\tilde{A}_j$ on every level for constructing the next prolongation/restriction: This means that if we have arrived at the matrix A_j we should use the (already Jacobi scaled) matrix $\tilde{A}_j$ for the next step and therefore define the next prolongation via $B = \lambda I - \tilde{A}$ and apply it to the diagonally scaled $A^{(j)}$ like in (5.8).

It is possible to include the Jacobi scaling directly in equation (5.10) in the form

$$(D_1^{-1} + \tilde{C}_1(D_2^{-1} + \tilde{C}_2(D_3^{-1} + \ \cdots (D_k^{-1} + \tilde{C}_k\tilde{C}_k^T)\cdots)\tilde{C}_2^T)\tilde{C}_1^T)A \qquad (5.11)$$

The equation (5.11) leads to a first Matrix Multilevel Algorithm:

Algorithm 9 (Additive MML with eigenvalues in 1D)
We start with the scaled matrix $\tilde{A}_1$
- On every level compute $B_j = \alpha I - \tilde{A}_j$ *and use* $C_j = B_j(:, 2:2:n)$ *for prolongation and restriction.*
- After computation of $A_{j+1} = C_j^T \tilde{A}_j C_j$ *we step forward with the scaled matrix* $\tilde{A}_{j+1}$.
- Based on the matrices C_j *and* $D_{j+1} = diag(C_j^T \tilde{A}_j C_j)$, $j = 0, ..., l$, *we can recursively implement the multiplication of the matrix (5.11) with a given vector and thus use the preconditioned CG method.*

Similiarly to (5.10), we can apply any levelwise direct preconditioner M_j on the matrix $\tilde{A}_j$ and get

$$(M_1 + \tilde{C}_1(M_2 + \tilde{C}_2(M_3 + \cdots (M_k + \tilde{C}_k\tilde{C}_k^T)\cdots)\tilde{C}_2^T)\tilde{C}_1^T)\tilde{A}\ .$$

Furthermore we can include, e.g. two smoothing steps on every level by replacing M_k by $2M_k - M_k A_k M_k$ in order to imitate a two-fold application of the preconditioner M_k. If M_k is not symmetric, we can use instead the symmetrized matrix $M_k + M_k^T - M_k A_k M_k^T$.
For the Jacobi or Gauss-Seidel iteration we often introduce damping factors. The same is possible here if we replace the preconditioner in (5.10) by

$$(I + \omega_1\tilde{C}_1(I + \omega_2\tilde{C}_2(I + ...)\tilde{C}_2^T)\tilde{C}_1^T) = (I + \omega_1\tilde{C}_1\tilde{C}_1^T + \omega_2\tilde{C}_1\tilde{C}_2\tilde{C}_2^T\tilde{C}_1^T + ...)\ .$$

In view of (5.1) and the analysis of Section 5.1 a factor $\omega < 1$ may be necessary to reduce the maximum eigenvalue to be $\leq \alpha$. A factor $\omega > 1$

can be helpful for faster convergence if it is possible to enlarge the small eigenvalues without changing λ_{max}.

5.3.4 Towards multiplicative multilevel algorithms

So far, we have used generating systems in order to devise new prolongations and restrictions and we have presented our approach in a purely additive setting: Our algorithms are of the same type as BPX [9] or MDS [120]. From the work by Griebel [57], [58] we know that the MDS-preconditioner is nothing but the Block Jacobi method applied on the generating system and that multiplicative multigrid algorithms can simply be viewed as iterative methods of Gauss-Seidel type on the generating system. Furthermore, in their abstract convergence framework Griebel and Oswald [59] point out why optimal order preconditioning via the additive multilevel algorithms implies optimal order convergence of corresponding multiplicative multilevel algorithms. Their theoretical reasoning justifies using the new prolongations/restrictions also in multiplicative variants. In the following we very briefly describe how the ideas of Algorithm 9 can be used in a multigrid V-cycle.

Algorithm 10 (Multiplicative MML with eigenvalues in 1D)
We start with the scaled matrix $\tilde{A}_1$.
- On every level use $B_j = \alpha I - \tilde{A}_j$ *and* $C_j = B_j(:, 2:2:n)$ *for prolongation and restriction.*
- After computation of $A_{j+1} = C_j^T \tilde{A}_j C_j$ *we step forward with the scaled matrix* $\tilde{A}_{j+1}$ *as the coarse grid matrix*
- When going down we restrict the residuals using $R_j = D_{j+1}^{-1/2} C_j^T$ *and when going up we use* R_j^T *for prolongation.*

Now we can either employ a multigrid V-cycle as a solver or as a preconditioner within the CG method (by performing one V-cycle with the residual r and initial guess zero). As we already remarked in Section 2.5, for complex problems in Scientific Computing, we would in general recommend to use V-cycles as preconditioners for CG guarantees convergence. Nevertheless we shall in the following numerical tests mainly employ our V-cycles as solvers, because that way it will become much more clearly visible if one has picked prolongations/ restrictions suitable for the problem at hand – or not.

5.4 Algorithmic issues and numerical tests

We will now point how to use Algorithms 9 and 10 in practice and report test results for several examples.

5.4.1 General setting and example problems

For practical implementations of the one-dimensional MML we will consider different variations:

- We can use the Matrix Multilevel transfer operators in Algorithms 9 or 10 and compare them with the standard transfer operators.
- We can need to derive estimates for $\alpha = \lambda_{max}(A_j)$ on every level as it is not at all practical to compute the exact eigenvalue: Different choices will be discussed in subsection 5.4.3.
- For the additive variants of the MML we apply the conjugate gradient method with preconditioner of the form (5.11) and zero starting vector.
- We also use V-cycles with one presmoothing and one postsmoothing. step as solvers; our smoother within these cycles is the symmetric Gauss-Seidel method and we start with zero initial guess.
- As stopping criterion we employ

$$\|r^{(k)}\|_2/\|r^{(0)}\|_2 \leq rex \tag{5.12}$$

  in the whole chapter: In (5.12) rex is a small positive constant and $r^{(k)}$ denotes the residual after k iterations.
- In our tables n will always denote the number of unknowns.

Example 7 (One-dimensional elliptic test problems)
As numerical examples we consider uniform finite difference discretizations of the 1D elliptic PDE

$$(a(x)u(x)_x)_x = f(x) \tag{5.13}$$

with homogenous Dirichlet boundary conditions and right hand side $f(x) \equiv 1$ on the unit interval for

(EX1) $a(x)$ *constant,*
(EX2) $a(x) = 1 + sin(32\pi x)^2$,
(EX3) $a(x) = 1 + exp(2\pi x)sin(2\pi x)^2$,
(EX4) $a(x) = 1 + exp(\pi x)sin(8\pi x)^2$,
(EX5) $a(x) = 1 + exp(2\pi x)sin(8\pi x)^2$,
(EX6) $a(x) = 1 + exp(8\pi x)sin(8\pi x)^2$,
(EX7) $a(x)$ *is piecewise constant with ten different values in* $[0.1, 2.1]$.
(EX8) $a(x)$ *is piecewise constant with ten different values in* $[0.1, 1.5E + 4]$.

Note that (EX1) is nothing but the 1D Laplacian (2.1).

5.4.2 Comparing condition numbers

In this subsection we can still afford always to choose α as the exact eigenvalue of A:
The next tables display the condition numbers for the additive Matrix Multilevel Method based on Algorithm 9 and computing α_j on every level (MML), only on the finest level (MML0), for the standard MDS-method (MDS), and for the Jacobi-preconditioned original problem.

n	MML	MML0	MDS	$D^{-1}A$
2^5	4.55	6.32	4.60	414.3
2^6	5.43	7.26	5.12	1.7E3
2^7	6.34	8.19	5.62	6.6E3
2^8	7.26	9.14	6.11	2.7E4

Table 5.1. Condition number example (EX1), $a(x) = const.$

n	MML	MML0	MDS	$D^{-1}A$
2^5	4.55	6.32	4.60	414
2^6	5.191	7.98	6.30	1.9E3
2^7	6.12	7.93	98.8	7.1E3
2^8	7.15	9.00	80.6	2.8E4

Table 5.2. Condition number example (EX2), $a(x) = 1 + sin(32\pi x)^2$.

n	MML	MML0	MDS	$D^{-1}A$
2^5	4.37	6.08	36.4	605.4
2^6	5.37	7.16	32.0	2.2E3
2^7	6.32	8.16	29.9	8.8E3
2^8	7.26	9.13	31.2	3.6E4

Table 5.3. Condition number example (EX4), $a(x) = 1 + exp(\pi x)sin(8\pi x)^2$.

n	MML	MML0	MDS	$D^{-1}A$
2^5	4.37	6.10	250.5	2.6E3
2^6	5.37	7.17	149.4	6.2E3
2^7	6.32	8.18	97.5	1.7E4
2^8	7.26	9.14	86.3	6.2E4

Table 5.4. Condition number example (EX5), $a(x) = 1+exp(2\pi x)sin(8\pi x)^2$.

n	MML	MML0	MDS	$D^{-1}A$
2^5	4.25	5.94	8.5E8	6.3E9
2^6	5.47	7.17	5.7E8	1.3E10
2^7	6.42	8.23	3.7E8	2.7E10
2^8	7.32	9.19	2.2E8	5.5E10

Table 5.5 Condition number example (EX6), $a(x) = 1 + exp(8\pi x)sin(8\pi x)^2$.

For the above examples we get nearly the same condition numbers for the two different additive matrix multilevel preconditioners. We see that the MML is very robust with respect to the function $a(x)$ - whereas the standard MDS preconditioner is not - and the condition numbers only grow slightly with the problem size n.

5.4.3 Estimating the largest eigenvalue

Obviously, it is far too expensive to use the exact maximum eigenvalue $\alpha = \lambda_{max}$ on every level.

Hence our idea is to get an estimate for α by running the **Lanczos method** (see e.g. [54], ch. 2) up to $|m|$ steps and

setting $\alpha = \tilde{\lambda}_{max}$ (denoted by $m > 0$ in our tables) or
setting $\alpha = \tilde{\lambda}_{max} + \tilde{\lambda}_{min}$ (denoted by $m < 0$ in our tables).

There is even a more "lazy" way to get a reasonable estimate for α. Remember that we only work with diagonally scaled matrices on every level, i.e. all our matrices $\tilde{A}_j$ have ones on their diagonals. Furthermore we assumed that A was a symmetric weakly diagonally dominant matrix. Hence Gershgorin's theorem (see Axelsson [2], pp. 127) tells us that $\alpha = 2$ is an upper estimate for the maximum eigenvalue λ_{max}:

Theorem 7 (Gershgorin circles)
The spectrum $\mathcal{S}(A)$ of the matrix $A \in \mathbb{C}^{n\times n}$ is enclosed in the union of the discs

$$C_j = \{z \in \mathbb{C} : |z - a_{jj}| \leq \sum_{k\neq j} |a_{jk}|\}, \qquad 1 \leq j \leq n,$$

and in the union of the discs

$$C_j^{'} = \{z \in \mathbb{C} : |z - a_{jj}| \leq \sum_{k\neq j} |a_{kj}|\}, \qquad 1 \leq j \leq n.$$

That is,

$$\mathcal{S}(A) \subset (\cup_{j=1}^{n} C_j) \cap (\cup_{j=1}^{n} C_j^{'})$$

As we know that $\alpha = 2$ is indeed an upper bound, we will assume that the corresponding transfer operators preserve positive definiteness and that all the coarse level matrices are weakly diagonally dominant as well. (This may seem obvious; anyway, we shall analyze and prove it formally in subsection 5.5.2) Hence we shall try to use the estimate $\alpha = 2$ on every level. This will be denoted by $\lambda_{est} = 2$ in our tables.

In the next table we compare the influence of the choice of α for the convergence of the Matrix Multilevel Method. It turns out that as well with the simple choice $\lambda_{est} = 2$ as with only one or two iterations of the Lanczos process we get eigenvalue estimates that lead to the same iteration numbers in the Matrix Multilevel Method as using the exact maximum eigenvalue λ_{max}:

n	MDS	λ_{max}	$\lambda_{est} = 2$	m=2	m=3	m=4	m=-1	m=-2	m=-3
2^6	37	10	10	10	10	10	11	11	11
2^7	45	12	12	12	12	12	12	12	12
2^8	53	14	14	15	15	14	14	14	14
2^9	61	16	17	28	21	17	17	17	17
2^{10}	72	18	18	42	34	28	18	18	18
2^{11}	80	*	20	61	49	43	20	20	20
2^{12}	98	*	22	92	73	60	23	22	22

Table 5.6. CG iteration numbers for additive preconditioners for example (EX6) with $rex = 10^{-4}$. The final eight columns compare the different choices for α within the MML. (A $*$ denotes that the computation would have been too expensive.)

All our tests confirm that the "lazy" estimate $\lambda_{est} = 2$ is indeed sensible – and, in general, we would recommend to use it.
However, we might equally well employ Lanczos: In that case we recommend to set $\alpha = \tilde{\lambda}_{max} + \tilde{\lambda}_{min}$ where $\tilde{\lambda}_{max}$ and $\tilde{\lambda}_{min}$ are estimates for largest and smallest eigenvalue of A. Table 5.6 shows clearly that very few Lanczos steps are sufficient in order to compute these estimates. Note that Lanczos yields inner approximations to the extremal eigenvalues and that α should be greater or equal λ_{max}, because otherwise the projected matrix can get indefinite. Note that the costs for estimating α with the Lanczos method correspond roughly to the costs for one Matrix Multilevel iteration step – and hence the choice to use Lanczos is not quite expensive, either.
We finally remark that sometimes the projection with $\lambda_{est} = 2$ gives better results and sometimes the projections based on estimating the maximum eigenvalue via Lanczos does better.

5.4.4 Numerical results for 1D problems

The following four tables clearly demonstrate the superiority of the new MML transfer operators in comparison to standard prolongations/restrictions – the first two deal with additive preconditioners, the following two with V-cycle solvers. Other than the standard approach the MML approach is very robust with respect to the function $a(x)$ and we observe optimal computational behaviour in many cases:

n	MDS	MML: Alg. 9	$\lambda_{est} = 2$	$m = -2$	$m = -3$
2^7	32		13	13	13
2^8	43		14	14	14
2^9	51		16	16	16
2^{10}	57		17	17	17
2^{11}	66		18	18	18
2^{12}	80		19	19	19

Table 5.7. CG iteration numbers for example (EX7), $rex = 10^{-4}$.

n	MDS	MML: Alg. 9	$\lambda_{est} = 2$	$m = -2$	$m = -3$
2^7	60		16	17	16
2^8	81		18	20	18
2^9	97		19	21	20
2^{10}	110		20	23	22
2^{11}	125		22	27	25
2^{12}	167		23	31	28

Table 5.8. CG iteration numbers for example (EX8), $rex = 10^{-4}$.

n	Standard MG	MML: Alg. 10	$\lambda_{est} = 2$	$m = -2$	$m = -3$
2^7	646		6	6	6
2^8	1381		6	6	6
2^9	> 2000		7	6	6
2^{10}	> 2000		7	6	6
2^{11}	> 2000		7	6	5
2^{12}	> 2000		7	7	5

Table 5.9. Numbers of V-cycle iterations for example (EX6), $rex = 10^{-6}$.

n	Standard MG	MML: Alg. 10	$\lambda_{est} = 2$	$m = -2$	$m = -3$
2^7	641		7	7	7
2^8	1223		7	7	7
2^9	> 2000		8	8	7
2^{10}	> 2000		8	6	6
2^{11}	> 2000		8	6	6
2^{12}	> 2000		7	6	6

Table 5.10. Numbers of V-cycle iterations for example (EX7), $rex = 10^{-6}$.

Let us finally take up our introductory example $T_2 = tridiag(1, 2, 1)$ from Section 5.1 again and show that the Matrix Multilevel approach can indeed be applied successfully for weakly diagonally dominant problems which are not M-matrices.

n	MDS	MML: Alg. 9	$\lambda_{est} = 2$	$m = -1$	$m = -2$	$m = -3$
2^6	53		7	7	7	7
2^7	113		7	7	7	7
2^8	247		7	7	7	7
2^9	330		7	7	7	7

Table 5.11. CG iteration numbers for $T_2 = tridiag(1, 2, 1)$, $rex = 10^{-4}$.

5.5 Analysis of the MML – and a new variant

Let us take a closer look at the choice $\alpha = \lambda_{max} = 2$ in Algorithms 9 and 10 and investigate why it is in fact sensible.

5.5.1 MML prolongations with $abs(A)$ for M-matrices

Let us assume A is a tridiagonal M-matrix, e.g. one of the problems (EX1)-(EX8) from Example 7. Then we see that for the diagonally scaled version $\tilde{A}$ of A the prolongation operator $B = 2 * I - \tilde{A}$ could also be expressed as $B = abs(\tilde{A})$.

Following Hackbusch [62], pp. 212, we see that there is another way to derive transfer operators of the form $B = abs(A)$: Let us choose

$$a(x) = \begin{cases} a^-, & 0 < x < \xi \\ a^+, & \xi < x < 1 \end{cases} \tag{5.14}$$

in our 1D elliptic model equation (5.13). For this interface problem a uniform second-order finite difference discretization on the level l is given in the form $A_l u_l = f_l$ with

$$A_l = h_l^{-2}[-a(x - \frac{h_l}{2}) \quad a(x - \frac{h_l}{2}) + a(x + \frac{h_l}{2}) \quad - a(x + \frac{h_l}{2})]$$

in stencil notation. Certainly, ξ should be a grid point – for otherwise the discretization error might become huge – and we obtain

$$A_l = \begin{cases} h_l^{-2} a^- [-1 \quad 2 \quad -1] & \text{for} \quad 0 < x < \xi \\ h_l^{-2} [-a^- \quad a^- + a^+ \quad a^+] & \text{for} \quad x = \xi \\ h_l^{-2} a^+ [-1 \quad 2 \quad -1] & \text{for} \quad \xi < x < 1 \end{cases} \tag{5.15}$$

Now let us assume that ξ is not a grid point on the next coarser level $l-1$. With v_{l-1} being a vector on that coarse level, we can compute an appropriate prolongation operator p using the homogeneous difference equation

$$(A_l p v_{l-1})(\xi) = 0. \tag{5.16}$$

From (5.16) we get

$$(p v_{l-1})(\xi) = \frac{a^-}{a^- + a^+} v_{l-1}(\xi - h_l) + \frac{a^+}{a^- + a^+} v_{l-1}(\xi + h_l) \tag{5.17}$$

at the point ξ whereas we can apply standard linear interpolation for the rest of the grid points: If we write this prolongation in stencil notation it reads as

$$p = [p_{-1}(x) \quad p_0(x) \quad p_1(x)]$$

with $p_0(x) = 1$ and

$$p_{-1}(x) = \begin{cases} a^+/(a^- + a^+) & \text{if } x = \xi + h_l \\ 0.5 & \text{otherwise} \end{cases}$$

and

$$p_1(x) = \begin{cases} a^-/(a^- + a^+) & \text{if } x = \xi - h_l \\ 0.5 & \text{otherwise} \end{cases}$$

Of course, the corresponding restriction operator for a multigrid treatment of (5.14) is given by the transpose of the prolongation we computed.

Now we have derived in detail a suitable prolongation operator p for the problem (5.14) using the homogeneous difference equation (5.16) which we could also express in MML terms as $C = B(:, 2:2:n)$ with $B = abs(A_l)$.

From the previous analysis we understand why $\alpha = \lambda_{est} = 2$ gives an appropriate choice within Algorithms 9 and 10 in case A is an M-matrix. But the consequences of the above reasoning go much further:

First of all, we would like to emphasize very strongly that the idea underlying the homogeneous difference equation (5.16) is that the interface problem (5.14) has a continuous solution with a discontinuous derivate, i.e. the prolongation based on $abs(A)$ takes into account the non-smooth behaviour of the solution. For more details on "continuity interpolation" we refer to [1], [77], [56], ch. 5, or [62], pp. 212.
Obviously, from an analytic point of view is no "better choice" for a prolongation operator to be used for our problem (5.14) than the one satisfying (5.16). If we base the Matrix Multilevel method on eigenvalue estimates this also explains why we should use scaled matrices on every level: If we do not scale our matrix, then the prolongation operator based on $B = \alpha I - A$ with $\alpha = \lambda_{max}$ might no longer be close to the one satisfying (5.16). But if we work with the scaled matrix $\tilde{A}$ and use $\alpha = \lambda_{est} = 2$ then on the basis of $B = \alpha I - \tilde{A}$ we get a prolongation matrix matching (5.16). Note that this also provides a new interesting link between the largest eigenvalue, diagonal scalings and transfer operators based on homogeneous difference equations in multigrid algorithms.

Secondly, we see that for symmetric M-matrices we can derive a new MML algorithm based on $B = abs(A)$ which use the formulation (5.10), i.e. we can this time build a preconditioner without scaling our matrices on each individual level:

Algorithm 11 (Additive MML with $abs(A)$ and no scalings in 1D)
We start with the **M-matrix** $A = A_1$.
- On every level use $B_j = abs(A_j)$ *and* $C_j = B_j(:, 2:2:n)$ *for prolongation and restriction.*
- After computation of $A_{j+1} = C_j^T A_j C_j$ *save the diagonal entries in* D_{j+1}.
- Based on the matrices C_j *and* D_j, $j = 0, ..., l$, *we can recursively implement the multiplication of the matrix (5.10) with a given vector and thus use the preconditioned CG method.*

The multiplicative version of Algorithm 11 is straightforward: It can be viewed as an analogue of Algorithm 10 without individual scalings on every level and applicable only to M-matrices.

Algorithm 12 (Mult. MML with $abs(A)$ and no scalings in 1D)
We start with the **M-matrix** $A = A_1$.
- On every level use $B_j = abs(A_j)$ *and* $C_j = B_j(:, 2:2:n)$ *for prolongation*

and restriction.
- Compute $A_{j+1} = C_j^T A_j C_j$ to set up the coarse grid matrix
- When going down we restrict the residuals using $R_j = C_j^T$ and when going up we use R_j^T for prolongation.

5.5.2 Properties of the new MML transfer operators

In 5.4.3 we said that we dare to use the estimate $\alpha = \lambda_{max} = 2$ on every level, because we assume that our transfer operators will preserve weak diagonal dominance (and hence positive definiteness) on every level. To be on the safe side, we would now like to make these statements precise. Our presentation follows the Ph.D. thesis of P. De Zeeuw [37], ch. 3:
Obviously, the MML crucially depends on the fact that we are using Galerkin coarse grid operators with an underlying variational principle, i.e. we obtain the coarse level representations via

$$R_{l-1} = P_{l-1} \tag{5.18}$$
$$A_{l-1} = R_{l-1} A_l P_l. \tag{5.19}$$

With $\langle \cdot, \cdot \rangle_{l-1}$ and $\langle \cdot, \cdot \rangle_l$ denoting the Euclidian inner product on the spaces corresponding to the coarser level $l-1$ and the finer level l, respectively, we see that

$$\langle A_{l-1} u_{l-1}, v_{l-1} \rangle_{l-1} = \langle A_l P_l u_{l-1}, P_l v_{l-1} \rangle_l \tag{5.20}$$

for all vectors u_{l-1} and v_{l-1} in the space belonging to the level $l-1$. From (5.20) we immediately understand that Galerkin coarsening with $R_l = P_l^T$ preserves symmetry.

Anyway, we would like to have stronger results. For an M-matrix A we can easily prove that MML-prolongations based on $B = abs(A)$ preserve the M-matrix property:
Let 1_{l-1} and 1_l denote the vectors of ones on the spaces belonging to the levels $l-1$ and l, respectively. Trivially, prolongations based on $B = abs(A)$ satisfy $P_l 1_{l-1} = c_l * 1_l$ with a constant $c_l \in \mathbb{R}$ and we can prove the following lemma (see [37], p. 48):

Lemma 8 (Preservation of M-matrix property)
Let the coarse level matrices in a multigrid scheme be constructed via Galerkin coarsening according to (5.18) and (5.19) and let the prolongation operators P_l satisfy $P_l 1_{l-1} = c_l * 1_l$ with a constant $c_l \in \mathbb{R}$ on every level l. Then there

holds:
(i) if every row sum of the matrix A_l equal zero, then every row sum of A_{l-1} equals zero.
(ii) if every column sum of the matrix A_l equal zero, then every column sum of A_{l-1} equals zero.

The proofs can both be given in one line each: As for (i) observe that

$$A_{l-1}1_{l-1} = R_{l-1}A_lP_l1_{l-1} = R_{l-1}c_l * (A_{l-1}1_{l-1}) = R_{l-1}0_{l-1} = 0_{l-1}$$

Similarly, (ii) holds because

$$A_{l-1}^T1_{l-1} = (R_{l-1}A_lP_l)^T1_{l-1} = R_{l-1}A_l^TP_l1_{l-1} = R_{l-1}c_l * (A_{l-1}^T1_{l-1}) = 0_{l-1}.$$

Finally, note that the above lemma is valid even for nonsymmetric M-matrices.

5.5.3 A general MML algorithm for one-dimensional problems and numerical results

We have not forgotten that our goal was an efficient multigrid algorithm which also works in the case when we are not dealing with M-matrices: If A is a symmetric weakly diagonally dominant tridiagonal matrix, then there is an obvious generalization of the choice $B = abs(A)$, namely

$$B = 2 * diag(A) - A. \tag{5.21}$$

Furthermore, it is clear that if we build prolongations via (5.21) on the finest level, then the matrix A_2 on the second level will already have become an M-matrix, because all the positive entries in $A = A_1$ will have been multiplied appropriately with negative weights. Note that the choice (5.21) for setting up transfer operators reflects the original MML Algorithms 9 and 10 based on estimating the largest eigenvalue of a diagonally scaled matrix.

We can now formulate a general one-dimensional MML algorithm for weakly diagonally dominant M-matrices A:

Algorithm 13 (General additive MML in 1D – scalings optional)
If we do not want to scale on every level then we need to start with the matrix $A = A_1$ and to follow steps (1),(2) and (3).
If we choose the option to scale on every level, then we need to start with the

scaled matrix $\tilde{A}_1$ and to follow steps (1),(2a) and (3a) instead:
(1) On every level use $B_j = 2 * diag(A_j) - A_j$ and $C_j = B_j(:, 2:2:n)$ for prolongation and restriction.
(2) After computation of $A_{j+1} = C_j^T A_j C_j$ we save the diagonal entries in D_{j+1} and step forward with A_{j+1}.
(2a) After computation of $A_{j+1} = C_j^T \tilde{A}_j C_j$ we step forward with the scaled matrix $\tilde{A}_{j+1}$.
(3) Based on the matrices C_j and D_j, $j = 0, ..., l$, we can recursively implement the multiplication of the matrix (5.10) with a given vector and thus use the preconditioned CG method.
(3a) Based on the matrices C_j and $D_{j+1} = diag(C_j^T \tilde{A}_j C_j)$, $j = 0, ..., l$, we can recursively implement the multiplication of the matrix (5.11) with a given vector and thus use the preconditioned CG method.

The multiplicative counterparts of Algorithm 13 are given analogously to Algorithm 10 in case diagonal scalings are to be included and analogously to Algorithm 12 if we do not want to include diagonal scalings.
As we have explained before, we could in step (1) of Algorithm 13 also set $B_j = abs(A_j)$ for $j > 1$, i.e. as soon as we are no longer on the finest level.

Finally, we still need to test whether it is preferable to include the diagonal scalings or not. Therefore we revisit tables 5.7, 5.9 and 5.11:

n	MML: Alg. 13 with scalings	MML: Alg. 13 without scalings
2^7	13	22
2^8	14	22
2^9	16	26
2^{10}	17	29
2^{11}	18	39
2^{12}	19	39

Table 5.7a. CG iteration numbers for example (EX7), $rex = 10^{-4}$.

n	MML: Alg. 13 with scalings	MML: Alg. 13 without scalings
2^7	7	7
2^8	7	7
2^9	8	7
2^{10}	7	6
2^{11}	7	6
2^{12}	7	6

Table 5.9a. V-cycle iteration numbers for example (EX7), $rex = 10^{-6}$.

n	MML: Alg. 13 with scalings	MML: Alg. 13 without scalings
2^7	5	5
2^8	5	5
2^9	5	5
2^{10}	5	5
2^{11}	5	5
2^{12}	5	5

Table 5.11a. CG iteration numbers for $T_2 = tridiag(1, 2, 1)$ with $rex = 10^{-4}$.

From all our experiments we conclude that for additive preconditioners it comes out to be favourable to work with diagonally scaled matrices. As for the V-cycles our tests results showed that the scalings usually did not make any difference in terms of iteration counts.

5.6 The two-dimensional case

The aim of this section is to generalize the derived method to the higher dimensional case. Therefore we have to decompose the original ill-conditioned matrix into a sum of one-dimensional problems, e.g. $A = A_1 + A_2$. Here, A_1 and A_2 should be ill-conditioned, too, and such that the 1D Matrix Multilevel approach works well. Furthermore, both matrices should be independent in a certain sense, for otherwise the projections relative to A_1 lead to a strong change of A_2 and vice versa. Such a decomposition is given in a natural way by a PDE in terms of the parts relative to the derivatives in x-direction and in y-direction. In the following we mainly consider uniform finite difference

discretizations of the 2D elliptic PDE

$$(a(x,y)u(x,y)_x)_x + (b(x,y)u(x,y)_y)_y = f(x,y). \tag{5.22}$$

5.6.1 Algorithms for separable problems

Let us first focus on a separable PDE, i.e. $a(x,y) \equiv a(x)$ and $b(x,y) \equiv b(y)$. Then the matrix A can be written as a Kronecker sum (see e.g. [70])

$$A = (A_{K1} \otimes I) \; + \; (I \otimes A_{K2})$$

The matrix B that is applied in the standard multigrid approach is given not by a Kronecker sum but by a Kronecker product $B_1 \otimes B_2$ with $B_i = tridiag(1,2,1)$. Hence, we will also choose a matrix B as a Kronecker product $B = B_1 \otimes B_2$ and then use B to define the matrix C in the form $C = BP$ with P denoting the projection matrix.
Generalizing the one-dimensional MML, i.e. Algorithm 13, we will choose our prolongations to be $B = B_1 \otimes B_2$ with

$$B_1 := 2 * diag(A_{K1}) - A_{K1} \quad \text{and} \quad B_2 := 2 * diag(A_{K2}) - A_{K2}$$

Furthermore, the elementary projection matrix P can be chosen as $P = P_1 \otimes P_2$, and then on the second level we get

$$C^T AC \;=\; (C_1^T A_{K1} C_1 \otimes C_2^T C_2) + (C_1^T C_1 \otimes C_2^T A_{K2} C_2)\;.$$

In the next step we can again define the restriction matrix via the Kronecker product where the first factor is designed to improve on $C_1^T A_{K1} C_1$, and the second factor is related to $C_2^T A_{K2} C_2$. This is the direct generalization of the 1D approach to higher dimensions via the Kronecker product. The 2D projection is derived as the Kronecker product of the 1D projections for the separated problems in A_{K1} and A_{K2}.

In the previous sections we showed that we might prefer to apply the Matrix Multilevel method in connection with diagonal scalings on each level, particularly for additive preconditioners. This leads to difficulties in the higher dimensional case: Then the 1D and the 2D diagonal scaling do not correspond anymore. A resort would be to use the Jacobi scaled 1D matrices $A_1^{(j)}$ and $A_2^{(j)}$ with the diagonal matrices $D_1^{(j)}$ and $D_2^{(j)}$ for constructing the prolongation matrices on every level j; in the same way the matrix $A^{(j)}$ could

be scaled by $D_1^{(j)} \otimes D_2^{(j)}$. Furthermore, the diagonal of the matrix $A^{(j)}$ could be used for smoothing in the form (5.11).

The following algorithm summarizes the MML algorithm for separable problems both with and without diagonal scalings:

Algorithm 14 (Additive MML for problems with Kronecker structure $A = A_{K1} \otimes I + I \otimes A_{K2}$ – with optional scalings)
First set $A_1^{(1)} = A_{K1}$, $A_2^{(1)} = A_{K2}$ and $A^{(1)} = A$:
- OPTIONAL: On every level j scale the matrices $A_1^{(j)}$ and $A_2^{(j)}$ with their diagonals $D_1^{(j)}$ and $D_2^{(j)}$, respectively. Then also scale $A^{(j)}$ on every level with $kron(D_1^{(j)}, D_2^{(j)})$.
- Like in Algorithm 13 compute

$$B_i^{(j)} = 2 * diag(A_i^{(j)}) - A_i^{(j)}$$

and $C_i^{(j)} = B_i^{(j)}(:, 2:2:n)$ for $i = 1, 2$, on every level j. Then we get the prolongation by $C^{(j)} = C_1^{(j)} \otimes C_2^{(j)}$
- Step forward with the matrices $A_i^{(j+1)} = (C_i^{(j)})^T A_i^{(j)} C_i^{(j)}$ for $i = 1, 2$ and with $A^{(j+1)} = (C^{(j)})^T A^{(j)} C^{(j)}$
- Then we can implement an additive preconditioner for the CG method according to (5.10) or (5.11) in the cases with or without scalings, respectively.

Note that if we decide to include scalings we can expect very good results particularly for the case that for the original problem the 1D and 2D diagonal scalings coincide, e.g. if $diag(A_1) = const * I$ and $diag(A_2) = const * I$, or if the partial problems A_{K1} and A_{K2} are already diagonally scaled.

5.6.2 Algorithms for general problems in more than one dimension

Now let us consider general matrices of the form $A = A_1 + A_2$ where A_1 and A_2 can be solved efficiently by the 1D Matrix Multilevel approach. We have to replace the Kronecker products by usual multiplications of matrices. To this aim we use a kind of semicoarsening and think of the prolongations in the form

$$\begin{aligned} C = BP = (B_1 \otimes B_2)(P_1 \otimes P_2) &= (B_1 P_1 \otimes B_2 P_2) = \\ = (B_1 P_1 \otimes I)\,(I_r \otimes B_2 P_2) &= F_1 * F_2 \quad , \end{aligned}$$

where I_r is the identity matrix of half size and each matrix F_i reduces the size by a factor 1/2. Then we are able to write the prolongation in the following form

$$\begin{aligned} F = C_1 \otimes C_2 &= (B_1 P_1) \otimes (B_2 P_2) = \\ = (B_1 \otimes I)(P_1 \otimes I) * (I_r \otimes B_2 P_2) &= \hat{B}_1 (P_1 \otimes I) * (I_r \otimes B_2)(I_r \otimes P_2) = \\ = \hat{B}_1 (P_1 \otimes I) \ * \ \hat{B}_2 (I_r \otimes P_2) &= F_1 * F_2 \end{aligned} \tag{5.23}$$

where $\hat{B}_1 = 2 * diag(A_1) - A_1$ and $\hat{B}_2 = 2 * diag(\hat{A}_2) - \hat{A}_2$ with the matrix $\hat{A}_2 = A_2(2:2:n, 2:2:n)$ chosen to be the result of a trivial projection of A_2 in x-direction. Hence, we get the whole 2D prolongation by $F = F_1 * F_2$ with F_1 an $n \times (n/2)$ matrix, and F_2 an $(n/2) \times (n/4)$ matrix. Then FAF^T is an $(n/4) \times (n/4)$ matrix. Thereby we do not need any Kronecker structure of the given matrix A, but only two independent ill-conditioned parts A_1 and A_2. The numerical realisation is analogous to Algorithm 14:

Algorithm 15 (Additive MML for problems with structure $A = A_1 + A_2$ – with optional scalings)
First set $A_1^{(1)} = A_1$, $A_2^{(1)} = A_2$ and $A^{(1)} = A$:
- OPTIONAL: On every level j scale the matrices $A_1^{(j)}$ and $A_2^{(j)}$ with their diagonals $D_1^{(j)}$ and $D_2^{(j)}$, respectively. Then also scale $A^{(j)}$ on every level with $D_1^{(j)} * D_2^{(j)}$.
- Like in Algorithm 13 compute

$$\hat{B}_1^{(j)} = 2 * diag(A_1^{(j)}) - A_1^{(j)} \quad and \quad \hat{B}_2^{(j)} = 2 * diag(\hat{A}_2^{(j)}) - \hat{A}_2^{(j)}$$

with $\hat{A}_2^{(j)} = A_2^{(j)}(2:2:n, 2:2:n)$ on every level j. Then we get the prolongation $C^{(j)}$ via (5.23).
- Step forward with the matrices $A_i^{(j+1)} = (C_i^{(j)})^T A_i^{(j)} C_i^{(j)}$ for $i = 1, 2$ and with $A^{(j+1)} = (C^{(j)})^T A^{(j)} C^{(j)}$
- Then we can implement an additive preconditioner for the CG method according to (5.10) or (5.11) in the cases with or without scalings, respectively.

The multiplicative versions of Algorithms 14 and 15 are almost identical to the additive algorithms: Prolongations and restrictions are obtained in the same way and the identical coarse grid matrices are used. In case scalings are included the residuals are restricted on the level j by multiplication with $(kron(D_1^{(j)}, D_2^{(j)}))^{-1/2} * (C^{(j)})^T$ in the Kronecker case and with $(D_1^{(j)} D_2^{(j)})^{-1/2} *$

$(C^{(j)})^T$ in the general case, respectively; otherwise we can simply use $(C^{(j)})^T$ for restriction.

In general we can write A in the form $A = A_1 + \cdots + A_d$ for a d-dimensional problem, where each term is related to the i-th direction. Then for A_1 we can define a matrix $B_1 = 2 * diag(A_1) - A_1$ and apply the related prolongation F_1 to A_1, and trivial projections in x-direction on $A_2, ..., A_d$. Then, F_2 is given by the projected A_2, and elementary projections are again applied on $A_3, ..., A_d$ and so on. In the end we define $F = F_1 F_2 \cdots F_d$ with F_j of decreasing dimension.

5.6.3 Numerical Experiments for separable problems

The following numerical examples compare Algorithm 14 (Kronecker) and Algorithm 15 (directionwise approach) to the standard approach for 2D problems. Furthermore, we still have to investigate if and when to use scaled versions of our algorithms. In our tables scaled versions will be identified by a bracketed "SC".

Let us first take a look at separable problems: Tables 5.12 and 5.13 deal with additive preconditioners for CG:

n	MDS	MML	Alg. 14	Alg. 14 (SC)	Alg. 15	Alg. 15 (SC)
$2^4 * 2^4$	23		20	18	20	18
$2^5 * 2^5$	28		25	26	25	26
$2^6 * 2^6$	37		38	31	38	31
$2^7 * 2^7$	44		52	38	52	38

Table 5.12. CG iteration numbers for $A = A_{K1} \otimes I + I \otimes A_{K2}$, $rex = 10^{-4}$, with A_{K1} from example (EX1) and A_{K2} from example (EX7).

n	MDS	MML	Alg. 14	Alg. 14 (SC)	Alg. 15	Alg. 15 (SC)
$2^4 * 2^4$	28		26	13	26	13
$2^5 * 2^5$	38		32	15	32	15
$2^6 * 2^6$	57		51	17	51	17
$2^7 * 2^7$	82		69	19	69	19

Table 5.13. CG iteration numbers for $A = A_{K1} \otimes I + I \otimes A_{K2}$, $rex = 10^{-4}$, with A_{K1} the diagonally scaled matrix from example (EX3) and A_{K2} the diagonally scaled matrix from example (EX7).

Note that the problem investigated in table 5.13 is different in style from the one in table 5.12, because the matrices A_{K1} and A_{K2} were diagonally scaled in advance.

Tables 5.14 and 5.15 list iteration numbers for V-cycle solvers (with one pre- and one postsmoothing step of the symmetric Gauss-Seidel method) for separable problems:

n	MDS	MML	Alg. 14	Alg. 14 (SC)	Alg. 15	Alg. 15 (SC)
$2^5 * 2^5$	30		16	16	16	16
$2^6 * 2^6$	44		16	16	16	16
$2^7 * 2^7$	51		16	16	16	17
$2^8 * 2^8$	115		17	17	17	17
$2^9 * 2^9$	168		17	17	17	17

Table 5.14. Numbers of V-cycle iterations for A from table 5.12, $rex = 10^{-6}$.

n	MDS	MML	Alg. 14	Alg. 14 (SC)	Alg. 15	Alg. 15 (SC)
$2^5 * 2^5$	76		25	25	25	25
$2^6 * 2^6$	145		25	28	25	28
$2^7 * 2^7$	238		27	28	27	28
$2^8 * 2^8$	> 250		28	28	28	29
$2^9 * 2^9$	> 250		28	28	28	28

Table 5.15. Numbers of V-cycle iterations for $A = B \otimes I + I \otimes B$, $rex = 10^{-6}$, with B from example (EX7).

Tables 5.12 to 5.15 clearly demonstrate the superiority of the MML transfer operators in comparison to standard prolongations and restrictions. Like in 1D, diagonal scalings might pay off in additive preconditioners. Finally, we would like to emphasize that for the problems in tables 5.14 and 5.15 the MML shows optimal computational behaviour – in the sense that the numbers of iterations needed are independent of the problem size.

5.6.4 Numerical experiments for general two-dimensional problems

We would now like to test general problems $A = A_1 + A_2$ without Kronecker structure. Therefore we consider uniform finite difference discretizations of 2D elliptic PDEs of the form (5.22) with right hand side $f(x, y) \equiv 1$ and homogenous Dirichlet boundary conditions on the unit square.

Let us first take a look at three problems where diagonal scalings seem reasonable and might lead to a better performance of the MML, i.e. we look at the case $diag(A_1) = diag(A_2)$ or where the diagonal entries of A_1 and A_2 are at least of the same orders of magnitude:

n	Standard MG	MML	Algorithm 15	Algorithm 15 (SC)
$2^5 * 2^5$	16		8	8
$2^6 * 2^6$	18		8	8
$2^7 * 2^7$	22		8	8
$2^8 * 2^8$	23		8	8
$2^9 * 2^9$	23		8	8

Table 5.16. Numbers of V-cycle iterations for a problem of the form (5.22) with $a(x, y) = b(x, y) = 1 + exp(8 * (x + y)) * sin(2 * (x + y))^2$, $rex = 10^{-6}$.

n	Standard MG	MML	Algorithm 15	Algorithm 15 (SC)
$2^5 * 2^5$	26		13	13
$2^6 * 2^6$	33		13	13
$2^7 * 2^7$	62		13	19
$2^8 * 2^8$	66		17	19
$2^9 * 2^9$	68		17	20

Table 5.17. Numbers of V-cycle iterations for a problem of the form (5.22) with $a(x, y) = 1 + exp(16 * (x + y)) * sin(2 * (x + y))^2$ and $b(x, y) = 1 + exp(17 * (x + y)) * sin(2 * (x + y))^2$, $rex = 10^{-6}$.

Let us also take a look at an example that is not an M-matrix in order to demonstrate the broader applicability of the two-dimensional MML compared to standard multigrid. This matrix M was obtained by first setting up the block tridiagonal matrix $A = T \otimes I + I \otimes T$ with $T = tridiag(1, 2, 1)$. Afterwards we only left the first and last off-diagonal blocks as well as the

second and last but one diagonal block untouched, but we made all the other off-diagonal entries jump between 1 and -1 by flipping signs.

n	Standard MG	MML	Algorithm 15	Algorithm 15 (SC)
$2^5 * 2^5$	79		5	5
$2^6 * 2^6$	221		5	5
$2^7 * 2^7$	> 250		5	5
$2^8 * 2^8$	> 250		5	5
$2^9 * 2^9$	> 250		5	5

Table 5.18. Numbers of V-cycle iterations for 'jump matrix' M, $rex = 10^{-6}$.

Finally, let us go for a completely different example that is frequently used to test algebraic multigrid approaches, the so-called **cut-square problem** (see e.g. [5] or [4]): It is a version of (5.22) with discontinuous coefficients and we define it by

$$a(x,y) = b(x,y) = \begin{cases} 1 & \text{for } (x,y) \in [0,1]^2 \backslash [0.25, 0.75]^2 \\ val & \text{for } (x,y) \in [0.25, 0.75]^2 \end{cases} \tag{5.24}$$

For such an interface problem we can no longer expect diagonal scalings to be reasonable – and this is confirmed in our numerical experiments:

n	Standard MG	MML	Algorithm 15	Algorithm 15 (SC)
$2^5 * 2^5$	104		11	> 250
$2^6 * 2^6$	111		11	> 250
$2^7 * 2^7$	116		11	> 250
$2^8 * 2^8$	120		11	> 250
$2^9 * 2^9$	123		11	> 250

Table 5.19. Numbers of V-cycle iterations for the cut-square problem (5.24) with $val = 100$, $rex = 10^{-6}$.

Table 5.19 is very remarkable: MML without scalings shows optimal computational behaviour and beats standard multigrid significantly. On the other hand, if we include diagonal scalings within Algorithm 15 then the performance deteriorates completely.

5.6.5 Brief analysis of the two-dimensional MML prolongations

We would now like to explain the behaviour observed in table 5.19 in more detail:
Let us start with a five-point discretization of the cut-square problem (5.24) which we will again write as $A = A_1 + A_2$: When we run Algorithm 15 on that problem, we apply first apply a semicoarsening step in x-direction based on $B_1 = abs(A_1)$ which is then followed by coarsening in y-direction based $B_2 = abs(\hat{A}_2)$ with $\hat{A}_2 = A_2(2:2:n, 2:2:n)$. According to subsection 5.5.1 we can write these directionswise prolongation operators as stencils in the forms

$$p^{(x)} = [p_{-1}^{(x)} \quad 1 \quad p_1^{(x)}] \qquad \text{and} \qquad p^{(y)} = [p_{-1}^{(y)} \quad 1 \quad p_1^{(y)}]^T \tag{5.25}$$

with $p_{-1}^{(x)} + p_1^{(x)} = 1$ and $p_{-1}^{(y)} + p_1^{(y)} = 1$. It is now very obvious why scaling the matrices A_1 and A_2 individually before computing the transfer operators would lead to totally corrupted stencils; basically, all the information on the underlying differential equation (5.24) and directional dependencies would be lost if we scaled A_1 and A_2 in advance.

Note that we can merge the two one-dimensional stencils (5.25) into one big stencil expressing the two-dimensional prolongation operator as

$$\mathbf{p} = \begin{bmatrix} p_{-1}^{(x)}p_{-1}^{(y)} & p_{-1}^{(x)} & p_{-1}^{(x)}p_1^{(y)} \\ p_{-1}^{(y)} & 1 & p_1^{(y)} \\ p_1^{(x)}p_{-1}^{(y)} & p_1^{(x)} & p_1^{(x)}p_1^{(y)} \end{bmatrix}$$

Obviously, for any prolongation p_l defined via the above stencil there holds $p_l 1_{l-1} = 1_l$ on every level l: Hence Lemma 8 is applicable and we can be sure that Algorithm 15 will preserve the M-matrix property on every level. (Note that both statement and proof of Lemma 8 are independent of the dimension.)

We would finally like to mention that ideas for prolongations very similar to (5.25) have been proposed by Alcouffe, Brandt, Dendy and Painter in [1] and by Griebel in [56], ch. 5, for M-matrices arising from diffusion equations and convection-diffusion problems, respectively. However, their algorithms differ from Algorithm 15 in certain aspects; in particular, they do not set up coarse grid versions of the matrices A_1 and A_2 in the coordinate directions.

Other successful matrix-dependent prolongation operators have been proposed by Dendy in his "black box multigrid" approach in [34] and [35] on the basis of "platting" discretization stencils.

5.6.6 Summary, conclusions and outlook

We have developed a purely matrix-dependent multilevel scheme which can be applied to symmetric weakly diagonally dominant matrices A having the same sparsity pattern as a uniform 9-point discretization of the Laplacian, i.e. A is block tridiagonal with tridiagonal blocks.
As we have pointed out in the last two subsections, it might be dangerous to include diagonal scalings in Algorithm 15 in higher dimensions – and, hence, except for special cases outlined in subsections 5.6.3 and 5.6.4, we strongly recommend not to do it.
As a major result of our investigations we would at this point like to repeat Algorithm 15 in a multiplicative version without mentioning the optional diagonal scalings. The reader may view it as the "final" version of the MML algorithm for two-dimensional problems:

Algorithm 16 (Multiplicative MML for 2D problems with structure $A = A_1 + A_2$)
First set $A_1^{(1)} = A_1$, $A_2^{(1)} = A_2$ and $A^{(1)} = A$:
Like in Algorithm 13 compute

$$\hat{B}_1^{(j)} = 2 * diag(A_1^{(j)}) - A_1^{(j)} \quad and \quad \hat{B}_2^{(j)} = 2 * diag(\hat{A}_2^{(j)}) - \hat{A}_2^{(j)}$$

with $\hat{A}_2^{(j)} = A_2^{(j)}(2:2:n, 2:2:n)$ on every level j. Then we get the prolongation $C^{(j)}$ via (5.23).
- Step forward with the matrices $A_i^{(j+1)} = (C_i^{(j)})^T A_i^{(j)} C_i^{(j)}$ for $i = 1, 2$ and with $A^{(j+1)} = (C^{(j)})^T A^{(j)} C^{(j)}$
- When going down we restrict the residuals using $R^{(j)} = (C^{(j)})^T$ and when going up we use $C^{(j)}$ for prolongation.

A possibility for future investigations could be to test Algorithm 16 – or its three-dimensional counterpart – for challenging convection-diffusion problems, e.g. with discontinuous coefficients. However, we are aware that there are very successful and established matrix-dependent transfer operators for convection-diffusion problems by De Zeeuw [36]. First experiments for rather

simple convection-diffusion equations are indicating that these transfer operators are strongly superior to Algorithm 16. But note that this is not actually surprising as De Zeeuw's prolongations are particularly designed for convection-diffusion problems, whereas the MML was meant to be a general approach for symmetric weakly diagonally dominant matrices.

5.7 Case study on elliptic equations with highly oscillatory coefficients

We have established that the MML gives a very robust scheme suitable for elliptic problems with highly oscillatory or discontinuous coefficients. In the following case study we would like to compare the behaviour of the MML for problems with rapidly oscillating coefficients to an approach by Engquist and Luo (see [82], [45] and [44]). They proposed to set up coarse grid operators analytically by discretizing the corresponding homogenized equations. Similarly to the MML, their methods are also motivated via eigenvalues: One characteristic feature of problems with highly oscillatory coefficients is that the smallest eigenvalues do not belong to very smooth eigenfunctions – and thus, Engquist's and Luo's algorithms try to cure the fact the standard multigrid can not represent well the smooth eigenfunctions on coarser grids.

5.7.1 The work by Engquist and Luo

Following [82], ch. 1, let us consider a uniform finite difference discretization of the two-point boundary value prolem

$$(b(\frac{x}{\epsilon})u_\epsilon(x)_x)_x = f(x) \tag{5.26}$$

with homogenous Dirichlet boundary conditions and continuous right hand side $f(x)$ on the unit interval. In (5.26) $b_\epsilon(x) = b(\frac{x}{\epsilon})$ is a 1-periodic positive function.
From the theory of homogenization (see e.g. [3]) we know that as ϵ goes to 0, the solution of (5.26) converges strongly in the $\|\cdot\|_\infty$-norm to the solution u of the corresponding homogenized equation

$$-(\int_0^1 \frac{1}{b(s)}ds)^{-1}u_{xx} = f(x) \tag{5.27}$$

subject to the same boundary conditions as (5.26). Observe that (5.27) no longer has oscillating coefficients; instead, it is a (well-behaved) 1D Laplacian. Note that $\mu = (\int_0^1 \frac{1}{b(s)} ds)^{-1}$ denotes the *harmonic average*.
Based on the observation that standard multigrid with Galerkin coarsening has difficulties treating problems of the type (5.26) and that natural coarse grid operators fail completely, Engquist and Luo proposed to use a discrete version of (5.27) as the coarse grid representation.
Luo was able to prove the following theorem (see [82], pp. 51):

Theorem 8 (Twogrid with homogenized coarse grid operator)
We want to solve the linear system $A_{\epsilon,h}x = b$ which represents a discretization of (5.26). Assume we would like to solve this system via a twogrid method employing a discrete version A_H of (5.27) as the coarse grid representation. Let the smoothing operator be chosen as $S = I - \phi A_{\epsilon,h}$ with ϕ denoting the inverse of the largest eigenvalue of $A_{\epsilon,h}$. Then, like in (2.12), the twogrid operator M with γ smoothing steps can be written as

$$M = (I - PA_H^{-1}RA_{\epsilon,h})S^\gamma$$

The twogrid method converges, i.e.

$$\rho(M) \leq \rho_0 < 1,$$

whenever there exists a constant C such that either one of the following conditions is satisfied:
(i) the ratio of h to ϵ is less than $\epsilon^{4/3}$ and

$$\gamma \geq Ch^{-2}\epsilon^{4/9}ln(\epsilon);$$

(ii) the ratio of h to ϵ belongs to the set of Diophantine numbers and

$$\gamma \geq Ch^{-1.2}ln(h)$$

Note that if the theory of homogenization is applied in numerical analysis then the ratio of h and ϵ plays a major role: In particular, it should be irrational – a fact that has its mathematical foundations in ergodic theory (see [43] for details).

5.7.2 Numerical results in 1D

We will investigate four different choices to combine transfer operators and coarse grid representations to solve a discrete version of (5.26):

- Galerkin operator and standard prolongations (GAL – Std.),
- Galerkin operator and MML prolongations according to Algorithm 12 (GAL – MML),
- Homogenized operator and standard prolongations (HOM – Std.),
- Homogenized operator and MML prolongations according to Algorithm 12 (HOM – MML).

We choose the same example problem as Luo in her Ph.D. thesis (see [82], p. 63), i.e. we set

$$b_\epsilon(x) = 2.1 + 2 * \sin(2\pi \frac{x}{\epsilon}) \tag{5.28}$$

with the harmonic average

$$\mu = m(1/b_\epsilon)^{-1} = 1.5618$$

in equation (5.27).

Like in [82], ch. 4, we choose $\epsilon = \sqrt{2}h$ in our numerical examples, i.e. in the following table the value of ϵ depends on the number of unknowns n. Clearly, in the context of homogenized coarse grid representations it is very sensible to investigate on twogrid steps only.

n	GAL – Std.	GAL – MML	HOM – Std.	HOM – MML
2^7	90	7	85	76
2^8	97	7	92	82
2^9	100	7	96	86
2^10	106	7	114	99
2^11	108	7	104	93
2^12	112	7	107	96

Table 5.20. Iteration numbers for a twogrid method for solving (5.28) with $\epsilon = \sqrt{2}/h$ and two Gauss-Seidel steps for smoothing.

Table 5.20 points out that Galerkin coarsening with proper matrix-dependent

transfer operators is the clear winner. Anyway, with homogenized coarse grid operators one may still beat standard multigrid; furthermore, Engquist's and Luo's approach works better if combined with MML prolongations than with standard transfer operators.

5.7.3 Numerical results in 2D

As it is very common with convergence results for multigrid algorithms without Galerkin coarse grid operators there is no analogue to Theorem 8 for general problems in 2D, because one does not know enough about eigenvalues and eigenvectors of the system matrices in that case. However, as pointed out [82], pp. 58, Theorem 8 can be carried over to separable problems of the form (5.22) in case the coefficients are highly oscillatory in one direction only. Hence let us investigate uniform finite difference discretizations on the unit square of the separable problem

$$(a(x)u(x,y)_x)_x + (b_\epsilon(y)u(x,y)_y)_y = f(x,y) \tag{5.29}$$

with $a(x) = 1$ and $b_\epsilon(y) = 2.1 + 2 * \sin(2\pi\frac{y}{\epsilon})$ as before.

As pointed out in [82], pp. 56, the homogenized equation corresponding to (5.29) has the form

$$\bar{a} * u_{xx} + \mu * u_{yy} = f(x,y)$$

where $\bar{a}$ denotes the arithmetic average

$$\bar{a} = m(b_\epsilon) = \int_0^\epsilon a(y)dy = 2.1$$

and as before μ denotes the harmonic average

$$\mu = m(1/b_\epsilon)^{-1} = 1.5618$$

Like in 1D we will investigate four different choices to combine transfer operators and coarse grid representations to solve a discrete version of (5.29):

- Galerkin operator and standard prolongations (GAL – Std.),
- Galerkin operator and MML prolongations according to Algorithm 16 (GAL – MML),

- Homogenized operator and standard prolongations (HOM – Std.),
- Homogenized operator and MML prolongations according to Algorithm 16 (HOM – MML).

Here are the results for twogrid step with fixed $\epsilon = \sqrt{2}/128$.

n	GAL – Std.	GAL - MML	HOM – Std.	HOM – MML
$2^5 * 2^5$	27	9	41	37
$2^6 * 2^6$	85	10	76	60
$2^7 * 2^7$	73	9	70	61
$2^8 * 2^8$	93	9	81	75

Table 5.21. Iteration numbers for a twogrid method for solving (5.29) with $\epsilon = \sqrt{2}/128$ and two Gauss-Seidel smoothing steps.

Our observations are like in one dimension: Again, proper variational coarsening with the MML beats all the other approaches by far: We understand that "Galerkin coarsening gives good homogenization".

This case study emphasizes on the merits of Galerkin coarsening for certain classes of problems: In Chapter 4 we saw that for dense Toeplitz matrices it is preferable not to take the difficulties of setting up a Galerkin operator and replace it by an appropriate natural coarse grid operator instead. However, for the class of problems investigated in this section Galerkin coarsening with proper matrix-dependent prolongations and restrictions simply needs to be the method of choice.

Chapter 6

Image deblurring and the multigrid method of the second kind

Today image processing is maybe the most eminent field of applications of Toeplitz matrices (see e.g. [19], [20]). The most well-known example are dense matrices from image deblurring.
In the following we will study Toeplitz and BTTB matrices as they arise from deblurring models based on integral equations of the first kind. We shall use Tikhonov regularization to solve these inverse problems. Our investigations are based on a proposal for a multigrid algorithm by R. Chan, T. Chan and W. Wan [22] which employs a semi-iterative smoother. We shall put their algorithm into the context of the so-called "multigrid method of the second kind" by Hackbusch (see [62], ch. 16, and [63], ch. 5) which allows to treat integral equations of the second kind efficiently and uses Richardson as a smoother. Indeed, we confirm the need for a semi-iterative smoother expressed in [22]. However, we can significantly improve the algorithms by R. Chan, T. Chan and W. Wan by using a natural coarse grid operator: We shall explain clearly why an efficient $O(n \log n)$ algorithm for image deblurring can in general only be gained if coarse grid operators are constructed via rediscretizations. Finally, we test our new multigrid algorithms for reconstructing images subject to atmospheric turbulence blur and Gaussian white noise.
Parts of the results of this chapter have already been published in sections 4 and 5 of the paper [73].

6.1 Introduction

6.1.1 The one-dimensional model

Let us start with an idealized model for one-dimensional image deblurring. There we want to solve an integral equation of the first kind of the form

$$\mathcal{K}u(x) = \int_{\Omega} k(x - x')u(x')dx' \tag{6.1}$$

with a convolution kernel of the form $k(x) = exp(-x^2/\sigma^2)$ with $\sigma \in]0, 1[$ on the interval $\Omega = [-p, p]$. The operator $\mathcal{K}$ is often referred to as "Gaussian blur".
We can now discretize this integral equation on a uniform grid via the midpoint quadrature rule and will end up with a Toeplitz matrix (see e.g. [19], sec. 4.4, or [91], ch. 2): We work with the mesh size $h = \frac{2p}{n}$ and the midpoints

$$x_j = -p + (2 * j - 1) * \frac{h}{2}, \qquad j = 1, 2, \ldots, n.$$

Then we use the midpoint quadrature rule and the convolution operator (6.1) translates into

$$\mathcal{K}u(x_i) = \int_{-p}^{p} k(x_i - x')u(x')dx' \approx \sum_{j=0}^{n-1} k(x_i - x_j)u(x_j)h \approx [K\bar{u}]_i$$

with the symmetric positive definite Toeplitz matrix

$$K = h * toeplitz(k((1 - n) * h), \ldots, k(0), k(1 * h), \ldots, k((n - 1) * h)) \tag{6.2}$$

and the vector $\bar{u} = [u(x_1), \ldots, u(x_n)]^T$.
We finally note that discretizing an integral operator via quadrature is frequently referred to as *Nyström's method* going back to papers of E. Nyström [85] and [86] from 1928 and 1930, respectively.

6.1.2 Inverse problems and Tikhonov regularization

It is well known that the blurring matrices K are highly ill-conditioned and hence deblurring algorithms are extremely sensitive to noise [52], [80], i.e. we are dealing with an inverse problem and hence we need to regularize.

Let us go a little bit more into detail: The integral operator $\mathcal{K}$ defined in (6.1) is compact (see [117]) and hence we are dealing with an ill-posed problem. Thus it would be completely useless simply to invert the matrix K from (6.2); the computed solution would be totally corrupted by the small eigenvalues of K.
The following paragraphs are not meant to serve as an introduction to inverse and ill-posed problems: Instead for a sound and detailed overview on to this challenging and important subject we refer to the books [106], [81] and [42] for the mathematical theory and to [67] and [112] for the computational aspects.

Generally speaking, the idea of regularization is to replace an ill-posed problem with one that is well-posed. In this thesis we shall only investigate on Tikhonov regularization. Following [91], ch. 2, we can write this idea in an abstract Hilbert space setting as

$$u_\lambda = \arg\min_{u \in \mathcal{S}} (\|\mathcal{K}u - z\|^2 + \lambda \langle \mathcal{L}u, u \rangle), \tag{6.3}$$

i.e. we want to find a regularized solution u_λ – belonging to the regularization parameter λ – that minimizes the expression in the brackets on the right hand side of (6.3) on a given subspace $\mathcal{S}$ of the domain of our integral operator $\mathcal{K}$. We can think of (6.3) as a penalized least squares method and interpret $\|\mathcal{K}u - z\|^2$ as the "discrepancy functional" ruling how well the regularized solution fits the given data and $\langle \mathcal{L}u, u \rangle$ as the penalty functional.

In classical Tikhonov regularization we work in the space $L_2(\Omega)$ and set $\mathcal{L}$ simply to be the identity, i.e. $\langle \mathcal{L}u, u \rangle = \|u\|_2^2$. In that case after discretization our system matrix for solving (6.1) becomes

$$L = K + \lambda I \tag{6.4}$$

with regularization parameter λ. Note that we do not have to work with the normal equations in this special case, because K is guaranteed to be symmetric positive definite. (In other words: In general, (6.3) would result in a system matrix of the form $L = K^T K + \lambda I$.)

There is also another variant of Tikhonov regularization which is sometimes also used in the context of image restoration due to the fact that it penalizes a little better against solutions with spurious oscillations: Instead of the

L_2-norm the H_1-norm is used with

$$\mathcal{L}u = -\nabla \cdot \nabla = -\sum_{j=1}^{d} \frac{\partial^2 u}{\partial x_j^2},$$

i.e. $\mathcal{L}$ is the d-dimensional Laplace operator. In this case we speak of H_1-regularization. After discretization our system matrix (6.1) becomes

$$L = K + \lambda \triangle$$

with $\triangle = tridiag(-1, 2, -1)$ denoting the one-dimensional Laplacian. (Again, as K is s.p.d. we can avoid setting up the normal equations in this special case.) For simplicity, in the following our focus will lie on the L_2-based case (6.4). However, carrying over our reasoning to the H_1-based case will be straightforward.

6.1.3 The algorithm of R. Chan, T. Chan and W. Wan

Efficient multigrid treatment of image deblurring problems comes out to be rather difficult: So far, there has essentially been only one paper on the subject which was written by R. Chan, T. Chan and W. Wan [22]. (This paper can also be found as chapter 8 in Wan's Ph.D. thesis [113].)

The paper [22] reports that for the system matrices in question standard relaxation methods like Richardson fail as smoothers. (This problem is explained with the fact that small eigenvalues K belong to highly oscillatory eigenvectors whereas large eigenvalues can be associated with smooth eigenvectors.) To overcome this difficulty a semi-iterative smoother is used: They employ conjugate gradients with optimal cosine transform preconditioner [23]. The approach does not make any explicit use of Toeplitz structure; furthermore, standard prolongations and restrictions and Galerkin coarsening are used without any further explanation (see [22], p. 70).

Do the methods presented in Chapter 4 relate to this case? First of all, we need to state that obviously there is no underlying function connected with our matrices K from (6.2). However, if we simply assign functions to matrices of different size, we quickly observe that we are dealing with a "single zero of infinite order" located at $x_0 = \pi$.
Anyway, this information does not help us to devise multigrid transfer operators. The reasoning associated with (4.2) is no longer applicable as we certainly do not want to use anything like $b(x) = (1 - cos(x))^{\infty}$.

6.2 The multigrid method of the second kind enhanced by semi-iterative smoothing

Integral operators of the form (6.1) have been considered for multigrid treatment – however, not in connection with integral equations of the first kind like in the previous section, but in connection with integral equations of the second kind (which usually are much better behaved). We found out that the observations and algorithms reported in [22] make much more sense as soon as they are put into the context of the so-called "multigrid method of the second kind" which was developed mainly by Hackbusch in the eighties for the purpose of an efficient iterative treatment of discrete Fredholm integral equations of the second kind (see [62], ch. 16, and [63], ch. 5).

6.2.1 Appropriate transfer operators

Let us take up the problem of finding suitable transfer operators for (6.2) again with which we ended subsection 6.1.3.
Studying the work of Hackbusch, we see that this question has long been answered: As pointed out [63], sec. 5.3, discretizations K_l of the operator $\mathcal{K}$ and the corresponding prolongations P_l and restrictions R_l need to satisfy certain consistency conditions. From these conditions it follows that canonical choices for P_l and R_l need to be used, in particular the prolongations P_l need to be standard interpolation schemes of the same order as the discretization. For the restrictions R_l trivial injections (2.26) are shown to be the canonical choice, but it is pointed out that weighted schemes with $R_l = P_l^T$ also come out to be suited.
To be more specific let us go back to the discretizations of our integral operator (6.1) via the midpoint quadrature rule: In this case we know that the discretization error will be of order $O(h^2)$ and as explained in [63], p. 176 and p. 185, the prolongations P_l should be the standard linear interpolations given by (2.18). Employing trivial injections (2.26) as restrictions R_l comes out to be the most straightforward choice, because then

$$R_l * P_l = I \tag{6.5}$$

holds (see [63], pp.187, or [62], pp. 308, for details). When we test our algorithms it will later be interesting to check whether weighted restrictions or trivial injections lead to better numerical results.

As from the Toeplitz point of view we would like to emphasize that Hackbusch's theory clearly states that it would be totally inappropriate to scale the matrix via the diagonal matrices $diag(1, e^{\pm i\pi}, e^{\pm 2\pi}, ...)$ from (4.6) in order to move the "zero of infinite order" at $x_0 = \pi$ to the origin for the discrete integral operators K from (6.1). In this case shifting the zero via (4.11) would mean violating consistency conditions and thus corrupt the whole multigrid approach.

6.2.2 The multigrid method of the second kind

Let us view our system matrix $L = K + \lambda * I$ in terms of a Fredholm integral equation of the second kind: Setting $\tilde{k}(x, y) := -k(x, y)$ and obtaining a discretization $\tilde{K}$ via the midpoint quadrature rule we can rewrite (6.4) as

$$L = \lambda * I - \tilde{K} \tag{6.6}$$

in the standard form of an integral equation of the second kind.

It will be very helpful first to take a closer look at the the multigrid method of the second kind and convergence results known about it:
Let $\tilde{K}$ denote a discretization of our original integral operator of convolution type as before and $\lambda > 0$ be positive parameter. The multigrid method of the second kind for solving discrete Fredholm integral equations of the second kind written in the form

$$(I - (1/\lambda) * \tilde{K})x = (1/\lambda) * b \tag{6.7}$$

is a W-cycle solver which usually employs only one presmoothing step of the Richardson iteration and no postsmoothing. Now we know that the Richardson method – if it were used as a stand-alone solver – converges if and only if

$$\rho(\tilde{K}) < \lambda.$$

On the other hand for $\rho(\tilde{K}) \gg \lambda$ Richardson will diverge rapidly (see also [63], pp. 171).
In standard applications of integral equations of the second kind we mostly deal with the case $\lambda = 1$ and then the multigrid method of the second kind will usually work out perfectly for the given problem (6.6). However, as soon

as λ becomes small, problems can arise due to the fact that the Richardson smoother is strongly divergent, because the smoothing steps "worsen" the approximate solution in every iteration (and on each level): Of course, convergence of the Richardson iteration itself is not a condition for the convergence of the multigrid method of the second kind on the problem (6.7). However, as shown in [63], pp. 200, the convergence factor of the standard version of the multigrid method of the second kind (i.e. the variant with one Richardson smoothing step on every level) depends quadratically on $(1/\lambda)$; this means that for very small λ the multigrid method of the second kind will also diverge rapidly.

Furthermore, there are variants of the multigrid method of the second kind which are especially designed for the case of smaller λ and also employ Richardson smoothing: We need to mention in particular an algorithm by Hemker and Schippers [68] and an algorithm by Hackbusch which basically misses out every second smoothing step on each of the coarse levels (see [62], sec. 16.2.3, or [63], sec. 5.5.5). As pointed out in [63], p. 211, Hackbusch's method of missing out certain smoothing steps is not only cheaper computationally, but one is also never worse off than with the variant by Hemker and Schippers. Furthermore, Hackbusch could show that for his improved variant the convergence factor of the W-cycle solvers depends only linearly on $(1/\lambda)$.

6.2.3 Good smoothing via conjugate gradients

In plenty of numerical tests we have checked very carefully that even the improved variants of the multigrid method of the second kind employing Richardson smoothing do not lead to convergent algorithms for the small regularization parameters λ we need to deal with in image deblurring problems. As explained before the small values of λ in (6.4) render the multigrid method of the second kind impractical and make it diverge quickly. Note also that L in (6.4) usually exhibits huge condition numbers due to a vast range of magnitudes of eigenvalues.

According to the observations of R. Chan, T. Chan and W. Wan all our numerical experiments also confirmed the need for a semi-iterative smoother: As we explained in the previous subsection, the problems of Richarson smoothing are mainly due to the fact that it diverges rapidly for small λ. Thus an appropriate smoother would be a scheme that never diverges for problems of

the form (6.4) – and the (preconditioned) conjugate gradient method is an algorithm known to have this property (see e.g. [19]). As we shall see in the next subsection conjugate gradients combined with the circulant preconditioner from Lemma 6 gives a very efficient smoother.

Finally, we would like to emphasize that we improve the algorithms from the paper [22] significantly in one aspect: We again prefer to use a natural coarse grid operator – instead of Galerkin coarsening used by R. Chan, T. Chan and J. Wan (see [22],p. 70) – in order to preserve Toeplitz structure on the coarse levels. Note also that it is problematic to compute the optimal T. Chan circulant preconditioner – which is needed within our PCG smoothing – for a general dense matrix $A \in \mathbb{R}^{n \times n}$ (– like e.g. a coarse grid matrix obtained via the Galerkin approach –) with a computational effort less than $O(n^2)$; on the other hand, as long as we construct our coarse level matrices via rediscretizations we will have Toeplitz structure on all levels and can set up our preconditioner with $O(n)$ efforts as explained in subsection 3.2.3.

Finally, note that the conjugate gradient algorithm can not be expressed as a stationary iterative method in the form (2.8). Thus our multigrid cycles are no longer available as preconditioners for standard Krylov subspace solvers: In fact, our multigrid preconditioner changes during the iteration as a result of the PCG-smoothing. However, we could use so-called "flexible" Krylov subspace methods, like e.g. the FGMRES variant by Saad [93], which allows to use a different preconditioner in every iteration step. But as our system matrix (6.4) is symmetric positive definite, symmetric versions of flexible Krylov solvers would also come into account (see [32]).

6.2.4 Numerical results for one-dimensional problems

In the following we give numerical results for discretizations of the one-dimensional deblurring problem (6.1) on $\Omega = [-1, 1]$ using L_2-based Tikhonov regularization with different regularization parameters λ. As we said we have implemented a multigrid algorithm using conjugate gradients with the optimal circulant preconditioner [28] as a smoother. In all our tables the multigrid solvers employ two presmoothing and no postsmoothing steps.

Throughout the whole chapter we always employ the following stopping criterion to obtain the iteration counts we list in our tables:

$$\frac{\|r^{(j)}\|_\infty}{\|r^{(0)}\|_\infty} \leq 10^{-6}$$

Again, $r^{(j)}$ denotes the residual after j iterations and $r^{(0)}$ the original residual, i.e. we stop iterating when the relative residual corresponding to the maximum norm is less or equal 10^{-6}.

Within our multigrid cycles we also wanted to check whether it is preferable to use trivial injection (2.26) for restriction what is motivated by the fact that it satisfies condition (6.5) or if the standard full-weighting operator (2.20) gives the better results. We will distinguish these two choices by the abbreviations "MG (triv)" and "MG (lin)", respectively.

The following tables 6.1a, 6.1b and 6.1c list the iteration counts for W-cycle solvers with natural coarse grid operators and two presmoothing steps of circulant-preconditioned CG for the choice $\sigma = 0.1$ in (6.1) and for the three regularization parameters $\lambda = 1e-3$, $\lambda = 1e-4$ and $\lambda = 1e-5$. For comparison we also list the iteration counts of CG with circulant preconditioning as a stand-alone solver, abbreviated by "Circ-CG".

number of unknowns	512	1024	2048	4096	8192	16384	32768
MG (triv)	5	5	4	3	3	2	3
MG (lin)	5	4	4	3	3	3	3
Circ-CG	9	9	9	9	9	9	9

Table 6.1a. Iteration counts for 1D deblurring problem (6.4) with $\lambda = 1e-3$.

number of unknowns	512	1024	2048	4096	8192	16384	32768
MG (triv)	6	5	5	5	4	3	3
MG (lin)	9	7	6	5	5	4	4
Circ-CG	15	15	16	15	15	15	15

Table 6.1b. Iteration counts for 1D deblurring problem (6.4) with $\lambda = 1e-4$.

number of unknowns	512	1024	2048	4096	8192	16384	32768
MG (triv)	10	10	8	6	4	4	3
MG (lin)	37	26	17	12	9	7	6
Circ-CG	27	25	27	26	26	26	26

Table 6.1c. Iteration counts for 1D deblurring problem (6.4) with $\lambda = 1e-5$.

Tables 6.1a to 6.1c show that by using circulant-preconditioned conjugate gradients as a smoother we can obtain the typical convergence behaviour of the multigrid method of the second kind also for the case of very small λ, i.e. iteration numbers even decline for a larger number of unknowns. Hence the idea to employ a semi-iterative smoother can be seen as an extension of the multigrid method of the second kind in order to handle very small λ.

Furthermore, we observe that using trivial injection (2.26) as the restriction operator leads to lower iteration counts in our W-cycle solvers. Hence the theoretical condition (6.5) which comes up in the mathematical analysis of Hackbusch is also meaningful in practice. Finally, we would like to emphasize that the multigrid structure comes out to be sensible and to pay off from the point of view that for large problems and small λ our W-cycle solvers need significantly less PCG smoothing steps on the finest level than circulant-preconditioned CG used as a stand-alone solver.

We would like to admit right now that the comparisons with fixed regularization parameters λ reported in the previous tables may seem slightly questionable from the point of view of solving an inverse problem from signal or image processing. Certainly, the regularization parameter would normally not be picked without looking at the matrix first. For criteria for choosing the regularization parameter like e.g. the generalized cross-validation (GCV) method, the C_L-method or L-curve based schemes we again refer to the books by Hansen [67] or Vogel [112]. Anyway, the focus of this work lies on efficient numerical linear algebra and not on regularization and inverse problems; hence we are deliberately presenting our numerical results in this way in order to focus on the strong connection of our algorithms and the multigrid method of the second kind.

In order to strengthen this connection to the multigrid method of the second kind we also applied our algorithms to a problem with a kernel very much different from the blurring operator given by (6.1). The example is taken from [63], p. 124, like (6.1) it is of convolution type but the kernel is defined by

$$k(x,y) = -\frac{1}{\sqrt{|x-y|}}. \tag{6.8}$$

Note that this kernel exhibits a discontinuity.

The following table lists iteration counts for Fredholm integral equations of

the second kind with system matrices

$$L = K - \lambda I$$

with K denoting a discretization of the integral operator with the non-smooth kernel (6.8). Like before, our W-cycle solvers use two steps of PCG for presmoothing:

number of unknowns	512	1024	2048	4096	8192	16384	32768
MG (triv)	5	4	3	3	3	3	2
MG (lin)	5	4	4	4	3	3	2
Circ-CG	7	7	7	7	7	7	7

Table 6.2. Iteration counts for a discrete Fredholm integral equation of the second kind: The kernel is chosen to be (6.8) and $\lambda = 1e - 4$.

Comparing tables 6.1b and 6.2 we reckon that the problem with the kernel (6.8) is less ill-conditioned and hence the discretized problem is easier to solve. In fact, it can proved to be true that the smoother the kernel, the faster the singular values decay to zero – and hence the more ill-conditioned are the discretization matrices (see e.g. [67]).

6.3 The twodimensional case

6.3.1 The model and its discretization

It is straightforward to carry over the algorithms from the previous section to practical (i.e. two-dimensional) image deblurring problems: There we are dealing with a two-dimensional Gaussian blur, i.e. we need to solve an integral equation of the first kind of the form

$$\mathcal{K}u(x,y) = \int_\Omega k(x-x', y-y')u(x',y')dx'dy' \tag{6.9}$$

with a convolution kernel $k(x,y) = exp(-(x^2+y^2)/\sigma^2)$ with $\sigma \in]0,1[$ on the square $\Omega = [-p,p]^2$. This kernel models atmospheric turbulence blur and it is used in practice e.g. for the restoration of satellite images.
Analogously to Section 6.1, we discretize via midpoint quadrature and end

up with a positive definite BTTB matrix K having a "single zero of infinite order" at $x_0 = (\pi, \pi)$ (see e.g. [19], sec. 4.4, or [91], ch. 2):
To see this let us again work with the uniform mesh size $h = \frac{2p}{n}$ and the midpoints

$$x_i = -p + (2 * i - 1) * \frac{h}{2}, \quad y_j = -p + (2 * j - 1) * \frac{h}{2}, \qquad i, j = 1, 2, \ldots, n.$$

Then we use the midpoint quadrature rule and the two-dimensional convolution operator (6.9) translates into

$$\begin{aligned} \mathcal{K}u(x_i, y_j) &= \int_{-p}^{p} \int_{-p}^{p} k(x_i - x', y_j - y') u(x', y') dx' dy' \\ &\approx h^2 * (\sum_{s=0}^{n-1} \sum_{r=0}^{n-1} k(x_i - x_r, y_j - y_s) u(x_r, y_s)) \\ &\approx [K\mathbf{u}]_{\mathbf{i,j}} \end{aligned}$$

with the symmetric positive definite BTTB matrix

$$K = h^2 * BTTB([\mathbf{k_{1-n}}, \mathbf{k_{2-n}}, \ldots, \mathbf{k_0}, \mathbf{k_1}, \ldots, \mathbf{k_{n-1}}]) \tag{6.10}$$

having the columns

$$\mathbf{k_j} = [\mathbf{k}((\mathbf{1-n}) * \mathbf{h}, \mathbf{j} * \mathbf{h}), \ldots, \mathbf{k}(\mathbf{0}, \mathbf{j} * \mathbf{h}), \ldots, \mathbf{k}((\mathbf{n-1}) * \mathbf{h}, \mathbf{j} * \mathbf{h})]^{\mathbf{T}}$$

and the vector $\mathbf{u}$ of unknowns formed by lexicographic row-wise ordering.
Anyway, as for the mathematical theory concerning regularization and the multigrid method of the second kind the treatment of the two-dimensional convolution operator (6.9) does not differ at all from the one-dimensional case.
Like in subsection 6.2.4 we can build efficient multigrid algorithms by employing conjugate gradients with either the optimal BCCB preconditioner by T. Chan and J. Olkin [31] from Remark 4 or the Block circulant extension preconditioner from Algorithm 8 as a smoother. Note that this idea can only lead to a practical $O(n \log n)$ image deblurring algorithm if we get our coarse grid operators via rediscretization. In particular, observe that the Block circulant extension preconditioner from Algorithm 8 is only defined for BTTB matrices at all.

6.3.2 Numerical results

The following tables test our algorithms for various regularization parameters. We are dealing with a discretization of the two-dimensional deblurring operator (6.9) on $\Omega = [-1, 1]^2$. Again, our W-cycle solvers employ two presmoothing and no postsmoothing steps.

Anyway, this time more different variants are to be considered than in the 1D case: Within our multigrid algorithms we wish to test two different restrictions and two different BCCB preconditioners inside our PCG smoother. Like in subsection 6.2.4 "triv" will indicate that we use trivial injection (2.26) for restriction whereas "lin" identifies the full weighting operator (2.22). As the BCCB preconditioner by T. Chan and J. Olkin [31] from Remark 4 minimizes the Frobenius norm we indicate it by "FR-CI" whereas "EX-CI" stands for the block circulant extension preconditioner. Analogously, "FR-CI-CG" and "EX-CI-CG" identifty BCCB-preconditioned conjugate gradient methods as stand-alone solvers.

The following tables 6.3a, 6.3b and 6.3c list the iteration counts for our W-cycle solvers with natural coarse grid operators for the choice $\sigma = 0.05$ in (6.9) and for the three regularization parameters $\lambda = 1e-3$, $\lambda = 1e-4$ and $\lambda = 1e-5$. The linear systems are of the form (6.4), i.e. we use L_2-based Tikhonov regularization.

number of unknowns	$64*64$	$128*128$	$256*256$	$512*512$	$1024*1024$
MG (FR-CI, triv)	6	5	4	4	4
MG (FR-CI, lin)	7	5	5	4	4
FR-CI-CG	12	12	11	11	11
MG (EX-CI, triv)	3	3	3	3	3
MG (EX-CI, lin)	3	3	3	3	3
EX-CI-CG	8	8	8	8	8

Table 6.3a. Iteration counts for 2D deblurring problem (6.9) with $\lambda = 1e-3$.

number of unknowns	$64*64$	$128*128$	$256*256$	$512*512$	$1024*1024$
MG (FR-CI, triv)	15	9	6	5	4
MG (FR-CI, lin)	20	12	8	5	5
FR-CI-CG	24	25	25	25	25
MG (EX-CI, triv)	7	6	5	5	4
MG (EX-CI, lin)	8	7	6	4	4
EX-CI-CG	18	17	17	17	17

Table 6.3b. Iteration counts for 2D deblurring problem (6.9) with $\lambda = 1e-4$.

number of unknowns	$64*64$	$128*128$	$256*256$	$512*512$	$1024*1024$
MG (FR-CI, triv)	42	32	21	14	9
MG (FR-CI, lin)	68	30	20	24	21
FR-CI-CG	39	41	44	43	44
MG (EX-CI, triv)	22	23	18	13	7
MG (EX-CI, lin)	16	16	21	19	17
EX-CI-CG	37	40	41	42	42

Table 6.3c. Iteration counts for 2D deblurring problem (6.9) with $\lambda = 1e-5$.

From tables 6.3a to 6.3c we can observe the typical convergence behaviour of the multigrid method of the second kind like in one dimension: For fixed regularization parameter λ iteration counts decrease for larger matrix sizes. Furthermore, our multigrid algorithms can also handle very small regularization parameters λ.

Furthermore, we also observe that trivial injection for restriction does a better job than full weighting. However, the differences are a little less striking than in the one-dimensional case. We also confirm the observation by Vogel (see [112], ch. 5) that the Block circulant extension preconditioner usually gives slightly faster convergence than the optimal BCCB preconditioner from [31]. Due to its faster convergence it also gives a better smoother.

We have so far said hardly anything about H_1-based Tikhonov regularization. As already observed in [22] in one-dimensional tests the idea carries over without any problems. Anyway, this is not quite surprising: As we

know how well the discrete Laplacian can be handled by multigrid and as our algorithms use transfer operators that can also be used for the discrete Laplacian, we would be rather dismayed if H_1-based Tikhonov regularization would cause problems.

The following table 6.4 compares iteration counts for solutions of (6.9) with H_1-norm regularization. We choose $\sigma = 0.02$ in (6.9) and the regularization parameter is $\lambda = 1e-4$. In table 6.4 "pure CG" stands for the unpreconditioned CG algorithm; all the other abbreviations are as in tables 6.3a to 6.3c. Again, we work on the domain $\Omega = [-1,1]^2$.

number of unknowns	$64*64$	$128*128$	$256*256$	$512*512$	$1024*1024$
MG (FR-CI, triv)	12	9	7	6	6
MG (EX-CI, triv)	5	4	4	4	3
pure CG	25	27	25	29	34
FR-CI-CG	14	16	16	18	19
EX-CI-CG	10	11	11	12	12

Table 6.4. Iteration counts for 2D deblurring problem (6.9) with $\lambda = 1e-4$ and H_1-norm regularization.

Indeed, we have been able to develop a fast multigrid algorithm that be can used for deblurring images. Figure 6.1 shows the reconstructed solutions based on both L_2- and H_1-regularization. The original picture was subject to atmospheric turbulence blur given by the operator (6.9) with $\sigma = 0.02$ and to Gaussian white noise with a signal-to-noise ratio of 100. Our regularization parameter was $\lambda = 5 * 10^{-5}$. We solved the arising linear system via multigrid methods with natural coarse grid operators, the trivial injection as the restriction and block circulant extension preconditioned CG as the smoother.
The picture of a satellite we used is provided by the Starfire optical range, USAF Phillips Laboratory at Kirkland AFB, New Mexico. It is frequently employed as a test problem in image deblurring.

6.4 Outlook and conclusions

We have started with an algorithmic idea by R. Chan, T. Chan and J. Wan [22] and put it into the context of the multigrid method of the second

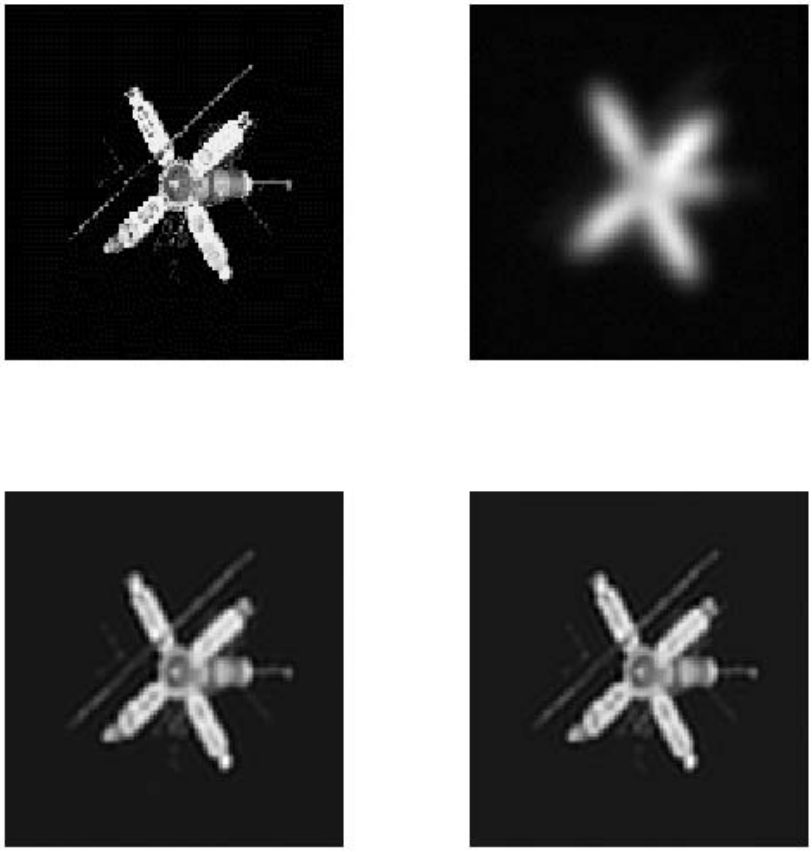

Figure 6.1: Reconstruction of an image subject to atmospheric turbulence blur (6.9) with $\sigma = 0.02$ and to Gaussian white noise with signal-to-noise ratio of 100. On the top left we see the original image, on the top right the blurred and noisy image, on the bottom left the restoration based on the L_2-norm and on the bottom right the restored based on the H_1-norm. The H_1-norm based restoration is marginally better. The regularization parameter was $\lambda = 5 * 10^{-5}$.

kind. By getting coarse grid representations via rediscretization instead of a Galerkin approach we have come up with an efficient algorithm that can be used to solve linear systems arising in image deblurring efficiently in $O(n \log n)$ speed. However, we would certainly like to be honest and express that we do not expect our multigrid solvers to succeed in the engineering practice. As it is well known – and as we also have shown in our numerical experiments – there are BCCB preconditioners for the image deblurring applications we have discussed which lead to highly efficient $O(n \log n)$ solvers if used for preconditioning conjugate gradients. Due to their simplicity, their robustness and the ease of their implementation we reckon that they will be preferable from an engineer's viewpoint to our sophisticated W-cycle solvers employing these preconditioned conjugate gradient methods as smoothers even for extremely large problems.

We have pointed out how the new algorithms can be interpreted as generalizations of the multigrid method of the second kind: In fact, preconditioned conjugate gradients can be employed as smoothers in multigrid methods for general discrete Fredholm integral equations of the second kind – not necessarily of convolution type – to give convergence in the case of very small λ. On the aspect of preconditioned conjugate gradients for smoothing it would be nice to have much more analysis of our algorithms. This will certainly be subject of our future research and we would hope to understand the new algorithms as completely as the classical variant of the multigrid method of the second kind with Richardson smoothing. However, we are aware that as soon as semi-iterative smoothers are involved, analysis is usually significantly much more difficult than in the normal case when only stationary iterative methods (2.8) are employed for relaxation. For existing work about semi-iterative smoothers in multigrid algorithms we would finally like to refer to [64], pp. 332.

To us, it was particularly interesting to understand that the inverse problems investigated in this section can not be viewed under the same terms as the multigrid methods for Toeplitz matrices generated by functions with a finite number of zeros of finite order. In fact, here we are dealing with a "zero of infinite order" which is not located at the origin – but there is an underlying analysis telling us clearly not to shift it to the origin via (4.11).

Chapter 7

A new optimal preconditioner for high-resolution image reconstruction

In this chapter we study the efficient solution of linear systems arising from the problem of high-resolution image reconstruction with multisensors: There a high-resolution image is reconstructed from four undersampled shifted, degraded and noisy low resolution images.
We start with a few general remarks on superresolution and explain why the investigated technique of high-resolution image reconstruction is important e.g. in enhancing the quality of pictures of the ground taken from a satellite. Then we introduce the mathematical model and its discretization: We will study both zero (Dirichlet) boundary conditions, i.e. a dark background around our image is assumed, as well as Neumann boundary conditions, i.e. it is assumed that the scene outside is a reflection of the original scene at the boundary. As observed by R. Chan and collaborators in [24] and [27] the Neumann boundary condition is preferable as zero boundary conditions give a ringing effect around the image.
In the ideal case when there are no calibration errors within our device of multisensors the system matrices for Dirichlet conditions will be of the form A^TA with A being a sparse ill-conditioned BTTB matrix whereas for Neumann conditions the upper left and lower right entries are changed such that A will become Block Toeplitz-plus-Hankel with Toeplitz-plus-Hankel blocks. Like in the previous chapter, we will employ the Tikhonov functional for regularization.

In [24] and [27] R. Chan and collaborators have been proposing effective preconditioners for the arising sparse linear systems in the Neumann case based on the fast cosine transform. Our goal was to develop an $O(n)$ preconditioner: We first explain why the multigrid methods from Chapter 4 can not be carried over; in the case with Dirichlet boundary conditions and no calibration errors the BTTB matrix A will be generated by a nonnegative function with an infinite number of zeros of order 2. However, there is a simpler way to devise an $O(n)$ preconditioner based on an "analytic" incomplete factorization. Finally, we will show the efficiency of our new preconditioner.

The original contributions of this chapter are the implementation of the new preconditioner, its integration into a powerful software package – which the author was handed by his friend Andy Yip during his stay at UCLA – and the numerical tests. However, the derivation of the preconditioner presented in 7.3.2 was performed by the author's advisor.

7.1 Introduction

Superresolution is a general term for restoring a high-resolution image from multiple undersampled shifted, degraded and noisy images [78], [84], [96], [109]. It has various practical applications, including aerial or facilities surveillance, consumer, commercial, medical, forensic and scientific imaging.

In the following we will study a special case of superresolution where a device consisting of a 2×2 sensor array is used to take four shifted low resolution images of the same scene. This technique has first been proposed by Bose and Boo in the paper [7] and is since then referred to as *high-resolution image reconstruction with multisensors*. The main reason for the development of the approach was that – due to hardware limitations – it may frequently not be possible to improve the resolution of an image any further. In particular, let us think of pictures of the ground taken from a satellite: Even if excellent sensors are used, it might still be difficult – if not impossible – to retrieve details on the ground. However, taking four appropriately shifted pictures of the same scene provides more spatial information and we can use this information to get an image of higher resolution as described in [7].

The technique of high-resolution image reconstruction [7] has been analyzed mathematically and numerically as well as further improved by R. Chan,

T. Chan and their collaborators in [24], [27], [29], [30] and [26]; and our presentation of the subject is very strongly influenced by their results. In particular, as we have already mentioned it was them who pointed out that it is preferable to replace the zero (Dirichlet) boundary conditions (employed in the original approach by Bose and Boo [7]) by Neumann conditions to avoid boundary artifacts around the reconstructed image.

7.2 The mathematical model and its discretization

7.2.1 Making use of low resolution images

We would like to introduce the mathematical model rather briefly here. More details can be found in [7] or [24].

We work with a sensor array – or *multisensor* – with 2×2 sensors: Each sensor has $N \times N$ sensing elements, i.e. pixels, and the pixel size is $T \times T$. The purpose is to restore an image of higher resolution $M \times M$ with $M = L \times N$.

Certainly, we can only gain a higher spatial resolution out of the four low resolution images if the sensors are shifted appropriately from each other, i.e. there need to be subpixel displacements between the sensors. In the ideal case the sensors are shifted from each other by a value proportional to $(T/2) \times (T/2)$. However, in practice small perturbations around these ideal subpixel locations may occur due to imperfection of the mechanical imaging system. Fortunately, these so-called *calibration errors* can be detected and measured by the manifacturer during the camera calibration process.
Let us place the reference sensor in the position $[0, 0]$. Then for $k, l \in \{0, 1\}$ with $(k, l) \neq (0, 0)$, the horizontal and vertical displacements $d^x_{k,l}$ and $d^y_{k,l}$ of the $[k, l]$-th sensor array with respect to the reference sensor are given by

$$d^x_{k,l} = \frac{T}{2}(k + \epsilon^x_{k,l}) \qquad \text{and} \qquad d^y_{k,l} = \frac{T}{2}(l + \epsilon^y_{k,l})$$

with $\epsilon^x_{k,l}$ and $\epsilon^y_{k,l}$ denoting the normalized horizontal and vertical displacement errors. We can assume that

$$|\epsilon^x_{k,l}| < \frac{1}{2} \qquad \text{and} \qquad |\epsilon^y_{k,l}| < \frac{1}{2}$$

for otherwise the low resolution images would be overlapped so much that restoring a high-resolution image is rendered impossible (see also [7], [30]).

Anyway, in practice we may expect values for $\epsilon^x_{k,l}$ and $\epsilon^y_{k,l}$ significantly smaller in modulus than $\frac{1}{2}$.

Let f be the original scene. Then we can model the observed low resolution image $g_{k,l}$ for the $[k,l]$-th sensor by

$$g_{k,l}[m,n] = \frac{1}{T^2} \int_{T(n-\frac{1}{2})+d^y_{k,l}}^{T(n+\frac{1}{2})+d^y_{k,l}} \int_{T(m-\frac{1}{2})+d^x_{k,l}}^{T(m+\frac{1}{2})+d^x_{k,l}} f(x,y)dxdy + \eta_{k,l}[m,n] \qquad (7.1)$$

for the pixels $m, n = 1, \ldots N$, with $\eta_{k,l}$ standing for the noise corresponding to the $[k,l]$-th sensor. To be more precise: $g_{k,l}[m,n]$ denotes the average intensity registered at the $[m,n]$-th pixel for the $[k,l]$-th sensor.
Now we intersperse the four low resolution images to form a large $M \times M$ image by assigning

$$g[2*(m-1)+k, 2*(n-1)+l] = g_{k,l}[m,n] \qquad (7.2)$$

We call g the *observed high-resolution image*. It is already an improvement over the low resolution samples; however, it is still blurred. Figure 7.1 shows how a 4×4 image is formed out of the values of four coarser images of 2×2 pixels.

To obtain a better image than the observed high-resolution image from (7.2), we need to solve (7.1) for f. We shall do this using the rectangular quadrature rule: Note that this is also a reasonable approximation to the physics of our multisensor, meaning that for each pixel in the high-resolution image f we are going to restore the intensity is constant for each point within that pixel (see [7]). If we now carry out the numerical integrations in (7.1) for each of the four sensors and perform the reordering given by (7.2) afterwards, then we will obtain a system of linear equations which relates the unknown values $f[i,j]$ to the given low resolution pixel values $g[i,j]$. In the first place this linear system would be underdetermined (and not square), because some evaluations of the rectangular rule in (7.1) also involve points outside the scene. In other words: The boundary values of g are influenced by the values of f outside the scene – and in order to set up a sensible model we need to impose boundary conditions. (Finally, note that this need for boundary conditions is due to fact we are dealing with a finite-dimensional convolution operator in this model – whereas in Chapter 6 when we studied Fredholm integral equations of the first kind the convolution operators given by (6.1) and (6.9) were infinite-dimensional and hence there was no need to impose boundary conditions.)

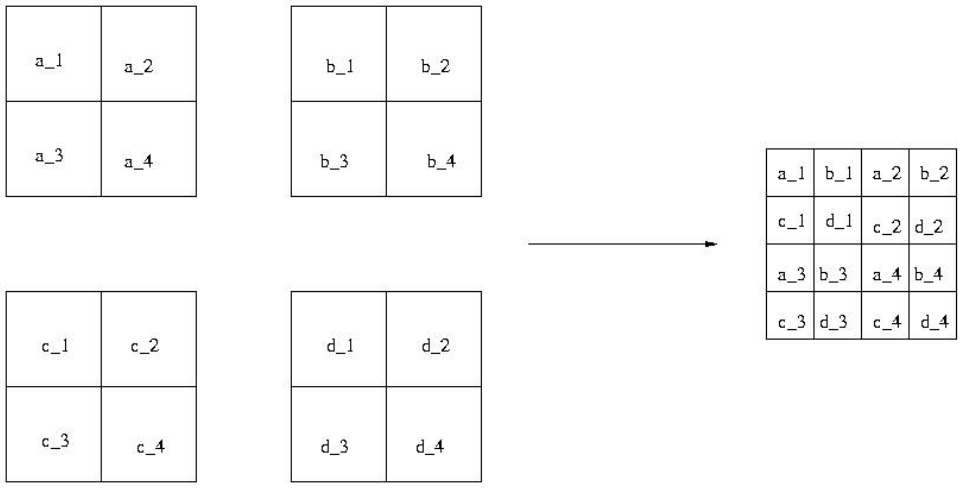

Figure 7.1: A model for the construction of the observed high-resolution image

7.2.2 Imposing boundary conditions

In the original approach Bose and Boo [7] assigned zero boundary conditions, i.e. they assumed there was a dark background outside the scene in the image reconstruction.

Let $\mathbf{g}$ and $\mathbf{f}$ denote respectively the discretizations of the observed high-resolution image g constructed by (7.2) and the high-resolution image f we want to restore; furthermore, let $\mathbf{g}$ and $\mathbf{f}$ be ordered column by column. Then under the zero boundary condition the blurring matrix belonging the $[k,l]$-th sensor can be described as

$$\tilde{\mathbf{H}}_{\mathbf{k,l}}(\epsilon) = \tilde{\mathbf{H}}^{\mathbf{x}}_{\mathbf{k,l}}(\epsilon) \otimes \tilde{\mathbf{H}}^{\mathbf{y}}_{\mathbf{k,l}}(\epsilon)$$

where $\tilde{\mathbf{H}}^{\mathbf{x}}_{\mathbf{k,l}}(\epsilon)$ is a tridiagonal Toeplitz matrix given by

$$\tilde{\mathbf{H}}^{\mathbf{x}}_{\mathbf{k,l}}(\epsilon) = \frac{1}{2}\begin{pmatrix} 1 & h^{x+}_{k,l} & & & 0 \\ h^{x-}_{k,l} & 1 & h^{x+}_{k,l} & & \\ & \ddots & \ddots & \ddots & \\ & & h^{x-}_{k,l} & 1 & h^{x+}_{k,l} \\ 0 & & & h^{x-}_{k,l} & 1 \end{pmatrix}$$

with

$$h_{k,l}^{x\pm} = \frac{1}{2} \pm \epsilon_{k,l}^{x}. \tag{7.3}$$

The tridiagonal Toeplitz matrix $\tilde{\mathbf{H}}_{\mathbf{k,l}}^{\mathbf{y}}(\epsilon)$ modelling the blur in y-direction is defined analogously. However, ringing effects will be visible at the boundary of the restored image in the case that f is not indeed rather dark at the boundary.

To overcome these inconvenient boundary artifacts, the use of Neumann boundary conditions was established in [24] and [27]: There we assume that the scene immediately outside is a reflection of the original scene at the boundary. Under the reflecting boundary condition the corresponding blurring matrices $\mathbf{H}_{\mathbf{k,l}}^{\mathbf{x}}(\epsilon)$ and $\mathbf{H}_{\mathbf{k,l}}^{\mathbf{y}}(\epsilon)$ are still tridiagonal, but the upper left and the lower right entry are changed. We can view the matrices $\mathbf{H}_{\mathbf{k,l}}^{\mathbf{x}}(\epsilon)$ and $\mathbf{H}_{\mathbf{k,l}}^{\mathbf{y}}(\epsilon)$ to have Toeplitz-plus-Hankel structure, i.e. we display them as

$$\tilde{\mathbf{H}}_{\mathbf{k,l}}^{\mathbf{x}}(\epsilon) = \frac{1}{2}\begin{pmatrix} 1 & h_{k,l}^{x+} & & & 0 \\ h_{k,l}^{x-} & 1 & h_{k,l}^{x+} & & \\ & \ddots & \ddots & \ddots & \\ & & h_{k,l}^{x-} & 1 & h_{k,l}^{x+} \\ 0 & & & h_{k,l}^{x-} & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} h_{k,l}^{x-} & 0 & & & 0 \\ 0 & 0 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 0 & 0 \\ 0 & & & 0 & h_{k,l}^{x+} \end{pmatrix}$$

with $h_{k,l}^{x\pm}$ defined as before in (7.3) and $\mathbf{H}_{\mathbf{k,l}}^{\mathbf{y}}(\epsilon)$ defined analogously. Of course, the blurring matrix belonging to the $[k,l]$-th sensor under the Neumann boundary condition is again given by the Kronecker product

$$H_{k,l}(\epsilon) = H_{k,l}^{x}(\epsilon) \otimes H_{k,l}^{y}(\epsilon).$$

Finally, the blurring matrix $\mathbf{H}(\epsilon)$ for the whole multisensor needs to be constructed from the four blurring matrices $\mathbf{H}_{\mathbf{k,l}}(\epsilon)$ of the individual sensors by

$$\mathbf{H}(\epsilon) = \sum_{k=0}^{1}\sum_{l=0}^{1} \mathbf{D}_{\mathbf{k,l}}\mathbf{H}_{\mathbf{k,l}}(\epsilon) \tag{7.4}$$

In equation (7.4) $\mathbf{D}_{\mathbf{k,l}}$ are diagonal matrices with diagonal elements equal to 1 if the corresponding component of $\mathbf{g}$ comes from the $[k,l]$-th sensor and zero otherwise, i.e. these diagonal matrices simply reflect the way the observed high-resolution image is built in (7.2). Of course, in the case of zero

boundary conditions the blurring matrix $\tilde{\mathbf{H}}(\epsilon)$ is constructed identically.

Note that the blurring matrices which came up in the modelling of this inverse problem will be highly ill-conditioned. Furthermore, they will in general be nonsymmetric and indefinite. Like in Chapter 6 we will again employ Tikhonov regularization and then our systems become

$$(\tilde{\mathbf{H}}(\epsilon)^T\tilde{\mathbf{H}}(\epsilon)+\alpha\mathbf{R})\mathbf{f}=\tilde{\mathbf{H}}(\epsilon)^T\mathbf{g} \quad \text{and} \quad (\mathbf{H}(\epsilon)^T\mathbf{H}(\epsilon)+\alpha\mathbf{R})\mathbf{f}=\mathbf{H}(\epsilon)^T\mathbf{g} \quad (7.5)$$

for Dirichlet and Neumann boundary conditions, respectively, with $\mathbf{R}$ denoting a discretization of the regularization functional in (6.3), i.e. the identity in case of the L_2-norm and the 2D discrete Laplacian with the respective boundary conditions in case of the H_1-norm.

7.3 A new $O(n)$ preconditioner

After regularization the matrices from (7.5) will no longer be "asymptotically" ill-conditioned. However, the condition numbers arising in practice will still be huge – and hence it is far from recommendable to solve the systems (7.5) via CG without a good preconditioner.
In [24] and [27] quite effective preconditioners based on the fast cosine transform have been proposed for the case of Neumann boundary conditions in (7.5). On the other hand, as soon as the fast cosine transform is involved the computational complexity for the numerical solution of the sparse linear systems (7.5) will be of order $O(n\log n)$. Hence we are interested in developing an efficient $O(n)$ preconditioner.

7.3.1 Difficulties in a multigrid approach

The first idea would be to attempt to carry over the multigrid algorithms for BTTB systems from Section 4.5. Hence let us first take a look at the ideal spatially invariant case (i.e. no calibration errors) with Dirichlet boundary conditions: Then we know from the previous section that the blurring matrix $\tilde{\mathbf{H}}(\mathbf{0})$ is given by

$$\tilde{\mathbf{H}}(\mathbf{0})=A\otimes A \quad \text{with} \quad A=\frac{1}{2}*tridiag(\frac{1}{2},1,\frac{1}{2}), \qquad (7.6)$$

i.e. $\tilde{\mathbf{H}}(\mathbf{0})$ is a s.p.d. BTTB matrix generated by the function

$$\kappa_1(x,y)=\frac{1}{4}*(1+\cos(x))*(1+\cos(y)).$$

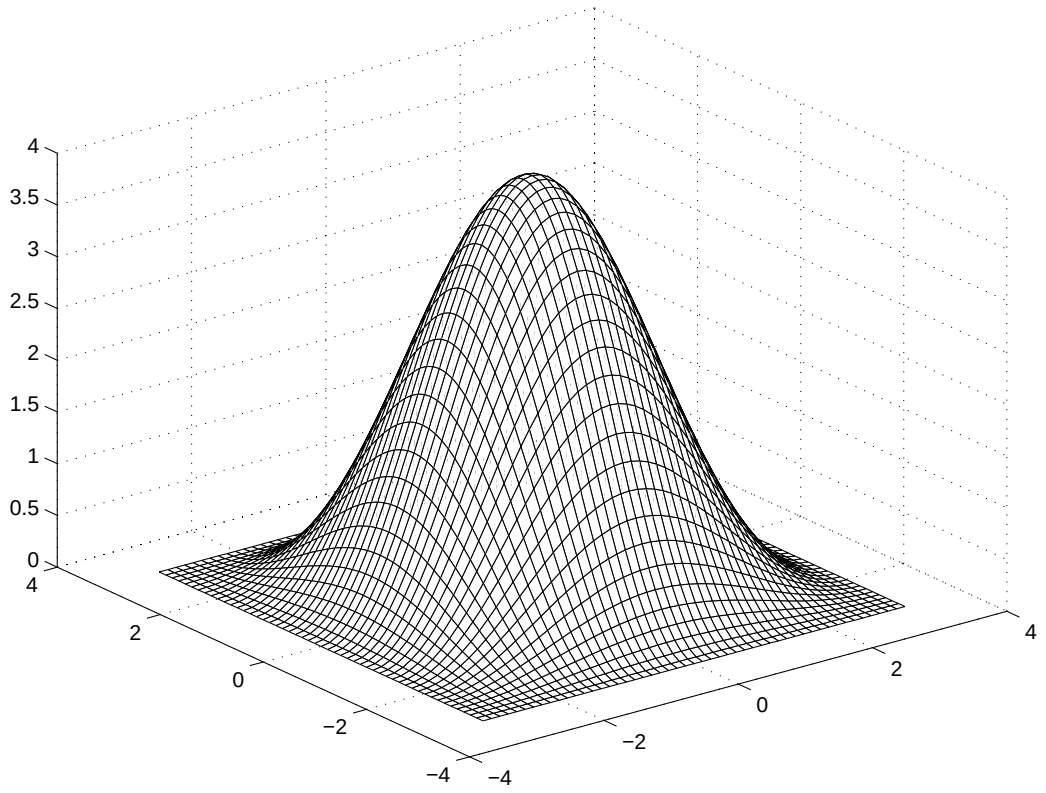

Figure 7.2: A plot of the function $\kappa_2(x, y) = (1 + \cos(x)) * (1 + \cos(y))$ on the square $\Omega = [-\pi, \pi]^2$. We see that κ_2 is zero along the lines $x = \pm\pi$ and $y = \pm\pi$.

However, this function is zero in $]-\pi, \pi]^2$ whenever $x = \pi$ or $y = \pi$, meaning that it exhibits an infinite number of zeros of order 2. In Figure 7.2 we plot the function $\kappa_2(x, y) = 4 * \kappa_1(x, y) = (1 + \cos(x)) * (1 + \cos(y))$.
Thus we have seen that our multigrid methods from Section 4.5 are not applicable in this case.

7.3.2 An efficient idea simpler than multigrid

Let us stay with the **spatially invariant case** first: It is clear that the matrix (7.6) can be inverted in $O(n)$ spead as it is a Kronecker product of a banded matrix with itself – and, of course, the same is true for its counterpart $\mathbf{H(0)}$ in the case of reflecting boundary conditions. (Remember that

there holds $A \otimes A = (A \otimes I) \cdot (I \otimes A)$ and that A is tridiagonal.)
If there are no calibrations errors, then basically the only reason preventing us from inverting the systems (7.5) directly via their Kronecker product structure is the compulsory regularization term. However, this leads to a promising idea for a preconditioner.

Let start with the L_2-based case, i.e. $R = I$ in (7.5). Then the system matrices can be written as

$$((A \otimes A)^T (A \otimes A) + \alpha * I)\mathbf{f} = (A \otimes A)^T \mathbf{g} \tag{7.7}$$

with $A = \frac{1}{2} * tridiag(\frac{1}{2}, 1, \frac{1}{2})$ in the Dirichlet case and the same A with upper left and lower right entry changed from $\frac{1}{2}$ to $\frac{3}{4}$ in the Neumann case.
A very straightward idea for a preconditioner for (7.7) making use of the Kronecker products "hidden inside" would be

$$P = (A^T A + \sqrt{\alpha} I) \otimes (A^T A + \sqrt{\alpha} I) \tag{7.8}$$

We would like to underline briefly that (7.8) is indeed a sensible choice for a preconditioner: We know that for the smallest eigenvalue of the system matrix

$$(A \otimes A)^T (A \otimes A) + \alpha * I$$

α is an extremely close lower bound, i.e. we may say it is of order $O(\alpha)$.
Now let us employ the following heuristics: Let ϱ be the smallest eigenvalue of A. If the system matrix in (7.7) and the preconditioner (7.8) were having the same eigenvectors, then the we could see from the Rayleigh quotient via

$$\frac{4 * \varrho^2 + \alpha}{4 * \varrho^2 + \alpha + 2 * \sqrt{\alpha} * \varrho^2} \longrightarrow O(\sqrt{\alpha})$$

that the smallest eigenvalue is improved to order $O(\sqrt{\alpha})$ what should lead to a significantly faster convergence of the CG algorithm. (Note that it is plain that any sensible regularization parameter α in image processing will be significantly smaller than 1.)

In the **spatially variant case** we will build our preconditioner as follows:
First of all, we sample the calibration errors in the x- and y-direction, i.e. we compute

$$\bar{\epsilon}^x = \frac{1}{4} * (\epsilon^x_{1,0} + \epsilon^x_{0,1} + \epsilon^x_{1,1}) \quad \text{and} \quad \bar{\epsilon}^y = \frac{1}{4} * (\epsilon^y_{1,0} + \epsilon^y_{0,1} + \epsilon^y_{1,1}) \tag{7.9}$$

Note that the factor $\frac{1}{4}$ reflects that there is never any calibration error connected with the $[0,0]$-th reference sensor. Afterwards we set up the matrices $\mathbf{H}(\bar{\epsilon}^{\mathbf{x}})$ and $\mathbf{H}(\bar{\epsilon}^{\mathbf{y}})$ as in 7.2.2 subject to the same boundary condition as the linear system to be solved. Then our preconditioner reads as

$$(\mathbf{H}(\bar{\epsilon}^{\mathbf{x}})^T\mathbf{H}(\bar{\epsilon}^{\mathbf{x}}) + \sqrt{\alpha} * R) \otimes (\mathbf{H}(\bar{\epsilon}^{\mathbf{y}})^T\mathbf{H}(\bar{\epsilon}^{\mathbf{y}}) + \sqrt{\alpha} * R) \tag{7.10}$$

with R denoting the discretization of our regularization operator, i.e. the identity in the L_2-based case and the discrete 1D Laplacian with corresponding boundary conditions in the H_1-based case. Note that this preconditioner can be viewed as a kind of an "analytic factorization" of the system matrices (7.5) – and that the two banded factors can be inverted quickly.

7.4 Numerical results

There are a lot of aspects we need to test in connection with our new preconditioner (7.10).

First of all, we wish to state that plenty of numerical experiments have confirmed that our way of sampling the calibration errors (7.9) is very reasonable. In particular, we have compared it to an analogous sampling with factor $\frac{1}{3}$: Although this did usually not make a big difference in the iteration counts, we were never better off than with our choice (7.9), i.e. with the factor $\frac{1}{4}$, in our tests.

In our tables we will list iteration counts of the preconditioned conjugate gradient method which we obtained employing

$$\frac{\|r^{(j)}\|_2}{\|r^{(0)}\|_2} \leq 10^{-6}$$

as the stopping, i.e. we stop iterating when the relative residual with respect to the Euclidian norm is less or equal 10^{-6}.

In the following we have tried our preconditioner for various calibration errors, for both Dirichlet and Neumann boundary conditions and for Tikhonov regularization with both the L_2- and the H_1-norm using different regularization parameters α. Again, we note that in practice it might be preferable to pick a certain criterion, like e.g. the C_L-method, yielding a regularization parameter for each individual matrix – and like in Chapter 6 we refer to the

books by Hansen [67] and Vogel [112] for different schemes to choose regularization parameters. And although we admit that our decision simply to test different regularization paramaters in our tables is slightly questionable, we wish to state that the same kinds of comparisons are being done by R. Chan and his collaborators (compare e.g. [24], [26]).

Firstly, we would like to point out that in our way to construct the preconditioner we have indeed picked an appropriate power of α in (7.10):
From an analytic viewpoint we have already pointed out in the previous section why we propose to use $\sqrt[2]{\alpha}$ in (7.10); however, there is still an important point in comparing it to other choices. In the following two tables 7.1a and 7.1b we also list the iteration numbers for the variants with α, $\sqrt[3]{\alpha}$ and $\sqrt[4]{\alpha}$ in (7.10):

number of unknowns	pure CG	(7.10) with $\sqrt[2]{\alpha}$	α	$\sqrt[3]{\alpha}$	$\sqrt[4]{\alpha}$
$16 * 16$	83	21	37	34	41
$32 * 32$	111	21	81	33	43
$64 * 64$	124	24	128	38	48
$128 * 128$	116	23	142	38	50

Table 7.1a. Iteration counts for Dirichlet boundary conditions and L_2-norm based Tikhonov regularization with $\alpha = 1e-3$. There was a uniform calibration error of $\epsilon = 0.1$.

number of unknowns	pure CG	(7.10) with $\sqrt[2]{\alpha}$	α	$\sqrt[3]{\alpha}$	$\sqrt[4]{\alpha}$
$16 * 16$	201	27	33	47	68
$32 * 32$	269	32	96	58	81
$64 * 64$	273	31	247	57	82
$128 * 128$	315	36	413	65	97

Table 7.1b. Iteration counts for Dirichlet boundary conditions and L_2-norm based Tikhonov regularization with $\alpha = 1e-4$. There was a uniform calibration error of $\epsilon = 0.1$.

From tables 7.1a and 7.1b we can see that our preconditioner (7.10) is in fact very efficient and reduces the iteration counts significantly in comparison to the unpreconditioned CG algorithm (denoted by "pure CG" in our tables). For the small regularization parameter $\alpha = 1e-4$ used in table 7.1b our

preconditioner reduces the numbers of iterations needed even by a factor of approximately 9.
Furthermore, we see that our mathematical reasoning in setting up (7.10) with respect to the power of α is strongly confirmed numerically. Iteration counts are always larger for $\sqrt[3]{\alpha}$ and they worsen more strongly for $\sqrt[4]{\alpha}$. The variant with α does not pay off at all and for large problems it might even need much more iterations than the unpreconditioned CG algorithm.

Thus in the following we shall only compare conjugate gradients with our preconditioner (7.10) and unpreconditioned conjugate gradients, abbreviated by "PCG" and "CG, respectively.

number of unknowns	$16*16$	$32*32$	$64*64$	$128*128$	$256*256$
PCG, $\alpha = 1e-3$	28	29	26	26	25
CG, $\alpha = 1e-3$	105	116	110	109	107
PCG, $\alpha = 1e-4$	35	44	44	43	42
CG, $\alpha = 1e-4$	270	282	315	286	273

Table 7.2. Iteration counts for Neumann boundary conditions and L_2-norm based Tikhonov regularization. There was a uniform calibration error of $\epsilon = 0.1$.

number of unknowns	$16*16$	$32*32$	$64*64$	$128*128$	$256*256$
PCG, $\alpha = 1e-3$	26	26	26	25	26
CG, $\alpha = 1e-3$	64	71	71	68	69
PCG, $\alpha = 1e-4$	43	43	41	41	41
CG, $\alpha = 1e-4$	145	173	174	180	173

Table 7.3. Iteration counts for Dirichlet boundary conditions and H_1-norm based Tikhonov regularization. We employed a calibration error with six different values $\epsilon_{k,l}^{x,y} \in]0, \frac{1}{4}]$.

number of unknowns	$16*16$	$32*32$	$64*64$	$128*128$	$256*256$
PCG, $\alpha = 1e-3$	27	27	27	26	26
CG, $\alpha = 1e-3$	61	62	62	60	60
PCG, $\alpha = 1e-4$	48	48	47	47	47
CG, $\alpha = 1e-4$	151	166	166	163	165

Table 7.4. Iteration counts for Neumann boundary conditions and H_1-norm based Tikhonov regularization. We employed a calibration error with six different values $\epsilon_{k,l}^{x,y} \in]0, \frac{1}{4}]$.

The above tables clearly confirm that our new preconditioner (7.10) works very well and gives optimal order convergence also in the cases of H_1-based Tikhonov regularization and larger calibration errors.

In figures 7.3 and 7.4 we use the technique of high-resolution image reconstruction for a picture of a parrot – or to be more precise: a cockatiel. We see that the Neumann boundary condition leads to slightly better restored images. However, we admit that the ringing effect resulting from the use of Dirichlet boundary conditions is hardly visible on paper, although one can clearly see it on a screen.

7.5 Conclusions

We have developed a new preconditioner for sparse matrices resulting from the problem of high-resolution image reconstruction with multisensors and shown its numerical efficiency in various tests. In this case it came out to be much simpler to devise an $O(n)$ method by using the Kronecker product structure of the linear systems involved than to go for a multigrid scheme.

It will be interesting to carry over our preconditioner to the reconstruction of colour images according to [26]: In that case the system matrices are still of the structue (7.5) within and across the red, green and blue channels. Respecting these colour channels and their combinations, the linear systems to be solved become even larger and the need for an efficient preconditioner is even more viable.
On the other hand, we agree that our preconditioner is tailored to the technique of reconstructing a high-resolution image out of low resolution images taken by a 2×2 multisensor; hence it will be difficult to carry over the same preconditioning approach to general superresolution problems.

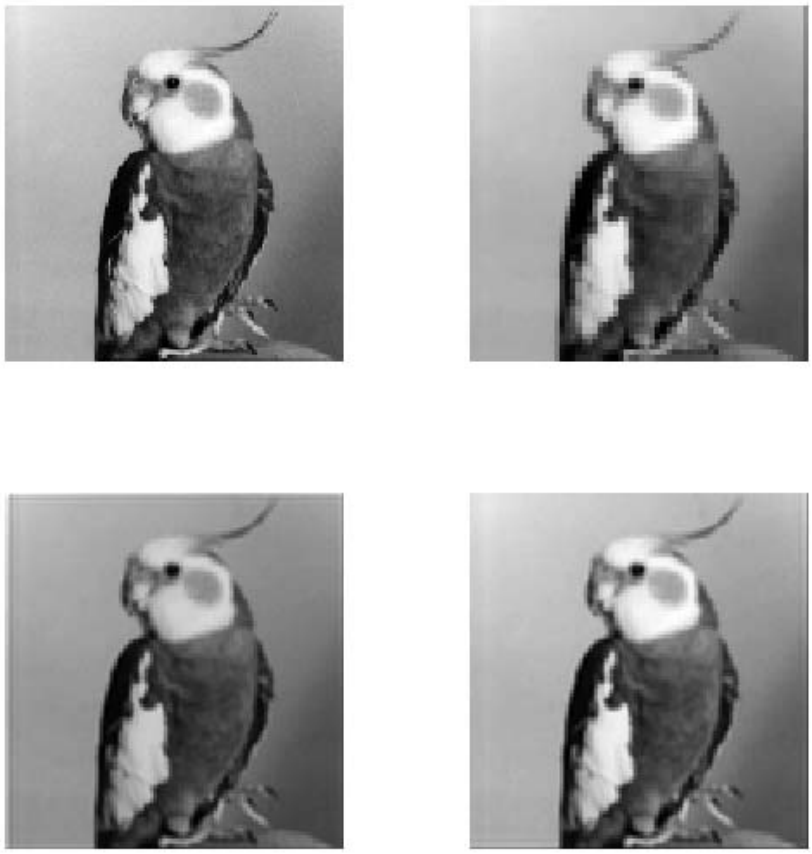

Figure 7.3: The technique of high-resolution image reconstruction applied to an image of a cockatiel. This time we use Tikhonov regularization based on the L_2-norm. On the top left we see the original image, on the top right a coarse image taken by the $[0,1]$-th sensor, on the bottom left the restoration with zero boundary conditions and on the bottom right the restoration with reflecting boundary conditions. Observe also the ringing effect for Dirichlet boundary conditions which is reasonably visible at the top and on the left of the restored image.

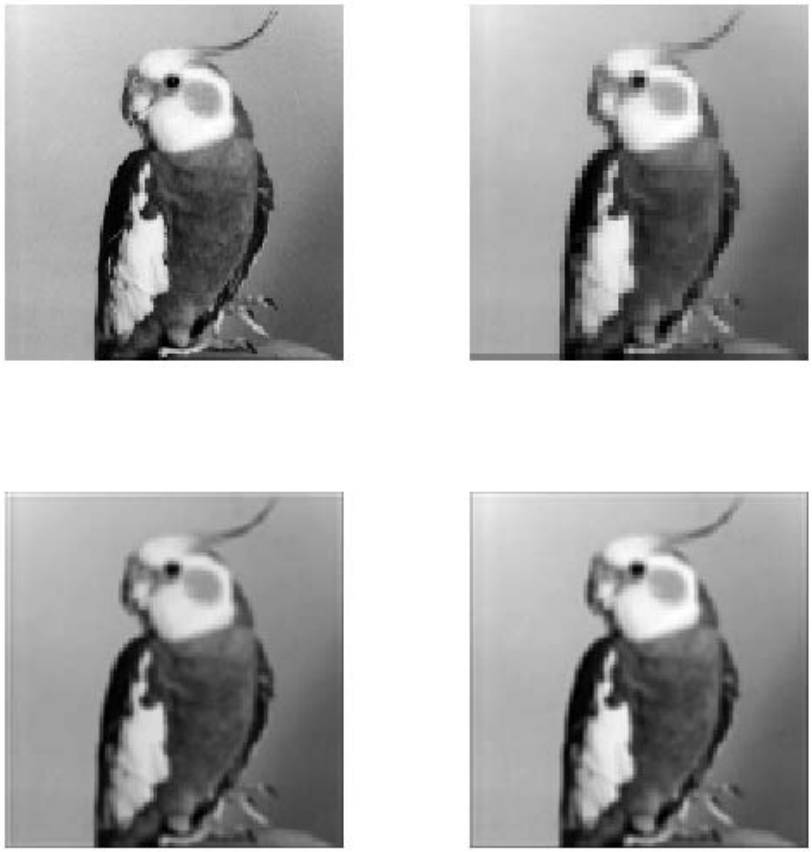

Figure 7.4: The technique of high-resolution image reconstruction applied to an image of a cockatiel. This time we use Tikhonov regularization based on the H_1-norm. On the top left we see the original image, on the top right a coarse image taken by the $[1, 0]$-th sensor, on the bottom left the restoration with zero boundary conditions and on the bottom right the restoration with reflecting boundary conditions.

Finally, from the Toeplitz point of view, we found it remarkable to get a different characterization of ill-posedness for this convolution problem than in the previous chapter. Still, in this case the zeros are only of order two – but there is an infinite number of them.

Bibliography

[1] R. Alcouffe, A. Brandt, J. Dendy and J. Painter, The multi-grid method for the diffusion equation with strongly discontinuous coefficients, *SIAM J. Sci. Stat. Comput.* **2** (1981) 430-454.

[2] O. Axelsson, Iterative solution methods, Cambridge University Press (1996).

[3] A. Bensoussan, J. Lions, G. Papanicolaou, Asymptotic analysis for peridic structure, Studies in Mathematics and Its Applications, Vol. 5, North Holland (1978).

[4] M. Bollhöfer and V. Mehrmann, Algebraic Multilevel Methods and Sparse Approximate Inverses, Preprint SFB393/99-22, Dept. of Mathematics, TU Chemnitz (1999).

[5] J. Bordner and F. Saied, MGLab: An interactive multigrid environment, In: Proc. of the Seventh Copper Mountain Conf. on Multigrid Methods, Vol. CP 3339, NASA (1996) 57-71.

[6] F. Bornemann and H. Yserentant, A basic norm equivalence for the theory of multilevel methods, *Numer. Math.* **64** (1993), 455-476.

[7] N. Bose and K. Boo, High-resolution image reconstruction with multisensors, *Int. J. of Imaging Systems and Technology* **9** (1998) 294-304.

[8] D. Braess, The contraction number of a multigrid method for solving the Poisson equation, *Numer. Math.* **37** (1981), 387-404.

[9] J. Bramble, J. Pasciak and J. Xu, Parallel multilevel preconditioners, *Math. Comp.* **55** (1990) 1-22.

[10] A. Brandt, Multilevel adaptive solution to boundary value problems, *Math. Comp.* **51** (1977) 389-414.

[11] A. Brandt, Rigorous quantitative analysis of multigrid: I. Constant coefficients two level cycle with L_2-norm, *SIAM J. Num. Anal.* **31** (1994) 1695-1730.

[12] A. Brandt and I. Livshits, Wave-ray Multigrid Method for Standing Wave Equations, *ETNA* **6** (1997) 162-181.

[13] W. Briggs, A Multigrid Tutorial, *SIAM*, 1987.

[14] O. Bröker and M. Grote, Sparse approximate inverse smoothers for geometric and algebraic multigrid, preprint(2000), to appear in Applied Numerical Mathematics .

[15] O. Bröker, M. Grote, C. Mayer and A. Reusken, Robust Parallel Smoothing for Multigrid via Sparse Approximate Inverses, *SIAM J. Sci. Comp.* **23** (2001) 1395-1416.

[16] X.-C. Cai and O.Widlund, Multiplicative Schwarz algorithms for some nonsymmetric and indefinite problems, *SIAM J. Numer. Anal.* **30** (1993), 936-952.

[17] R. Chan, Circulant preconditioners for Hermitian Toeplitz systems, *SIAM J. Matrix Anal. Appl.* **10** (1989) 542-550.

[18] R. Chan, Toeplitz preconditioners for Toeplitz systems with nonnegative generating function, *IMA J. Numer. Anal.* **11** (1991) 333-345.

[19] R. Chan and M. Ng, Conjugate Gradient Methods for Toeplitz Systems, *SIAM Review* **38** (1996) 427-482.

[20] R. Chan and M. Ng, Scientific Applications of Iterative Toeplitz Solvers, *Calcolo* **33** (1996) 249-267.

[21] R. Chan and P. Tang, Fast Toeplitz Solvers based on Band-Toeplitz Preconditioners, *SIAM J. Sci. Comp.* **15** (1994) 164-171.

[22] R. Chan, T. Chan and W. Wan, Multigrid for Differential-Convolution Problems Arising from Image Processing, In: Proc. of the Workshop on Scientific Computing, Hongkong 1997, Springer (1999) 58-72.

[23] R. Chan, T. Chan and C. Wong: Cosine Transform Based Preconditioners for Total Variation Deblurring, *IEEE Trans. Image Proc.* **8** (1999) 1472-1478.

[24] R. Chan, T. Chan, M. Ng and A. Yip, Cosine transform preconditioners for high resolution image reconstruction, *Lin. Alg. Appl.* **316** (2000) 89-104.

[25] R. Chan, Q. Chang and H. Sun, Multigrid method for ill-conditioned symmetric Toeplitz systems, *SIAM J. Sci. Comp.* **19** (1998) 516-529.

[26] R. Chan, M. Ng and W. Kwan, A Fast Algorithm for High-Resolution Color Image Reconstruction with Multisensors, In: Numerical Analysis and Its Applications, Second International Conference, NAA 2000, Rousse, Bulgaria, LNCS 1988, Springer (2001) 615-627.

[27] R. Chan, M. Ng and A. Yip, High-Resolution Image Reconstruction with Neumann Boundary Condition, In: Proceedings of the Fourth Japan-China Seminar on Numerical Mathematics, Springer (1998) 11-21.

[28] T. Chan, An Optimal Circulant Preconditioner for Toeplitz Systems, *SIAM J. Sci. Stat. Comp.* **9** (1988) 766-771.

[29] R. Chan, T. Chan, L. Shen and Z. Shen, Wavelet algorithms for high-resolution image reconstruction, preprint (2000), 24 pages.

[30] R. Chan, T. Chan, L. Shen and Z. Shen, Wavelet Deblurring algorithms for spatially varying blur from high-resolution image reconstruction, preprint (2001), 13 pages.

[31] T. Chan and J. Olkin, Circulant preconditioners for Toeplitz-block matrices, *Numerical Algorithms* **6** (1994) 89-101.

[32] T. Chan, E. Chow, Y, Saad and M. Yeung, Preserving symmetry in preconditioned Krylov subspace methods, *SIAM J. Sci. Comp.* **20** (1999) 568-581.

[33] J. Demmel, Applied Numerical Linear Algebra, SIAM (1997).

[34] J. Dendy, Black box multigrid, *J. Comput. Phys.* **48** (1982) 366-386.

[35] J. Dendy, Black box multigrid for nonsymmetric problems, *Appl. Math. Comput.* **13** (1983) 261-283.

[36] P. De Zeeuw, Matrix-dependent prolongations and restrictions in a black-box multigrid solver, *J. Comput. Appl. Math.* **33** (1990) 1-27.

[37] P. De Zeeuw, Acceleration of Iterative Methods by Coarse Grid Corrections, Ph.D. Thesis, CWI Amsterdam (1996).

[38] F. Di Benedetto, G. Fiorentino and S. Serra, C.G. preconditioning for Toeplitz matrices, *Comp. Math. Appl.* **25** (1993) 33-45.

[39] H. Elman and O. Ernst, Numerical experiences with a Krylov-enhanced multigrid solver for exterior Helmholtz problems, In: Mathematical and numerical aspects of wave propagation (Santiago de Compostela, 2000), SIAM (2000) 797-801.

[40] H. Elman, O. Ernst and D. O'Leary, A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations, *SIAM J. Sci. Comp.* **23** (2001) 1290-1314.

[41] L. Elsner, C. He and V. Mehrmann, Minimizing the condition number of a positive definite matrix by completion, *Numer. Math.* **69** (1994) 17-23.

[42] H. Engl, M. Hanke and A. Neubauer, Regularization of Inverse Problems, Kluwer Academic Publishers (1996).

[43] B. Engquist, Computation of oscillatory solutions for partial differential equations, *Lecture Notes in Mathematics* **1270** (1989) 10-22.

[44] B. Engquist and E. Luo, Multigrid methods for differential equations with highly oscillatory coefficients, in: Proc. of the Sixth Copper Mountain Conf. on Multigrid Methods, Vol. CP 3224, NASA (1993) 175-189.

[45] B. Engquist and E. Luo, Convergence of a Multigrid Method for Elliptic Equations with Highly Oscillatory Coefficients, *SIAM J. Numer. Anal.* **34** (1997) 2254-2273.

[46] G. Fiorentino, Tau Matrices and Generating Functions for Solving Toeplitz systems, Ph.D. thesis TD-4/97, University of Pisa (1997).

[47] G. Fiorentino and S. Serra, Multigrid methods for Toeplitz matrices, *Calcolo* **28** (1992) 283-305.

[48] G. Fiorentino and S. Serra, Multigrid methods for symmetric positive definite block Toeplitz matrices with nonnegative generating functions, *SIAM J. Sci. Comp.* **17** (1996) 1068-1081.

[49] G. Fiorentino and S. Serra, Multigrid methods for indefinite symmetric Toeplitz matrices, *Calcolo* **33** (1996) 223-236.

[50] D. Givoli, J. Keller, Exact non-reflecting boundary conditions, *J. Comput. Phys.* **82** (1989) 172–192.

[51] G. Golub, C. Van Loan, Matrix Computations, Third Edition, John Hopkins University Press (1996).

[52] R. Gonzalez and R. Woods, Digital Image Processing, Addison-Wesley (1992).

[53] A. Greenbaum, Analysis of a multigrid method as an iterative technique for solving linear systems, *SIAM J. Num. Anal.* **21** (1984), 473-485.

[54] A. Greenbaum, Iterative Methods for Solving Linear Systems, SIAM (1997).

[55] U. Grenander and G. Szegö, Toeplitz Forms and Their Applications, Second Edition, Chelsea (1984).

[56] M. Griebel, Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen-Transformations-Mehrgitter-Methode, TU München, Institut f. Informatik, TUM-I9007 (1990), SFB-Report 342/4/90. In German.

[57] M. Griebel, Multilevel algorithms considered as iterative methods on semidefinte systems, *SIAM J. Sci. Comput.* **15** (1994), 547-565.

[58] M. Griebel, Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen, B.G. Teubner (1994). In German.

[59] M. Griebel and P. Oswald, On the abstract theory of additive and multiplicative Schwarz algorithms, *Numer. Math.* **70** (1995), 163-180.

[60] M. Grote and T. Huckle, Parallel Preconditioning with Sparse Approximate Inverses, *SIAM J. Sci. Comp.* **18** (1997), 838-853.

[61] W. Hackbusch, Ein iterative Verfahren zur schnellen Auflösung elliptischer Randwertprobleme, Report 76-12, Institute for Applied Mathematics, University of Cologne, 1976. In German.

[62] W. Hackbusch, Multigrid Methods and Applications, Springer (1985).

[63] W. Hackbusch, Integralgleichungen, Teubner (1989). In German.

[64] W. Hackbusch, Iterative Lösung großer schwachbesetzter Gleichungssysteme, Teubner (1993). In German.

[65] W. Hackbusch, U. Trottenberg, Multigrid Methods, Springer (1982).

[66] M. Hanke and J. Nagy, Toeplitz approximate inverse preconditioner for banded Toeplitz matrices, *Num. Alg.* **7** (1994) 183-199.

[67] P.C. Hansen, Rank-Deficient and Discrete Ill-Posed Problems, SIAM (1998).

[68] P. Hemker and H. Schippers, Multiple grid methods for the solution of Fredholm integral equations of the second kind, *Math. Comp.* **36** (1981) 215–232.

[69] M.R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Stand.* **49** (1952) 409-436.

[70] R. Horn and C. Johnson, Topics in Matrix Analysis, Cambridge University Press (1989).

[71] T. Huckle, Matrix Multilevel Methods and Preconditioning, SFB-Bericht 342/11/98, Technical Report (1998), Technische Universität München. Old version of [72].

[72] T. Huckle and J. Staudacher, Matrix Multilevel Methods and Preconditioning, preprint (2000), accepted for publication in *BIT*, 20 pages. Currently available online via *http://www5.informatik.tu-muenchen.de/persons/staudacj.html*

[73] T. Huckle and J. Staudacher, Multigrid preconditioning and Toeplitz matrices, preprint, submitted to ETNA (2002), 23 pages. Also available as technical report TUM-I0202, Technische Universität München (2002), via *http://wwwbib.informatik.tu-muenchen.de/infberichte/2002/TUM-I0202.ps.gz*

[74] X. Jin, A note on preconditioned block Toeplitz matrices, *SIAM J. Sci. Comput.* **16** (1995) 951-955.

[75] X. Jin, Band Toeplitz preconditioners for block Toeplitz systems, *J. Comput. Appl. Math.* **70** (1996) 225-230.

[76] M. Kac, W. Murdoch and G.Szegö, On the extreme eigenvalues of certain Hermitian forms, *J. Rat. Mech. Anal.* **13** (1953) 767-800.

[77] R. Kettler, Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods, in: [65] (1982) 501-534.

[78] S. Kim, N. Bose and H. Valenzuela, Recursive reconstruction of high resolution image from noisy undersampled multiframes, *IEEE Trans. on Acoust., Speech and Signal Process.* **38** (1990) 1013-1027.

[79] T. Ku and C. Kuo, Design and analysis of Toeplitz preconditioners, *IEEE Trans. Signal Proc.* **40** (1992), 129-141.

[80] R. Lagendijk and J. Biemond, Iterative identification and restoration of images, Kluwer Academic Publishers (1991).

[81] A. Louis, Inverse und schlecht gestellte Probleme, Teubner (1989). In German.

[82] E. Luo, Multigrid method for partial differential equations with oscillatory coefficient, Ph.D. thesis, University of California, Los Angeles (1994).

[83] K.W. Morton and D.F. Mayers, Numerical Solution of Partial Differential Equations, Cambridge University Press (1994).

[84] N. Nguyen, Numerical algorithms for image superresolution, Ph.D. Thesis, Stanford University (2000).

[85] E. Nyström, Über die praktische Auflösung von linearen Integralgleichungen mit Anwendungen auf Randwertaufgaben der Potentialtheorie, *Soc. Sci. Fenn. Comment. Phys.-Math.* **4** (1928) 1-52. In German.

[86] E. Nyström, Über die praktische Auflösung von linearen Integralgleichungen mit Anwendungen, *Acta Mathematica* **54** (1930) 185-204. In German.

[87] P. Oswald, On discrete norm estimates related to multilevel preconditioners in the finite element method, In: Constructive Theory of Functions, Proc. Int. Conf. Varna 1991, Bulg. Acad. Sci. (1992) 203-214.

[88] P. Oswald, Multilevel Finite Element Approximation. Theory and Applications, Teubner (1994).

[89] B. Parlett, The Symmetric Eigenvalue Problem, Prentice Hall (1980).

[90] A. Reusken, Fourier analysis of a robust multigrid method for convection-diffusion equations, *Num. Math.* **71** (1995), 365-397.

[91] K. Riley, Two-level preconditioners for regularized ill-posed problems, Ph.D. thesis, Department of Mathematics, University of Montana at Bozeman (1999).

[92] J. Ruge and K. Stüben, Algebraic Multigrid, In: Multigrid Methods, S. F. McCormick (Ed.), SIAM (1987) 73-130.

[93] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, *SIAM J. Sci. Comp.* **14** (1993) 461-469.

[94] J. Schröder and U. Trottenberg, Reduktionsverfahren für Differenzengleichungen bei Randwertaufgaben I, *Num. Math.* **22** (1973), 37-68. In German.

[95] J. Schröder and U. Trottenberg, Reduktionsverfahren für Differenzengleichungen bei Randwertaufgaben II, *Num. Math.* **26** (1976), 429-459. In German.

[96] R. Schultz and R. Stevenson, Extraction of high-resolution frames from video sequences, *IEEE Trans. Image Process.* **5** (1996) 996-1011.

[97] S. Serra, Preconditioning strategies for asymptotically ill-conditioned block Toeplitz systems, *BIT* **34** (1994) 579-594.

[98] S. Serra, Conditioning and solution of Hermitian (block) Toeplitz systems by means of preconditioned conjugate gradient methods, In: Proc. in Advanced Signal Processing Algorithms, Architectures, and Implementations - SPIE conference, SPIE (1995) 326-337.

[99] S. Serra, Convergence analysis of two-grid methods for elliptic Toeplitz/PDEs Matrix-sequences, preprint (2001), submitted to *Numer. Math.*, 31 pages.

[100] S. Serra and C. Tablino Possio, Preliminary Remarks on Multigrid Methods for Circulant Matrices, In: Numerical Analysis and Its Applications, Second International Conference, NAA 2000, Rousse, Bulgaria, LNCS 1988, Springer (2001) 152-159.

[101] S. Serra and C. Tablino Possio, Multigrid methods for multilevel circulant matrices, preprint (2001), submitted to *SIAM J. Sci. Comp.*, 16 pages.

[102] B. Smith, P. Björstad and W. Gropp, Domain decomposition, Cambridge University Press (1996).

[103] G. Strang, A proposal for Toeplitz matrix calculations, *Stud. Appl. Math.* **74** (1986), 171-176.

[104] H. Sun, R. Chan and Q. Chang, A note on the convergence of the two-grid method for Toeplitz Systems, *Comp. Math. Appl.* **34** (1997) 11-18.

[105] H. Sun, X. Jin and Q. Chang, Convergence of the multigrid method for ill-conditioned Block Toeplitz systems, *BIT* **41** (2001) 179-190.

[106] A. Tikhonov and V. Arsenin, Solutions of ill-posed problems, V. H. Winston and Sons (1977).

[107] L.N. Trefethen and D. Bau, Numerical Linear Algebra, SIAM (1997).

[108] U. Trottenberg, C. Oosterlee and K. Schüller, Multigrid, Academic Press (2001).

[109] R. Tsai and T. Huang, Multiframe image restoration and registration, *Advances in Computer Vision and Image Processing* **1** (1984) 317-339.

[110] C. Van Loan, Computational Frameworks for the Fast Fourier Transform, SIAM (1992).

[111] R. Varga, Matrix iterative analysis, Prentice Hall (1962).

[112] C. Vogel, Computational Methods for Inverse Problems, book in preparation, to be published with SIAM (2002). Currently available online via *http://www.math.montana.edu/∼vogel/*

[113] W. Wan, Scalable and multilevel iterative methods, Ph.D. thesis, University of California, Los Angeles (1998).

[114] H. Widom, Toeplitz matrices, In: Studies in real and complex analysis, I. Hirshman Jr.(Ed.), Math. Ass. Am.(1965).

[115] G. Wittum, On the robustness of ILU–smoothing, *SIAM J. Sci. Stat. Comput.* **10** (1989) 699-717.

[116] J. Xu, Iterative methods by space decomposition and subspace correction, *SIAM Review* **34** (1992), 518-613.

[117] K. Yosida, Functional Analysis, Springer (1974).

[118] D. Young, Iterative solution of large linear systems, Academic Press (1971).

[119] H. Yserentant, Old and new convergence proofs for multigrid methods, *Acta Numerica* **2** (1993) 285-326.

[120] X. Zhang, Multilevel Schwarz methods, *Numer. Math.* **63** (1992) 521-539.